ulm university universität
uulm

Universität Ulm | 89069 Ulm | Germany

Fakultät für Ingenieurwissenschaften, Informatik und Psychologie
Institut für Datenbanken und Informationssysteme

Konzeption eines Mobile Serious Game zum Training der akustischen Lokalisierungsfähigkeit

Bachelorarbeit an der Universität Ulm

Vorgelegt von:
Alexander Böck
alexander.boeck@uni-ulm.de

Gutachter:
Prof. Dr. Manfred Reichert

Betreuer:
Dr. Marc Schickler

2019

Fassung 27. Mai 2019

Satz: PDF-LaTeX 2$_\varepsilon$

Kurzfassung

Die Digitalisierung unseres täglichen Lebens schreitet Jahr für Jahr weiter fort. Mobile Endgeräte wie Smartphones sind mittlerweile für den Großteil der Bevölkerung nicht mehr wegzudenken. Eine Vielzahl an Applikationen vernetzen uns mit unseren Freunden und Familie, planen unseren Alltag, überbrücken Langeweile oder unterstützen uns sogar bei gesundheitlichen und medizinischen Anliegen. Gerade letzteres hat in den vergangenen Jahren vieles zunehmend vereinfacht: Medizinische Apps bestehen nicht mehr nur aus Umfragen und Fragebögen, sondern unterstützen auch bei schwersten Leiden und bisher unheilbaren Krankheiten. Ihre Aufgaben liegen hierbei im Helfen beim Bewältigen und leichteren Leben mit den Einschränkungen. Leiden, die immer mehr zunehmen, sind vor Allem auditive Störungen.

Diese Arbeit zielt darauf ab ein mobiles Spiel zu entwickeln, welches das Training und die Therapien zur Verbesserung des Gesundheitszustandes unterstützen soll. Mit der Lokalisierung von Geräuschquellen im Raum wird das Richtungshören gefördert. Zusätzlich soll das Spielen durch die Stimulation des auditiven Zentrums dazu beitragen Hörleiden zu vermindern oder zumindest zeitweise auszublenden. Die Erkenntnisse aus der Umsetzung und Anwendung des Spieles werden am Ende evaluiert und eventuelle Ausblicke und Verbesserungen aufgezeigt.

Danksagung

An dieser Stelle möchte ich mich zu aller erst bei *Herrn Prof. Dr. Manfred Reichert* für die Begutachtung meiner Bachelorarbeit bedanken. Ebenso geht ein besonderer Dank an *Herrn Dr. Marc Schickler*, der mich als Betreuer jederzeit tatkräftig unterstützt hat.

Inhaltsverzeichnis

1

Einleitung

Um den Stellenwert des Hörens bei unserer Wahrnehmung, und damit auch in unserem Leben, einzuschätzen, reicht es vor die Tür zu gehen und die Augen nur einmal für eine Minute zu schließen. Man bekommt unzählige Eindrücke seiner Umwelt ohne sie zu sehen. Die Geräusche von Autos, die vorbeifahren, Menschen, die sich einige Meter entfernt unterhalten, Vögel, die in der Luft zwitschern und sogar der Wind in den Bäumen. Es gibt aber auch immer mehr Menschen, bei denen Schwierigkeiten bei der Gehörwahrnehmung auftreten oder diese zum Teil durch Erkrankungen wie AVWS (Auditive Verarbeitungs- und Wahrnehmungsstörung) beeinträchtigt oder durch Tinnitus überlagert wird.
Ähnlich wie bei „Using Mobile Serious Games in the Context of Chronic Disorders“ [1] sollen Smartphonenutzer mit einer Anwendung ihren Gesundheitszustand verbessern können. Bei diesem Spiel soll aber das spielerische Training der Gehörwahrnehmung, insbesondere des Richtungshörens, im Vordergrund stehen. Als Inspiration diente das Spiel „A Blind Legend “, welches sowohl für PC als auch mobil verfügbar ist. Dieses Spiel wurde als Audio-Only Game entwickelt, um auch Sehgeschädigten die Erfahrung eines hochqualitativen Videospiels zu ermöglichen. Dadurch, dass es keine Grafiken gibt, wird dem Nutzer nur durch verschiedene Sounds die Spielwelt dargestellt, in der man als blinder Ritter zusammen mit seiner Tochter Abenteuer erlebt. Mit den Richtungspfeilen der Tastatur bestimmt man, in welche Richtung sich der Charakter dreht, um zu kämpfen oder zu bewegen [2]. In den letzten Jahren wurden Anwendungen mit medizinischem Hintergrund für Android, iOS und andere Plattformen programmiert. Dieses Projekt wird mit der Game Engine Unity3D plattformübergreifend umgesetzt.

1.1 Ziel der Arbeit

Mit dieser Arbeit wird ein mobiles, plattformübergreifendes medizinisches Serious-Game entwickelt, welches spielerisch die 3D-Gehörwahrnehmung trainieren soll. Umgesetzt wird das Projekt mit der Game-Engine Unity3D. Der virtuelle Raum der Engine lässt es zu, verschiedene Geräusche um den im Mittelpunkt des Raumes stehenden Spieler zu platzieren. Der Spieler muss daraufhin die Richtungen, aus denen er besagte Töne hört, bestimmen. Durch eine minimalistische, aber initiativ bedienbare, Benutzeroberfläche ist die Positionsbestimmung einfach gestaltet. Mit verschiedenen Einstellungsmöglichkeiten und Spielmodi wird so die auditive Wahrnehmung stimuliert, trainiert und sensibilisiert. Zuletzt können die gespeicherten Spieldaten vom Nutzer ausgelesen werden oder für eine Archivierung und Auswertung als CSV-Datei exportiert und somit auch in Tabellenform verfügbar gemacht werden.

1.2 Struktur der Arbeit

In insgesamt sechs Kapitel unterteilt, werden die relevanten Aspekte dieser Arbeit ausgearbeitet. Um die medizinischen Hintergründe und technischen Aspekte besser zu verstehen werden zu Anfang, im zweiten Kapitel, die notwendigen Grundlagen vorgestellt. In Kapitel 3 wird die Planung des Projektes und die einzelnen Phasen des Vorgangs zusammengefasst. Anschließend wird in Kapitel 4 die Architektur und die Implementierung des Programms genauer beschrieben. Die fertige Anwendung wird danach im fünften Kapitel, in Hinsicht auf Umsetzung der Anforderungen, evaluiert. Am Ende in Kapitel 6 werden die Ergebnisse der Arbeit noch einmal im ganzen überblickt und ein kurzer Ausblick mit möglichen Erweiterungen aufgezeigt.

2

Grundlagen

In diesem Kapitel werden sämtliche Bereiche aufgegriffen und erläutert, welche für Hintergründe und Zusammenhänge der Arbeit nötig sind. Zuerst wird die Gehörwahrnehmung mitsamt ihrer möglichen Einschränkungen und Erkrankungen thematisiert. Danach werden die Komponenten, welche zur Umsetzung des Programms eingesetzt werden, vorgestellt.

2.1 Die auditive Lokalisierungsfähigkeit

Der für diese Arbeit relevante Aspekt der Lokalisierungsfähigkeit ist das Richtungshören. Darunter versteht man mit Hilfe des Gehörs Schallrichtungen sowohl in der Horizontal- wie auch in der Vertikalebene zu orten [3]. Durch die zusätzliche Fähigkeit des Entfernungshörens entsteht dadurch das räumliche Hören, also die Ortung einer Schallquelle im Dreidimensionalen. Die reine Unterscheidung, ob ein Geräusch von Rechts oder Links kommt, nennt man Lateralisation [4]. Innerhalb von nur zehn Mikrosekunden können Geräusche registriert werden und bei der Ortung eines Geräusches können laut Literatur (Blauert 1974 [3]) Genauigkeiten erreicht werden, welche nur ein Grad Abweichung aufweisen. Diese Genauigkeit nimmt aber rapide ab, desto schlechter die Bedingungen (Frequenz, Richtung, usw.) sind.

Das Richtungshören und räumliche Hören ist im alltäglichen Leben beinahe unersetzlich. Denn es lässt uns nicht nur Gefahrensituationen früher und auch ausserhalb unseres Sichtfeldes erkennen und orten. Zusätzlich dazu ist es so, dass viele Experten auf diesem Bereich das Richtungshören eindeutig mit dem Sprachverstehen (vor allem unter

Störgeräuschen) in Verbindung bringen. Ohne diese Fähigkeit fehlt dem Menschen die Möglichkeit zwei Schallquellen zu differenzieren und einzeln auszuwerten [5].

2.1.1 Einschränkungen der Lokalisierungsfähigkeit

Die Lokalisierungsfähigkeit kann durch verschiedene Faktoren wie Krankheit, Alter, Reizentzug und Vieles mehr beeinträchtigt sein. Einige dieser Möglichkeiten werden in diesem Abschnitt vorgestellt.

2.1.2 AVWS

Die Auditiven Verarbeitungs- und Wahrnehmungsstörungen, kurz AVWS, welche auch oft als „Zentrale Hörstörung “ bezeichnet werden, tritt bei ungefähr 2 bis 3 % aller Kindern und etwa 10 bis 20 % älterer Erwachsener auf und nimmt in der Zahl zu. Ursachen lassen sich oft nur vermuten und sind nur schwer nachzuweisen [6]. Symptome einer AVWS sind unter anderem Schwierigkeiten beim Richtungshören, welche bei den Betroffenen die Analyse, Merkfähigkeit, auditive Diskrimination, wie auch Sprachverständnis im Störgeräusch beeinträchtigt. Dies führt meist zu allgemeinen Lern- und Leistungsstörungen [7].

Zur Therapie wird empfohlen, dass ein isoliertes Training der Hörfunktion nicht ausreicht und daher auch alle anderen Sinne kompensatorisch mitgeschult werden sollten.

Beim Hörtraining selbst sollte Wert auf das lateralisierte Hören, Ordnungsschwellentraining, Differenizierungstraining, sowie Konzentrationsübungen gelegt werden [7]. Beispielsweise beschreibt Hendrik Berssenbrügge eine Therapiemethode, welche darauf abzielen sollte die betroffenen Teilfunktionen (z.B. Lokalisation, Differenzierung) zu trainieren [8].

2.1.3 Schwerhörigkeit

Es gibt verschiedene Arten der Schwerhörigkeit, jedoch nur einen der jeden Menschen betrifft. Während des Prozesses des Alterns kann die auditive Wahrnehmung stark

nachlassen, diesen Vorgang nennt man Presbyakusis, oder Altersschwerhörigkeit. Vor allem verschlechtert sich die Fähigkeit Töne in der oberen Hörfrequenz wahrzunehmen, worunter die Lokalisierungsfähigkeit unweigerlich leidet [9].

Weder durch Training, noch medikamentös kann dieser Prozess aufgehalten werden. Jedoch kann das Gehör immer noch durch regelmäßige Übung und Stimulation das gezielte Hören erlernt und verbessert werden [10].

2.1.4 Tinnitus

Der Tinnitus wird beschrieben als ein Geräusch das vom Betroffenen wahrgenommen wird (meist ein Piepsen oder Pfeifen). Diese Geräusche sind nicht auf akustische Signale aus der Umwelt zurückzuführen [11]. Durch die Überlagerung der Höreindrücke wird auch hier das Richtungshören beeinträchtigt.

Zur Behandlung wird unter anderem die Tinnitus-Retraining-Therapie zur eigenständigen Therapie genannt. Hierbei soll mit Hilfe von Beschallung die gesamte Aufmerksamkeit des Hörsystems beansprucht werden. Somit wird erreicht, dass der Betroffene den Tinnitus vorübergehend nicht mehr wahrnimmt. Die Therapie kann aber auch so weit führen, der Störton irgendwann ganz verschwindet. Mit dieser Therapiemethode beschäftigen sich zu dem in der Einleitung bereits Erwähnten [1] unter anderem auch Arbeiten wie „Using Wearables in the Context of Chronic Disorders - Results of a Pre-Study “ [12].

2.1.5 Weiteres

Im Allgemeinen geht die Forschung davon aus, dass das Richtungshören erst trainiert werden muss um richtig zu funktionieren. Es setzt sich größten Teils aus Konzentration und Erfahrungsschatz zusammen, beides kann durch Übung verbessert werden [9].

Das Richtungshören kann nicht nur durch das Alter, Störungen und Krankheitsbilder beeinträchtigt sein, sondern auch durch fehlende oder inadäquate akustisch Angebote. Dies beinhaltet unter Anderem nicht vorhandene Hörreize um daran zu lernen und Erfahrungen zu sammeln, aber auch Lärmbelästigungen und Reizüberflutungen [13].

2.1.6 Das Training

Die mit dieser Arbeit entstehende Anwendung versucht verschiedene Konzepte und Therapieansätze als mobiles Spiel umsetzen [14]. Somit soll durch die Verwendung spielerisch das Richtungshören der Anwender verbessert werden. Die Anwendung kann und soll keinesfalls die professionelle Behandlung von Experten ersetzen, sondern ist lediglich dafür gedacht eine zusätzliche Übungsplattform für Betroffene aber auch gesunde Menschen zu bilden, die ihre Gehörwahrnehmung verbessern wollen. Bei der Umsetzung wurden unter Anderem folgende Methoden berücksichtigt, welche sich mit den Behandlungsmethoden der vorher genannten Einschränkungen überschneiden sollen:

Die Lateralisation, welche eine immer wiederkehrende Behandlungsmethode für verschiedene Einschränkungen des Richtungshörens ist. Hierbei „wandern “ verschiedene Geräusche (Musik, Sprache, o.ä.) durch den Kopfhörer zwischen den Ohren hin und her, welche unterschieden werden müssen. Die Identifikation und Lokalisation von Geräuschen, welche oft in der logopädischen Behandlung angewandt wird. Hier werden Geräusche abgespielt, welche der Patient zuerst erkennen und dann die Quelle orten muss [6].

Durch diese verschiedenen Trainingsmethoden sollen Nervenstrukturen stimuliert und die Zusammenarbeit der beiden Hirnhälften verbessert werden. Die positiven Auswirkungen dieses Trainings schlagen sich in Aufmerksamkeit, Richtungshören, Merkfähigkeit und Unterscheidungsfähigkeit nieder [15].

2.2 Unity 3D

Spieleengines sind Frameworks die darauf spezialisiert sind den Spielverlauf, wie auch die visuelle Darstellung von Computerspielen zum Leben zu erwecken. Im Allgemeinen hat jede Engine ihre eigene Entwicklungsumgebung mit allen nötigen Werkzeugen zum verwirklichen von den meisten Konzepten [16]. In der heutigen digitalen Welt sind sie

nicht mehr wegzudenken. Immer mehr übernehmen sie auch weitere Sektoren der Softwareentwicklung über Spiele hinaus. Ob nun bei Automotive, Film, Architektur und vielen weiteren Bereichen, unterstützen solche Engines das kreative Schaffen. Unity ist eine der weltweit führenden Echtzeit-Spieleentwicklungsplattformen. Mit ihr sind circa die Hälfte aller zur Zeit auf dem Markt verfügbaren Spiele entstanden. Ihre weite Verbreitung ist aber nicht nur ihrer freien Verfügbarkeit, sondern auch der Vielzahl an Werkzeugen und Effizienz der Arbeitsabläufe geschuldet. Ein ebenso großer Vorteil von Unity ist das plattformübergreifende Anwendungsgebiet der damit entwickelten Programme. Auf mehr als 25 Plattformen können die Anwendungen installiert werden, unter Anderem Android und iOS, welche in diesem Projekt benutzt werden [17].

2.2.1 Scenes

Die „Szenen“ in Unity könnte man auch als die einzelnen Instanzen, Teilbereiche oder Level des Spiels bezeichnen. Sie enthalten die jeweiligen Umgebungen oder Menüs und sind daher, ähnlich wie Androids Activities [18].

2.2.2 Prefabs

Die wörtliche Übersetzung ist „vorgefertigt“ , das die Funktion eines Prefabs schon gut beschreibt. In Unity lassen diese Systeme zu Spielobjekte zu erstellen, konfigurieren und zwischenzulagern. Sie bilden die wieder benutzbaren Elemente, die innerhalb einer oben beschriebenen *Scene* instanziiert werden können und den Inhalt dieser bilden [19].

3

Anforderungen

Dieser Teil der Arbeit zielt darauf ab, den ersten Teil der Planungsphase des Projektes darzustellen. Jede Anwendung hat verschiedene Anforderungen, die sie erfüllen soll, diese werden hier genannt und erklärt. Der erste Punkt behandelt die funktionalen Anforderungen, welche abklären, welche *Funktionen* das Programm besitzen soll. Der zweite Punkt das, auf das bei der Qualitätssicherung geachtet werden muss.

3.1 Funktionale Anforderungen

Die Hauptaufgabe und somit auch das Hauptaugenmerk wurde so gesetzt, dass der Nutzer mit Hilfe der Anwendung spielerisch seine auditive Lokalisierungsfähigkeit trainieren kann. Dies beinhaltet nicht nur eine geeignete auditive Stimulierung, sondern auch einen angemessenen Funktionsumfang. Die Bewertung der Prioritäten der einzelnen Bereiche sind in Tabelle 3.1 mit Sternen(★) dargestellt. Hierbei stehen drei Sterne für Maximum (unverzichtbar) und ein Stern für das Minimum (unwichtig).

Anforderung	Priorität
Benutzer	★
Statistiken	★★★
Anpassungsmöglichkeiten	★★
Spielmodi	★★
Schwierigkeitsgrade	★★★
Datenexport	★★
Hilfe	★
Richtungshören	★★★

Tabelle 3.1: Funktionale Anforderungen

Benutzer

Da das Projekt als Offlineanwendung konzeptioniert wird, ist ein vollwertiges Benutzerkonto mit Passwort nicht notwendig. Damit aber trotzdem später zur Datenauswertung zwischen den Nutzern - oder zumindest ihrer gespeicherten Daten - unterschieden werden kann, soll es eine Möglichkeit geben, seinen Benutzernamen anzugeben und zu ändern.

Statistiken

Einer der Wichtigsten Bestandteile des Programms ist das Aufzeichnen und Anzeigen von verschiedenen Statistiken. Hiermit sind unter anderem die Rekordzeiten oder Highscores der einzelnen Modi gemeint, aber auch die Fehlerquote und Abweichungen der Fehlversuche in Grad. Für den Nutzer sollen diese Werte stets einsehbar sein und somit den Spielspaß erhöhen und ebenso einen besseren Wiederspielwert garantieren. Zusätzlich soll man durch diese Funktion seine Fortschritte und Verbesserungen, die durch das Training hervorgerufen werden, einsehen können.

Anpassungsmöglichkeiten

Um für den jeweiligen Nutzer ein am besten zugeschnittenes Spielerlebnis zu garantieren, soll das Programm verschiedene Einstellungsmöglichkeiten anbieten. Dazu gehört die Anpassung der angebotenen akustischen Reize mit einem Soundpool aus dem man verschiedene Geräuschschemata wählen kann. Diese sollen Sounds aus unterschiedlichen Bereichen des Lebens, sowie Frequenzbereichen abdecken. Des weiteren sollen wählbare Spielmodi zur Verfügung stehen, um mehr Abwechslung zu garantieren. Diese sollen zusätzlich noch über einstellbare Schwierigkeitsstufen verfügen, so dass der Spieler die Anforderungen an seinen Trainingsstand angleichen kann.

Spielmodi

Ein wichtiger Aspekt eines jeden Spiels ist es, dem Nutzer mehrere Möglichkeiten zu bieten, seine Fähigkeiten zu fordern. Ein weiterer Pluspunkt für verschiedene Modi ist, dass die Abwechslung einen positiven Effekt auf den Trainingsfortschritt der Lokalisierungsfähigkeit haben kann, da somit verschiedene Aspekte des Könnens abgedeckt werden. Bei einem Wellenmodus soll eher der Spaß im Vordergrund stehen und das Gefühl eines Arcade-Spiels aufkommen. Der Spieler soll sich vor auf ihn zukommenden Gegnern (Geräuschquellen) verteidigen, indem er die entsprechende Richtung findet. Hier sollen Highscores aufgestellt werden können, indem man länger überlebt und dadurch Motivation entsteht, diesen Rekord wieder zu schlagen. Bei dem zweiten Modus, einer Zeit-Herausforderung, soll dann die Lokalisierungsfähigkeit noch explizierter geprüft und verfeinert werden. Hier soll die Richtungsbestimmung exakter ablaufen, indem der Nutzer einen Zeiger direkt auf das Ziel ausrichtet. Bei Bestätigung soll geprüft werden, ob der angegebene mit dem Zielwinkel übereinstimmt. In diesem Modus soll der Fokus nicht auf einem Highscores liegen, sondern auf der schnellsten Zielfindung mit der geringsten Abweichung.

Schwierigkeitsgrade

Um das Spiel auf das Fähigkeiten- und/oder Trainingslevel des Nutzers anzugleichen, soll dieser die Möglichkeit haben, die Schwierigkeiten der Spielmodi einzeln zu wählen. Beim vorher erwähnten *Wave Mode* werden je nach Stufe mehr oder weniger Richtungen angeboten, aus denen die Geräusche erscheinen können. Durch mehr Auswahl soll sich das Finden des Ziels schwerer gestalten, durch weniger Auswahl leichter. Der *Time Mode* soll die Abweichungen der Richtungsangaben, die der Spieler tätigt um das Ziel zu bestimmen, mit verschiedenen Toleranzen abhandeln. Je schwerer die aktive Schwierigkeitsstufe, desto weniger Grad der Abweichung zum Ziel soll das Programm zulassen, um den Versuch als Treffer zu werten.

Datenexport

Damit die gespeicherten Statistiken wie Abweichungen, Trefferwahrscheinlichkeiten und Highscores weitere Verwendung finden können, sollen diese Daten exportiert werden können. Im Grunde sollen alle Werte des Nutzer mitsamt seines Benutzernamens exportiert werden, damit sich eine Identifizierung der Daten einfacher gestaltet. Möglichkeiten für die Speicherung wären zum Beispiel Dateien im CSV (Comma-Seperated-Values) Format. Somit können die gespeicherten Werte zur späteren Archivierung oder Auwertung übersichtlich im Tabellenformat gesammelt und eingesehen werden. CSV eignet sich sehr gut, da es von vielen Tabellen-Programmen, wie beispielsweise Microsofts Excel, ohne großen Aufwand angezeigt werden kann.

Hilfe

Zur Heranführung an den Ablauf und die Funktionen des Programms soll es einen Hilfe-Modus geben. Dieser soll direkt vom Hauptmenü aus wählbar sein, damit er für den Nutzer immer sichtbar ist. Beim Start dieses Modus sollen dem Spieler Erklärungen für die jeweiligen Menüpunkte, Spielmodi, Schwierigkeiten und Einstellungsmöglichkeiten zur Verfügung stehen. Jeder dieser Punkte soll als eine Art Tutorial in Bildform mit verschiedenen Beschreibungen der Vorgänge vorliegen.

Richtungshören

Der Wichtigste Teil des Programms ist die Lokalisierung von Geräuschquellen und das damit zusammenhängende Training der 3D-Gehörwahrnehmung durch die auditive Stimulation. Notwendig für die Anwendung sollen Stereo oder Surround-Sound Kopfhörer, ein 5.1-System oder Ähnliches zur räumlichen Klangwiedergabe sein. Die verschiedenen Vorgehensweisen und Modi wurden bereits im Abschnitt 3.1.4 vorgestellt. Weitere Einstellungsmöglichkeiten, die hierfür verfügbar sein sollen wurden bereits in 3.1.3 und die anwendbaren Schwierigkeitsgrade in 3.1.5 beschrieben. Während des Spielvorgangs sollen die Geräusche im virtuellen Raum auf einem Kreis um den Spielercharakter platziert werden. Der Abstand der Geräuschquelle zum Nutzer soll immer

einen Meter betragen. Die Wiedergabe soll solange wieder gestartet werden, bis - je nach Spielmodus - die nächste Runde gestartet oder das Spiel beendet wird. Zum Orten des jeweiligen Geräusches soll auf dem Smartphonedisplay die entsprechende Richtung gewählt werden. Fehlversuche, sowie der Grad der Abweichungen sollen dabei aufgezeichnet werden, damit der Nutzer seine Versuche bewerten kann. Nach Beendigung der Runde sollen die gesammelten Daten ausgewertet und dem Benutzer die Informationen des gerade beendeten Spiels angezeigt werden.

3.2 Nicht-funktionale Anforderungen

Für die Qualitätssicherung der Anwendung wurden Kriterien festgelegt, die als nicht-funktionale Anforderungen in diesem Teil der Arbeit zusammengefasst werden. Damit dem Nutzer das bestmögliche Erlebnis bei der Benutzung des Programms garantiert werden kann, bilden die nachfolgenden Punkte die Voraussetzungen. Mit der gleichen Form der Bewertung wie bei den funktionalen Anforderungen, sind die nicht-funktionalen in Tabelle 3.2 aufgezählt.

Unterhaltungswert

Ein sehr wichtiger Punkt einer solchen Applikation ist der Trainingseffekt, der aber nur durch die regelmäßige Nutzung auftreten kann. Die Motivation, das Programm täglich zu starten ist daher ein entscheidender Faktor. Deswegen soll das Projekt als

Anforderung	Priorität
Unterhaltungswert	★★
Mobilität	★★
Zuverlässigkeit	★★★
Look & Feel	★★
Leistung und Effizienz	★★★
Sicherheit	★★★
Offline Anwendung	★★

Tabelle 3.2: Nicht-Funktionale Anforderungen

Spiel umgesetzt werden, das verschiedene Modi sowie Highscores anbietet. Durch den Spielspaß soll sich diese von anderen Medizin- und Gesundheitsapps abheben, da sie mehr Langzeitmotivation bietet.

Mobilität

Es soll keine Einschränkungen geben wo und wann diese Anwendung benutzt werden kann. Daher soll die Funktion unabhängig von Internetanbindung wie auch dem verfügbaren Platz sein. Dadurch, dass auf Virtual Reality, Cardboard oder ähnliches verzichtet wird, werden die Bewegungssensoren des Gerätes nicht gebraucht und die Position und Drehung sind daher nicht relevant. Sämtliche Steuerung soll ausschließlich über Toucheingaben stattfinden. Die einzige Hardware, die der Nutzer braucht sollen das Smartphone sowie ein geeignetes (Stereo oder Surround) Audioausgabegerät sein.

Zuverlässigkeit

Die frustfreie Benutzung soll jederzeit für den User gegeben sein. Dies soll garantiert werden, indem keinerlei Fehler oder Abstürze im System vorkommen. Sämtliche unvorhergesehene Eingaben sollen verhindert oder abgefangen werden, bevor diese zu ungewolltem Verhalten führen können.

Look & Feel

Das Design und die Navigation sollen minimalistisch und selbsterklärend gewählt werden. Dadurch soll eine intuitiv zu bedienende Oberfläche entstehen ohne unnötige Zusätze. Die Elemente sollen bekannte Dinge aus dem wahren Leben (wie z.B Kompass als Richtungsanzeiger) aufgreifen und sich innerhalb des Programms wiederholen um Wiedererkennung zu schaffen. Feedback zu den Eingaben soll der Nutzer durch verschiedene Animationen bekommen.

Leistung und Effizienz

Die Speicherung von Daten, sowie Ladezeiten zwischen verschiedenen Szenen sollen minimal gehalten werden. Dies soll ermöglicht werden, indem die von Unity gegebenen Speichersysteme benutzt werden. Ebenso sollen durch geringere Hardwarebelastung Ladezeiten optimiert und Akkuverbrauch vermindert werden.

Sicherheit

Alle Daten sollen nur innerhalb des Programms mit Unity internen Systemen gehalten werden. Ebenso sollen keinerlei Daten online verfügbar sein oder versendet werden. Einzig der Dateiexport soll Informationen außerhalb speichern. Diese aber sollen vom Nutzer anonymisiert werden können.

Offline Anwendung

Für keine Funktion der Anwendung soll eine Internetverbindung nötig sein. Alle Daten sollen lokal auf dem Gerät gespeichert und gehandhabt werden. Dies soll die Mobilität verbessern, indem die App auch an Orten ohne Internetverbindung benutzbar ist, als auch die Handhabung und Verfügbarkeit für Menschen, die keinen mobilen Internetzugang besitzen.

4

Architektur & Implementierung

Im folgenden Kapitel werden die Konzepte und Entwürfe vorgestellt, welche als Grundlage zur Implementierung gedient haben. Dazu gehören unter Anderem die verwendeten Funktionen der Unity-Engine, die Zielplattformen für welche die Anwendung entwickelt wurde und das Datenmodell und die Datenhaltung auf der das Programm aufbaut. Schließlich werden die einzelnen Szenen als Bestandteile des Programms vorgestellt und die ursprünglichen Mockups den Screenshots des Ergebnisses gegenübergestellt.

4.1 Konzept

Diese App, mit dem Namen *Blind Adventure*, ist als mobiles Serious Game mit medizinisch-gesundheitlichen Hintergrund entwickelt worden. Die Zielgruppe sollen Menschen sein, deren auditive Lokalisierungsfähigkeit eingeschränkt ist oder die, welche ihre 3D-Gehörwahrnehmung trainieren wollen. Die Nutzer können in verschiedenen Modi spielerisch ihre Fähigkeiten verbessern, indem sie sich auf ein abgespieltes Geräusch konzentrieren und dieses orten.
Beim Modus *Time* werden die Sounds zufällig 360 Grad auf einem virtuellen Kreis um den Benutzer herum instanziiert. Der Nutzer muss daraufhin, während ein Timer läuft, einen Zeiger auf dem Display in die Richtung drehen aus der das Geräusch kommt. Der vorher gewählte Schwierigkeitsgrad legt fest, wie viel Grad die Eingabe des Benutzers von der tatsächlichen Position des Ziels abweichen darf, um noch als Treffer zu zählen. Abbildung 4.1 zeigt ein Beispiel eines solchen Vorganges. Hier wäre bei *Easy* die Richtungsangabe des Nutzers richtig, aber für die anderen beiden Schwierigkeitsgrade wäre der Versuch nicht mehr innerhalb des Toleranzbereiches. Wenn das Ziel gefunden

wurde wird die benötigte Zeit, die Fehlversuche, sowie die durchschnittliche Abweichung zum Ziel angezeigt.

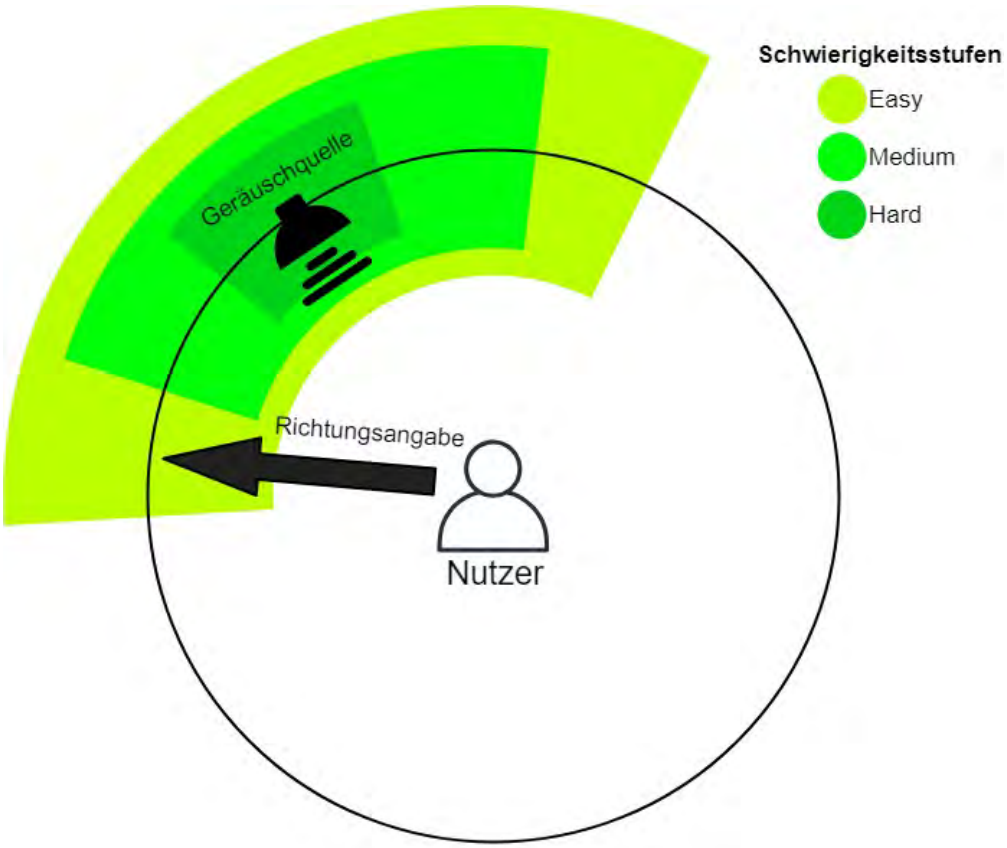

Abbildung 4.1: Aufbau des Time-Mode mit Darstellung der Abweichungstoleranzen

Beim Modus *Wave* ist der Ablauf anders aufgebaut. Das Geräusch wird zwar auch auf einem Kreis um den Nutzer positioniert, aber je nach gewähltem Schwierigkeitsgrad geschieht dies auf definierten Positionen. Je höher die Schwierigkeit, desto mehr Möglichkeiten gibt es für die Instanziierung. Bei *Easy* sind es vier (vor, hinter, links und rechts vom Benutzer), bei *Medium* acht und *Hard* zwölf Positionen. Jede Richtung in der eine Geräuschquelle sein kann wird durch einen Button auf dem Display repräsentiert. Um

den Sound zu lokalisieren muss der Spieler den entsprechenden Button drücken. Wenn der Versuch fehlschlägt wird eines von drei *Leben* abgezogen, andernfalls bekommt der Nutzer einen Punkt. Eine weitere Schwierigkeit die hinzukommt ist, dass sich die Geräusche auf den Spieler zubewegen. Sollte er es nicht schaffen das Ziel zu finden bevor es ihn erreicht, wird auch ein Leben abgezogen. In Abbildung 4.2 ist ein Beispielszenario mit Schwierigkeit *Easy* gezeigt bei dem der Sound von direkt vorne kommt. Nur der mit Grün hervorgehobene Button führt zu einem Punktgewinn statt eines Lebensabzugs. Das Spiel endet sobald der Spieler alle drei Leben aufgebraucht hat. Daraufhin werden ihm seine aktuellen Punkte und der aktuelle Highscore für die gewählte Schwierigkeit angezeigt.

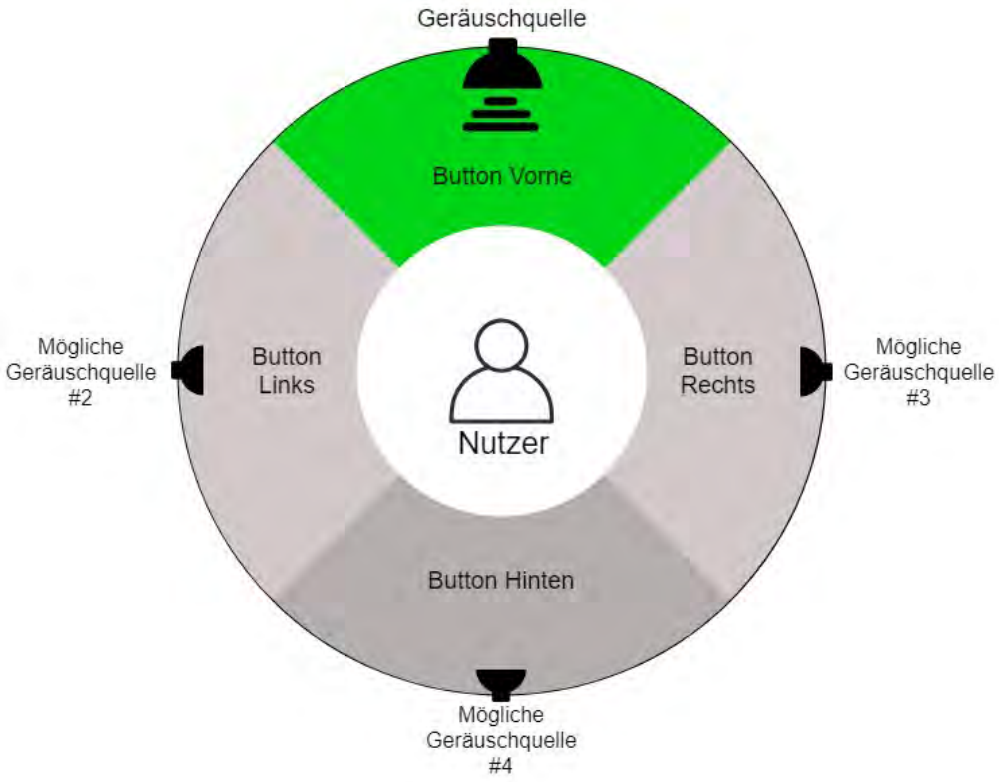

Abbildung 4.2: Aufbau des Wave-Mode mit Darstellung der Richtungsbuttons

4.2 Plattformen und Engine

Blind Adventure wurde für als plattformübergreifende Anwendung für mobile Endgeräte unter Android, iOS und Windows entwickelt. Ermöglicht wurde dies durch die Implementierung mit Unity3D in C-Sharp. Getestet wurde sowohl mit emulierter Hardware, als auch mit Android Smartphones verschiedener Versionen. Testen am PC war möglich, da keinerlei Sensoren dafür benötigt werden. Weitere Hardware waren verschiedene Kopfhörer sowie ein 5.1 System.

4.3 3D-Sound

Unity stellt eine vollwertige 3D Engine darstellt. Somit bringt sie auch sämtliche Funktionen für die Generierung von 3D Sound mit. Das Herzstück des Audiosystems bildet bei Unity der sogenannte *AudioListener*, er ist die Ausgabekomponente und somit das Verbindungsstück zwischen der Spielszene und der Hardware, die dann letztendlich den Sound wiedergibt. Je nach dem wo der *AudioListener* im virtuellen Raum platziert wird ist auch der Ort wo man den Sound wahrnimmt. Eine sogenannte *AudioSource* wiederum ist eine Komponente, die einen Sound abspielt. Auch diese kann im virtuellen Raum platziert werden um somit die Herkunft verschiedener Sounds festzulegen. Durch die Einstellung von „Spatial Blend “ auf 3D berechnet die Engine sowohl die Entfernung und Richtung des Sounds als auch auftretende Dopplereffekte bei Bewegung auf den Listener zu oder weg von ihm. In diesem Projekt wurde der *AudioListener* mittig im virtuellen Raum platziert. Die *AudioSources* werden mit verschiedenen Audioclips gespeist um daraufhin um den Listener herum instanziiert zu werden.

4.4 Datenmodell & Datenhaltung

Die Daten ,mit welchen das Programm arbeitet und verwaltet, sind in Abbildung 4.3 grafisch dargestellt. Die hier abgebildete Datenbasis wird durch das Unity-eigene Datenmodell der *Player Preferences* persistent auf dem Gerät abgespeichert. Abrufbar

sind sämtliche Daten mit Hilfe unterschiedlicher Methoden, welche ebenfalls von Unity verfügbar sind.

Eben diese Klasse `PlayerPreferences` stellt damit das Zentrum der Anwendung für die Langzeithaltung der Daten dar. Hier wird nur gepeichert, was auch zwischen Sitzungen und Neustarts nicht verloren gehen soll. Die persistente sowie die nur vorübergehend genutzten Informationen sollen somit bewusst voneinander getrennt werden. Von hier aus werden Informationen mit Hilfe von Key-Value-Paaren von anderen Klassen ausgelesen.

`Statistics` ist eine davon, sie bereitet die gespeicherten Daten zum Anzeigen vor. Hier werden die Rohdaten formatiert und als Highscores und Statistiken für die jeweiligen Modi und Schwierigkeitsstufen angezeigt.

Ähnliches passiert in `DataExport`. Hier werden alle Daten aus `PlayerPrefenences` abgerufen, welche für eine mögliche spätere Auswertung relevant sein könnten. Zusätzlich mit dem Datum werden diese Informationen dann in ein CSV-Format (Comma-Seperated-Values) gebracht. Mit Hilfe des StreamWriters schließlich als CSV-Datei exportiert und in der Ordnerstruktur des Programms abgespeichert.

Das Datenzentrum des laufenden Spiels befindet sich in den jeweiligen `Level`-Klassen hier werden alle Daten gehalten, die zur Erstellung und Ablauf einer Spielrunde gebraucht werden. Sie steuert den Radius um den Spieler und die Richtung in der die Gegner instanziiert werden. Je nach Einstellung, welche in `Settings` gespeichert werden, läuft das Level anders ab. Der *gameMode* entscheidet, ob ein `Pointer` oder ein, beziehungsweise mehrere `Buttons` erstellt werden müssen.

Die Gegner, welche erstellt werden, sind mit den Informationen in `Enemy` definiert. Hier sind die Audiodateien hinterlegt, welche abgespielt werden, während der Lokalisierung und bei einem Treffer. Welcher Sound abgespielt wird richtet sich nach dem gewählten *soundMode*. Je nach Spielmodus ist der *moveSpeed* gesetzt oder nicht.

Um die relevanten gesammelten Daten während des Spiels in die Statistiken und Highscores einfließen zu lassen, existiert die Klasse `Player` als Verbindungsstück. Hier werden auch die noch verfügbaren Leben, der aktuelle Punktestand, die gebrauchte

Zeit und der Abweichungsgrad gespeichert. Ebenso werden die Daten am Ende jedes Levels an die `GameOverPanel` weitergegeben für ein kurzes Feedback.

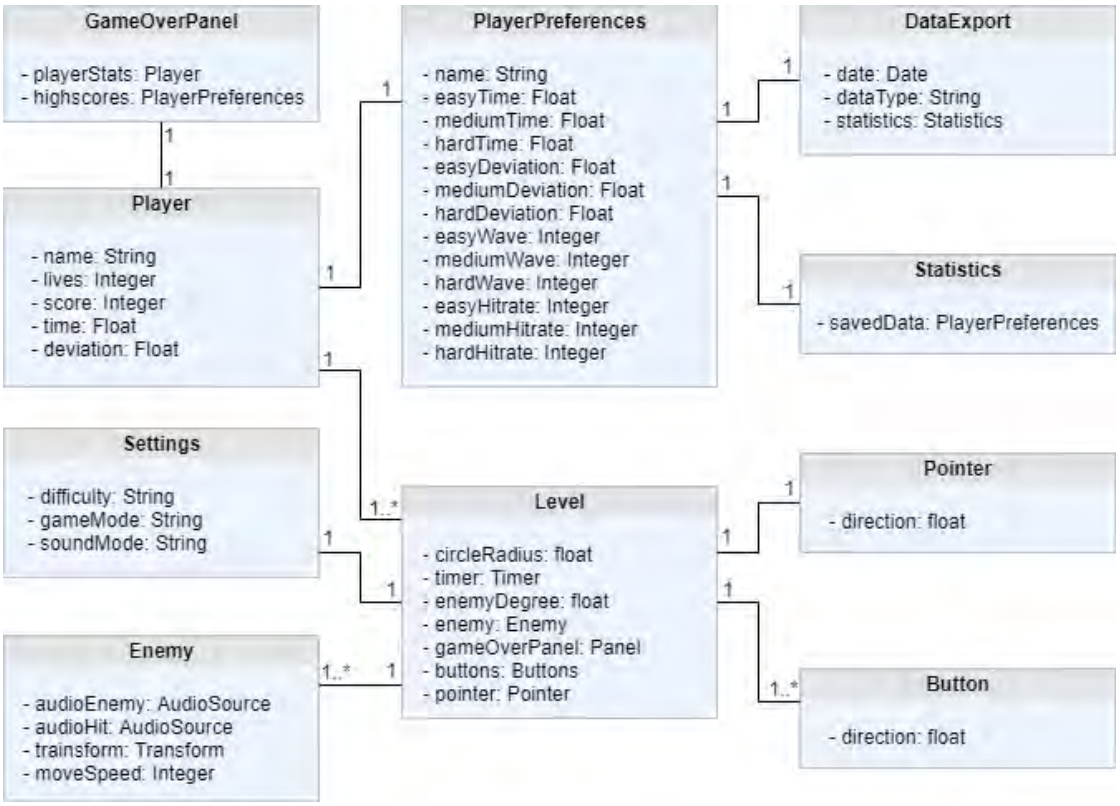

Abbildung 4.3: Datenmodell der Anwendung

4.5 Vorgehensweise

Die vorgestellte Datenverwaltung wurde mit ihren verschiedenen Instanzen durch C#-Klassen implementiert. Trotz ausgiebiger Planung und Analyse im vornherein mussten mehrere Anpassungen während der weiteren Entwicklung getätigt werden. Somit unterscheidet sich die endgültige Struktur des Programms leicht vom Entwurf.

Der Aufbau des kompletten Programms besteht aus mehreren `Scenes`, welche in Abbildung 4.4 zu sehen ist. Diese haben ihre eigenen Aufgaben und Funktionen, sowie verschiedene grafische Benutzeroberflächen. Wie die Szenen in Verbindung stehen und

designtechnisch umgesetzt wurden, wird in Folgenden beschrieben. Dazu werden die ursprünglich geplanten Oberflächen als Mockups den letztendlichen Screenshots des Spiels gegenübergestellt.

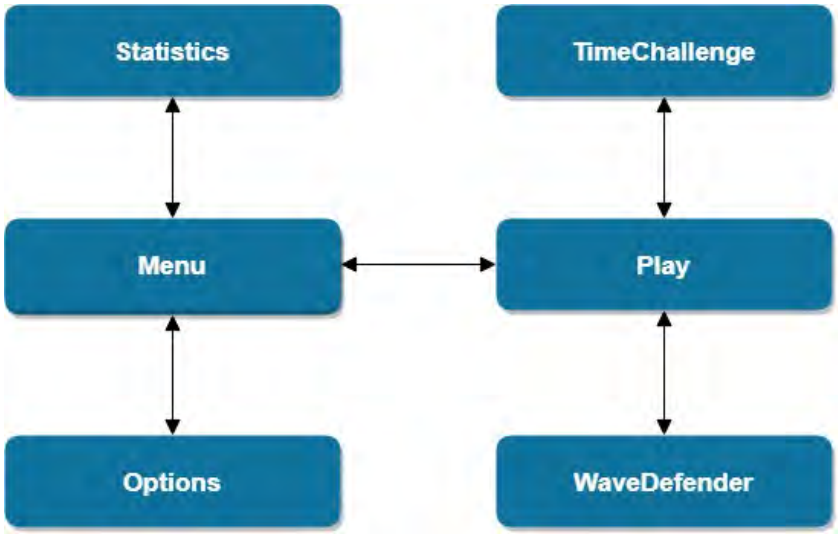

Abbildung 4.4: Aufbau der erstellten Scenes

4.5.1 Menu

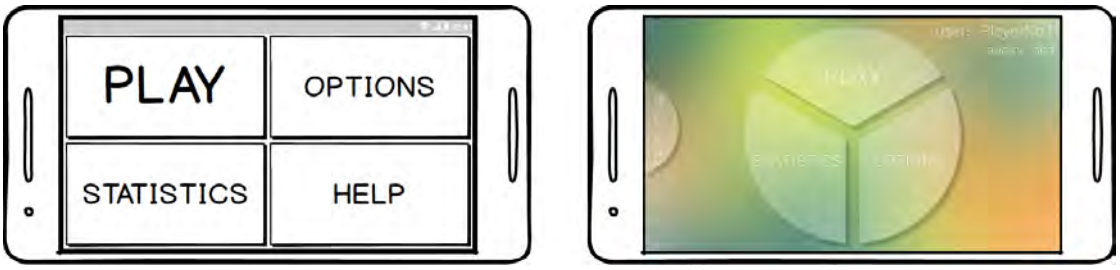

Abbildung 4.5: Mockup vs. Screenshot: Menu

Die `Menu`-Scene ist das Hauptmenü und somit der Mittelpunkt des Programms. Dieser Bildschirm ist auch das Erste, welches der Spieler nach dem Startvorgang der Anwendung sieht. Dadurch hat der Nutzer direkt einen groben Überblick über die Möglichkeiten

und Funktionen. Wie in Abbildung 4.5 zu sehen werden von hier aus verschiedene andere Szenen erreicht, darunter `Play`, `Options`, `Statistics` und `Help`. Wie auf dem Mockup zu sehen, sollten die Buttons zuerst bildfüllend quadratisch angeordnet werden. Um aber das radiale Design, das auch im Spiel selbst verwendet wird, wiederzuverwenden wurden die Buttons kreisförmig angeordnet. Somit wurde versucht etwas mehr Konstanz im allgemeinen Design zu schaffen und den Spieler daran zu gewöhnen. Das Wechseln zu den anderen Szenen funktioniert mit dem *SceneSwitcher*, welches als Prefab modelliert wurde und daher auch in anderen Szenen benutzt werden kann. Das Prefab besteht aus einem Sound für das Touchfeedback und einem Script, welches die Logik für sämtliche Szenenoperationen enthält. Per Code wird abgefragt in welche Szene gewechselt wird und mit der Unity-Methode *LoadScene* geladen (siehe Quelltext unter Listing A.1). Ebenso kann in dieser Szene die Anwendung mit dem Drücken des „back“-Buttons bei Android, und entsprechend bei anderen Betriebssystemen, beendet werden.

4.5.2 Options

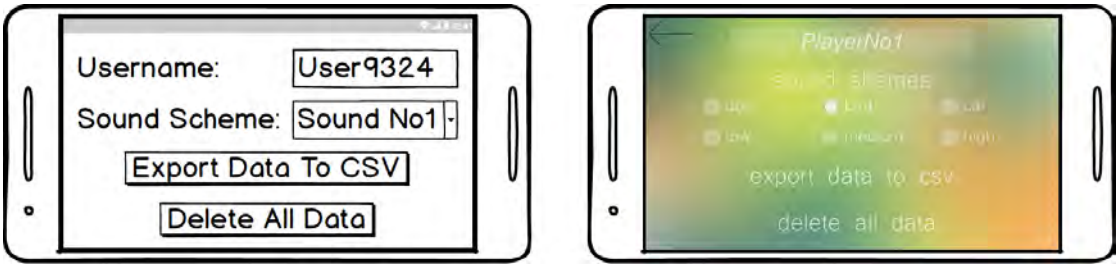

Abbildung 4.6: Mockup vs. Screenshot: Options

Die `Options`-Scene dient als Basis für Einstellungsmöglichkeiten und Anpassungen. Hier befindet sich ein Textfeld als Eingabemöglichkeit für den Benutzernamen. Nach Bestätigung des Namens wird dieser in den *PlayerPreferences*, wie in 4.4 beschrieben, als Key-Value-Paar abgelegt.

In diesem Bildschirm ist auch das Einstellen verschiedener Geräuschschemata möglich. Verfügbar sind aus dem alltäglichen Leben aufgenommene Geräusche, wie zum Beispiel

bellende Hunde, zwitschernde Vögel oder Ähnliches. Darüber hinaus gibt es aber auch digital generierte Sounds, welche spezielle Frequenzbereiche abdecken. Hierzu gehören beispielsweise besonders hohe Frequenzen für die Stimulation des Gehörs gegen Altersschwerhörigkeit. Dadurch kann ein breiteres Klientel angesprochen werden, sowie das Training weiter spezialisiert und optimiert werden.

In dieser Szene ist noch eine wichtige Funktion zu finden, nämlich das Exportieren von Daten. Durch Drücken des Buttons „Export Data “ wird im Hintergrund ein Prozess gestartet. Dieser liest sämtliche Daten aus den *PlayerPreferences* aus. Danach sortiert er diese und ordnet sie mit Kommas getrennt so an, dass sie dem CSV-Format entsprechen (Listing A.2). Mit Hilfe des Streamwriters wird eine Datei beschrieben, die mit dem Unity-File-System mit *.csv*-Endung erstellt und in der Ordnerstruktur des Programms abgespeichert wird.

Die letzte Funktion dieser Szene ist der Button „Delete Data “. Beim Aktivieren wird der Benutzername sowie sämtliche gesammelten Statistiken und Daten gelöscht. Die bereits exportierten Daten bleiben aber in der abgelegten Datei erhalten. Diese Funktion kann zum Beispiel genutzt werden, um Statistiken über einzelne Zeiträume zu sammeln oder Fortschritte genauer zu überwachen.

4.5.3 Statistics

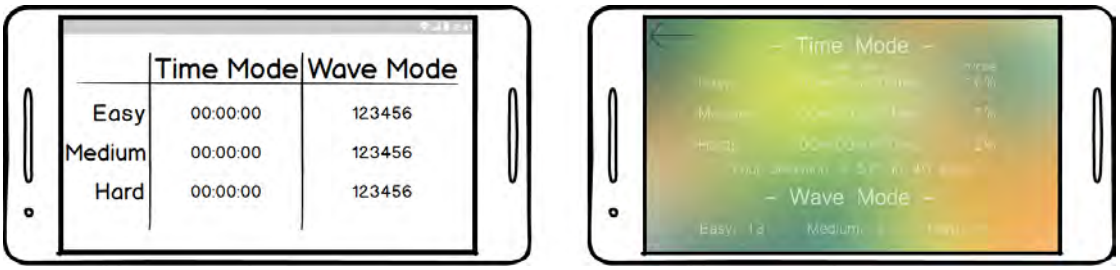

Abbildung 4.7: Mockup vs. Screenshot: Statistics

Die Szene für Statistiken ist das Datenzentrum der Anwendung. Alle bereits gesammelten Daten, Statistiken und Highscores werden angezeigt. Die Grundlage hierfür sind wie bei jeder Funktion, welche mit persistenter Datenhaltung arbeitet, die *PlayerPreferenzes*.

Aus eben diesen Daten werden die relevanten Informationen ausgelesen. Im Code in Listing A.3 ist das Auslesen der Daten eines Schwierigkeitsgrades zu sehen, ebenso wie das Formatieren dieser, damit sie für den Nutzer gut und einfach lesbar gemacht werden. Zum Beispiel sind die Zeiten in Floats abgespeichert, diese müssen zuerst in ein „Minuten : Sekunden : Millisekunden “ gebracht werden. Zu den angezeigten Statistiken gehören für den *Time*-Modus die Bestzeiten nach den Schwierigkeitsgraden sortiert. Desweiteren wird die Trefferquote wie auch die Gradzahl der durchschnittlichen Abweichung der Fehlversuche angezeigt. Beim *Wave*-Modus werden dagegen die Highscores für die jeweiligen Modi dargestellt. Dieser steht für die maximale Rundenanzahl oder *Waves*, welche der Spieler überstanden hat.

4.5.4 Help

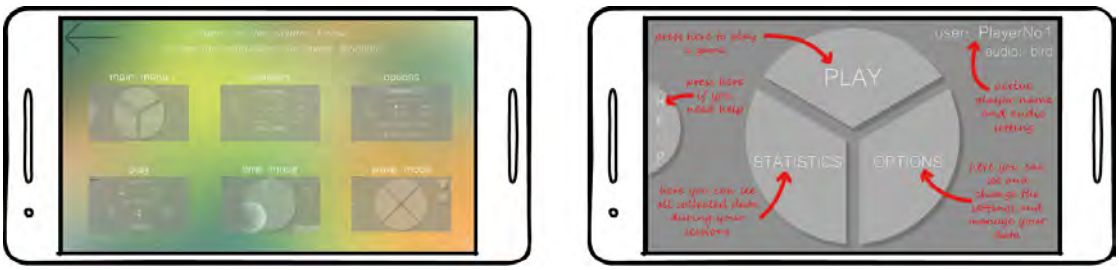

Abbildung 4.8: Hilfemenü mit Übersicht und ein einzelner Hilfe-Bilschirm

Ebenfalls direkt von `Menu` aus ist die Spielhilfe zugreifbar. Dadurch können vorallem neue Nutzer die wichtigsten Funktionen und Abläufe kennenlernen ohne weiter in der Anwendung danach suchen zu müssen. Wie im linken Bild in Abbildung 4.8 zu sehen bekommt der Nutzer zuerst einen Überblick über alle Bildschirme, die das Programm beinhaltet. Hier wird dem Spieler die Möglichkeit geboten mit einem Drücken auf das entsprechende Bild, dieses Fenster zu vergrößern und die Funktionen erklären zu lassen (Abbildung4.8, rechts). Zu `Statistics` werden die einzelnen angezeigten Informationen erklärt und wie diese zu Stande kommen. Bei `Options` wird erläutert wie man seinen Benutzernamen verändert, Daten exportiert und vom Gerät löscht. Ebenso werden die verschiedenen Soundschemata kurz kommentiert. Der Punkt `Play` sowie die

Modi werden am ausführlichsten beschrieben. Hier werden dem Nutzer die einzelnen Modi mit den jeweiligen Schwierigkeitsgraden und deren Unterschiede näher gebracht. Danach wird noch auf die Abläufe der verschiedenen Spielmodi eingegangen, sowie deren Spielkonzept und Siegbedingungen.

4.5.5 Play

In der Szene `Play` werden die Voreinstellungen getätigt, welche die Rahmenbedingungen des Spiels definieren. Die Auswahlmöglichkeiten wurden mit Hilfe von *ToggleGroups* umgesetzt, diese bilden eine Single-Choice-Auswahl von verschiedenen Optionen. In der ersten Gruppe wählt der Nutzer den Modus in dem er spielen will, entweder *Time Mode* oder *Wave Mode*. Die zweite Auswahl verlangt die Wahl zwischen den Schwierigkeitsgraden *Easy*, *Medium* oder *Hard*. Je nach getroffener Auswahl wird nach Drücken des Buttons *Start* entweder die Szene `TimeMode` oder `WaveMode` mit der aktivierten Schwierigkeit gestartet.

4.5.6 Time Mode

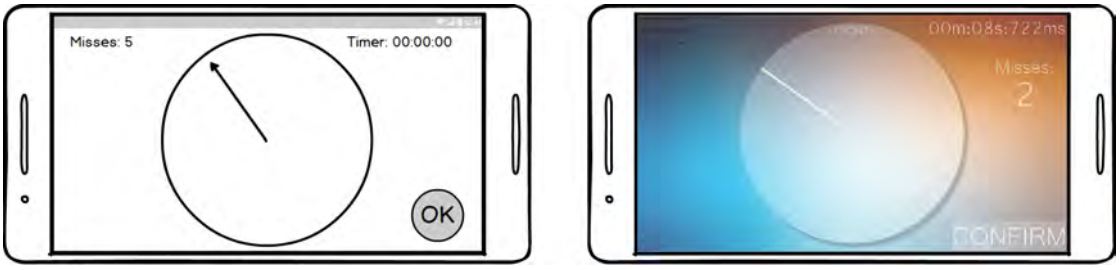

Abbildung 4.9: Mockup vs Screenshot: Time Mode

Bei ausgewähltem *Time Mode* wird diese Szene geladen. Über der Spieloberfläche wird beim Start ein großer „Start“-Button angezeigt. Solange dieser noch nicht berührt wurde, sind keine anderen Eingaben möglich und die Zeit der Szene steht still, ist also pausiert. Beim Drücken des Buttons startet das Spiel.

Der Spielercharakter befindet sich nun in der Mitte eines virtuellen Raumes. Da wir nicht die Sicht des Charakters, sondern nur sein Gehör brauchen, ist nur ein *AudioListener* mit diesem Prefab verbunden. Anschließend findet auf einer Position auf einem Kreis um den Spieler die Instanziierung des Gegners statt. Dies wird durch einen zufallsgenerierten Winkel umgesetzt, welcher über Winkelmaß zu einer einzelnen Position umgerechnet wird (siehe Listing A.4). Da der Zeitmodus gespielt wird, bleibt das Gegner-Prefab stationär an seiner Entstehungsposition und fängt nun an die angeheftete Audiodatei abzuspielen. Oben rechts im Bild befindet sich der *Timer*, dieser arbeitet unter Anderem mit der Unityklasse *Time* und *Timespan*, um die gebrauchte Zeit hochzuzählen und anzuzeigen. Den Großteil des Bildes nimmt der „Kompass “ ein, welcher dem Nutzer durch seine Beschriftung bei der Orientierung hilft. Auf diesem befindet sich der Richtungsanzeiger, welcher durch Berühren des Bildschirms um den Mittelpunkt gedreht werden kann. Dieser muss nun vom Spieler auf das Geräusch ausgerichtet und mit dem Button *confirm* rechts unten bestätigt werden. Nach Bestätigung wird geprüft, ob eine *OnTriggerStay*-Methode aktiv ist, welche dem System bestätigt, ob sich der Zeiger innerhalb des Toleranzbereichs befindet. Diese Methode bleibt aktiviert, wenn sich der *Collider* des Gegners mit dem des Zeigers überlappen (Collider sind dafür da Berührungen von verschiedenen Spielobjekten zu überwachen). Die Toleranzbereiche wurden umgesetzt, indem die Gegner bei einfacheren Schwierigkeitsgraden hochskalierte und bei schwereren runterskalierte *Collider* besitzen. Sollte die Überprüfung ein negatives Ergebnis zurückgeben, wird ein Fehlversuch vom System vermerkt und die Abweichung in Grad vom Zeiger zum Zielpunkt gespeichert. Die Fehlversuche werden rechts im Bild unter *Misses* angezeigt. Wenn ein positives Ergebnis zurückgegeben wird, dann stoppt die Zeit des *Timers* sowie das Spiel.

Anschließend wird ein *Panel* (quasi ein Pop-Up, Bild im Bild) über der Spieloberfläche instanziiert, welches das Ende der Spielrunde anzeigt. Ebenso wird hier die Zeit, welche benötigt wurde bis zum Fund des Ziels, wie auch die durchschnittliche Abweichung und die Fehlversuche während des Durchlaufs. Währenddessen werden die gesammelten Daten mit den vorhandenen in den *PlayerPreferences* verglichen und ersetzt oder verrechnet.

4.5.7 Wave Mode

Abbildung 4.10: Unterschiede zwischen den drei Schwierigkeitsmodi beim Wave Mode

Bei ausgewähltem *Wave Mode* wird diese Szene geladen. Über der Spieloberfläche wird beim Start, genauso wie beim Zeitmodus, ein großer „Start “ angezeigt. Das Spiel startet erst sobald der Nutzer diesen berührt.

Die Eigenschaften und Position des Spielercharakters werden hier nicht noch einmal beschrieben, diese unterscheiden sich nicht vom *Time Mode*, da hier auf das selbe Prefab zurückgegriffen wird. Der erste Unterschied findet sich aber schon beim Spawnen der Gegner. Hier wird genauso in einem zufälligen Winkel auf einem Kreis um den Mittelpunkt instanziiert, aber der Winkel wird je nach festgelegtem Schwierigkeitsgrad gerundet. Dieses Runden erfolgt bei *Easy* in 90 Grad Schritten, das bedeutet der Gegner entsteht entweder direkt vor, hinter, rechts oder links vom Spieler. Bei *Medium* wird in 45 Grad Schritten gerundet, so dass doppelt so viele Möglichkeiten entstehen: vorne, vorne-rechts, rechts, hinten-rechts, usw. Drei mal so viele Instanziierungspunkte gibt es dann bei *Hard*, hier werden durch das Runden auf 33,3 Grad-Schritte 8 Positionen möglich. Ebenso je nach Schwierigkeit werden Buttons, wie in Abbildung 4.10 zu sehen, an den möglichen Spawnpunkten instanziiert, also 4 bei *Easy*, 8 bei *Medium* und 12 bei *Hard*. Die Gegner starten direkt nach ihrer Entstehung mit der Wiedergabe der Audiodatei und bewegen sich geradlinig mit konstanter Geschwindigkeit auf den Spielercharakter zu. Dazu muss sich der Gegner zuerst auf die Position des Spielers ausrichten, wie in Listing A.5 unter der Methode *Start* zu sehen und danach wird per *Update*-Methode die Bewegung umgesetzt. Wenn der Collider des Gegners den Collider des Spielers berührt wird ein OnTriggerEnter-Event gestartet. Dieses Event startet eine Methode innerhalb der *Player*-Klasse, welche ein Integer „lives “ um 1 dekrementiert. Visualisiert werden die aktuellen Lebenspunkte rechts oben im Bild mit einem Herz

und einer Zahl darin. Daraufhin wird in der Gegnerklasse die *TargetFound*-Methode aus Listing A.5 gestartet, der Gegner gelöscht und ein neuer wird per Prefab und eben beschriebener Instanziierungsmethode erschaffen. Um den Gegner davon abzuhalten ein Leben abzuziehen, muss der Button gedrückt werden, welcher auf der Position des Gegners instanziiert wurde. In dem Falle wird der Gegner gelöscht, ein neuer instanziiert und dem Spieler ein Punkt gutgeschrieben. Die Punkte werden unterhalb der Lebens mit einem Stern und einer Zahl darin angezeigt. Sollte ein Button gedrückt werden, bei dem sich kein Gegner befindet, wird wieder ein Leben abgezogen. Sobald der Integer für die Lebenspunkte auf 0 fällt wird, ähnlich wie beim *Time Mode*, das Spiel angehalten und ein *Panel* instanziiert. Dieses PopUp-Fenster signalisiert dem Nutzer, dass das Spiel nun vorbei ist und zeigt ihm seinen erreichten Punktestand, sowie den Highscore für dieses Schwierigkeitsstufe an, welcher aus den *PlayerPreferences* abgerufen wird. Wenn der aktuelle Punktestand höher als der Highscore ist, wird dieser im Speicher überschrieben.

4.6 Android, iOS und Windows

Eine der großen Stärken von der Unity-Engine ist, dass man sein Programm auf den verschiedensten Plattformen lauffähig machen kann. Ich habe mir daher als Zielplattformen die gängigsten Betriebssysteme für Mobilgeräte des aktuellen Marktes ausgesucht. Um die Nutzbarkeit für diese Systeme zu erreichen mussten verschiedene Dinge angepasst werden. Über die „Build Settings “ in den Dateioptionen muss zuerst die Zielplattform verändert werden, nämlich Android, iOS oder Universal-Windows-Plattform. Im Falle von iOS wird dadurch ein Xcode-Projekt generiert, welches danach noch auf einem Mac ausgeführt und gebuilded werden muss. Dies ist notwendig, da man auf Windows keine iOS-Bundles generieren kann. Das Fehlen eines Mac-Systems ist auch der Grund warum am Ende keine lauffähige Variante des Programms für Apple-Geräte entstanden ist. Im Code selbst ist ebenfalls noch eine Veränderung nötig gewesen, diese war die Ordnerstruktur für den Export nochmals an das jeweilige Betriebssystem anzupassen.

4.7 Systemanforderungen

Das Projekt wurde so einfach wie möglich, ohne unnötige Komplexität umgesetzt. Ebenso sind die Geräte kaum eingeschränkt, da keinerlei weitere Sensoren wie Gyroskop oder Beschleunigungssensor benötigt werden. Dadurch ist die Anwendung auf allen Geräten lauffähig, welche die Mindestanforderungen für Unityprogramme erfüllen. Diese sind wie folgt:

- Android: OS 4.1 oder höher; ARMv7 CPU mit NEON-Support oder Atom CPU; OpenGL ES 2.0 oder höher
- iOS-Player erfordert iOS 9.0 oder höher
- Universal-Windows-Plattform: Windows 10 und eine Grafikkarte mit DX10 (Shader-Modell 4.0)-Fähigkeiten

5

Anforderungsabgleich

In Kapitel 3 wurden die Anforderungen an die Anwendung mitsamt ihrer Wichtigkeit ausgearbeitet. Nun wird im folgenden Kapitel die letztendliche Umsetzung bewertet. Diese Bewertung wird wieder mit Hilfe von Sternen (★) dargestellt, wobei ein Stern für das Schlechteste und fünf Sterne für das Beste stehen. Die Kriterien hierfür sind die Art der Umsetzung sowie die Qualität der Arbeit.

5.1 Funktionale Anforderungen

In dieser Sektion wird die Umsetzung der einzelnen funktionalen Anforderungen bewertet. Zur Übersicht über die Bewertungen der jeweiligen Punkte dient Tabelle 5.1. In den folgenden Abschnitten werden die vergebenen Sterne nochmal näher erläutert und begründet.

Funktionale Anforderung	Bewertung
Benutzer	★★★
Statistiken	★★★★
Anpassungsmöglichkeiten	★★★★
Spielmodi	★★★★★
Schwierigkeitsgrade	★★★★★
Datenexport	★★★★
Hilfe	★★★★
Richtungshören	★★★★

Tabelle 5.1: Bewertung der Umsetzung: Funktionale Anforderungen

Benutzer

Über die *Options*-Szene kann der Nutzer seinen Namen eintragen sowie wieder ändern. Dieser wird dann zur permanenten Datenhaltung unter den *PlayerPreferences* abgespeichert. Beim Datenexport wird der hinterlegte Benutzername zum identifizieren der Daten verwendet.

Diese Anforderung wurde als unwichtig eingestuft. Darum wurde die Anwendung so eingerichtet, dass alle Funktionen auch ohne Eingabe eines Namens arbeiten.

Bewertung : ★★★

Statistiken

Mit der *Statistics*-Szene wurde dieser Aspekt der Anforderungsanalyse umgesetzt. Hier werden sämtliche Daten aus den *PlayerPreferences* ausgelesen. Diese können dann nach Modus und Schwierigkeit sortiert eingesehen werden. Folgende Informationen werden hier für jeden Schwierigkeitsmodus einzeln angezeigt:

- Rekordzeiten beim Time Mode
- Trefferquoten beim Time Mode
- Durchnittliche Abweichung vom Ziel in Grad
- Highscores beim Wave Mode

Die angezeigten Daten wurden etwas weniger umfangreich als möglich gewählt um dem Nutzer eine bessere Übersicht zu schaffen. Eine ausführlichere Anzeige bekommt der Nutzer, indem er die Exportieren-Funktion verwendet.

Die gesammelten Daten können jederzeit über das Optionsmenü vom Gerät gelöscht werden.

Bewertung : ★★★★

Anpassungsmöglichkeiten

Einstellungsmöglichkeiten hat der Nutzer sowohl über die Szene *Optionen*, wie auch über *Play*. Anpassen kann er:

- Geräuschschema
- Spielmodus
- Schwierigkeitsgrad
- Benutzername
- Lautstärke

Das Geräusch-Thema kann der Nutzer permanent ändern, dies wird auch für die nächsten Sitzungen in den *PlayerPreferences* hinterlegt. Der gewünschte Spielmodus sowie der Schwierigkeitsgrad werden vor jedem Start einer Runde abgefragt.

Bewertung : ★★★★

Spielmodi

Des Spiel bietet dem Nutzer zwei verschiedene Modi an. Wie in der Anforderungen festgelegt gibt es zum Einen, den eher arcadelastigen Modus, bei dem sich der Spieler von herannahenden Geräuschquellen verteidigen muss. Zum Anderen gibt es den Zeit-Modus bei dem man das Ziel in möglichst kürzester Zeit mit den geringsten Fehlversuchen und Abweichungen finden muss. Bei beiden Modi lassen sich Rekorde aufstellen und brechen.

Bewertung : ★★★★★

Schwierigkeitsgrade

Für jeden der beiden Modi wurden drei verschiedene Schwierigkeitsgrade umgesetzt. Das Spiel wird mit den in den Anforderungen festgelegten Kriterien einfacher beziehungsweise schwerer gemacht.

Bewertung : ★★★★★

Datenexport

Der Datenexport kann wie geplant eingesetzt werden. Sämtliche in den *PlayerPreferences* gesammelten Daten werden durch Drücken des Buttons „Export Data “ im Optionsmenü exportiert. Diese befinden sich danach innerhalb der Ordnerstruktur der Anwendung auf dem eigenen Gerät. Da ein CSV Format verwendet wird und die Daten entsprechend formatiert wurden, können diese als Tabellen angezeigt werden.

Bewertung : ★★★★

Hilfe

Direkt vom Hauptmenü aus ist der Button „Help “ verfügbar. Hier sind Bilder mit Erklärungen zu sämtlichen Funktionen und Einstellungsmöglichkeiten des Programms zu betrachten. Ebenso werden hier die Abläufe der jeweiligen Spielmodi Schritt für Schritt erklärt.

Bewertung : ★★★★

Richtungshören

In den Szenen *TimeMode* und *WaveMode* werden die jeweiligen zuvor eingestellten Anpassungen verwendet. Nach Initialisierung des Spiels durch Berühren des Start-Buttons hört der Nutzer ein Geräusch, welches von einem Gegner-Prefab mit angehängter Audiodatei abgespielt wird. Dieses Geräusch befindet sich stets an einer zufällig generierten Stelle auf einem Kreis um den Spieler. Beim Zeitmodus bestimmt dieser die Richtung mithilfe eines Zeigers und bestätigt danach. Die Abweichung vom Ziel wird in Grad gespeichert. Wenn der Zeiger sich innerhalb des für den Schwierigkeitsgrad entsprechenden Toleranzbereichs befindet, dann endet die Runde. Beim Wellenmodus bestimmt der Spieler die Richtung, indem er auf die entsprechenden Buttons für

mögliche Zielrichtungen tippt. Bei einem falschen Button wird ein Leben abgezogen, bei einem Richtigen hingegen ein Punkt gutgeschrieben. Die Runde endet, wenn der Spieler keine Lebenspunkte mehr hat. Nach einer Spielrunde, unabhängig vom Modus, wird die Statistik für das gerade beendete Spiel, sowie der Highscore angezeigt.

Bewertung : ★★★★

5.2 Nicht-Funktionale Anforderungen

Dieser Abschnitt bewertet die Umsetzung der nicht-funktionalen Anforderungen. Die Übersicht der Ergebnisse sind in Tabelle 5.2 abzulesen. Auch hier gilt eine Sterne-Skala mit einem Stern Minimum und fünf Sterne Maximum. Begründungen zu den Bewertungen folgen in den nächsten Abschnitten.

Unterhaltungswert

Die Anwendung wurde wie geplant als Spiel umgesetzt. Abwechslung wird durch die verschiedenen Modi, Schwierigkeitsgrade und Geräusch-Themas geschaffen. Die Motivation wird aufrechterhalten, indem man durch den Trainingseffekt immer wieder seine Highscores brechen kann.

Bewertung : ★★★★

Nicht-Funktionale Anforderung	Bewertung
Unterhaltungswert	★★★★
Mobilität	★★★★★
Zuverlässigkeit	★★★★★
Look & Feel	★★★
Leistung und Effizienz	★★★
Sicherheit	★★★
Offline Anwendung	★★

Tabelle 5.2: Bewertung der Umsetzung: Nicht-Funktionale Anforderungen

Mobilität

Die maximale Mobilität wird dadurch erreicht, dass die Anwendung auf keinerlei Sensoren angewiesen ist, wie zum Beispiel Gyroskop oder Beschleunigungssensor. Die Bestimmung der Position findet rein mit Eingaben auf dem Bildschirm statt. Somit kann die Applikation stehend, wie auch sitzend oder liegend benutzt werden. Ein weiterer Punkt für bessere Mobilität ist, dass keinerlei Internetanbindung notwendig ist.

Bewertung : ★★★★★

Zuverlässigkeit

Bei der Entwicklung wurde großer Wert auf das Abfangen von möglichen Fehlern gelegt. Bei allen Abfragen aus dem internen Speicher (*PlayerPreferences*), wie auch beim Szenenwechsel und anderen Funktionen, wurden Exceptions und Standardausgaben eingerichtet. Somit läuft das Programm trotz einer fehlerhaften Abfrage weiter ohne abzustürzen. Während der Testphase sind keinerlei Abstürze oder Ähnliches aufgetreten.

Bewertung : ★★★★★

Look & Feel

Im Design wurden verschiedene Aspekte berücksichtigt um es selbsterklärend sowie übersichtlich zu gestalten. Der Richtungsanzeiger im *TimeMode* wurde so gewählt damit er einem Kompass gleicht. Zusätzlich dazu wurden die Richtungen noch beschriftet. Somit erkennt der Spieler etwas aus dem echtem Leben wieder und weiß im besten Falle gleich um die Funktion und was er zu tun hat. Dieses radiale Design wurde auch in den *WaveMode* und das Hauptmenü übernommen. So entsteht eine gewisse Konsistenz und Wiedererkennungswert für den Nutzer. An geeigneten Stellen wurden noch für das optische Feedback Animationen eingefügt, somit wirkt das Look & Feel abgerundet und ansprechender.

Bewertung : ★★★★

Leistung und Effizienz

Durch die Minimierung der Datengröße und Speicherung im unity-eigenen *PlayerPreferences*-System bleiben alle Zugriffszeiten minimal. Da Unity eine 3D-Engine ist und daher die Belastung für das Mobilgerät höher ausfällt, wurde darauf geachtet diese zu minimieren. Dies wurde zum Einen durch die Objektplatzierung erreicht, indem die Objekte im Bild so einfach wie möglich gewählt wurden. Zum Anderen wurden die Kameraeinstellungen noch zusätzlich verändert, indem die Projektion orthografisch gewählt wurde. Somit hat man durch die sehr gute Optimierung der Unity-Engine nur annähernd die Last einer 2D Anwendung.

Somit wurde erreicht, dass die App, außer beim Start, keinerlei merkbare Ladezeiten aufweist.

Bewertung : ★★★★

Sicherheit

Außer den exportierten Daten sind keine weiteren von außen zugreifbar, da diese innerhalb der Unityanwendung gehalten werden.

Bewertung : ★★★★

Offline Anwendung

Weder zum Start noch für weitere Funktionen des Programms ist eine Internetanbindung notwendig.

Bewertung : ★★★★★

6

Zusammenfassung & Ausblick

In diesem Kapitel wird das Projekt nochmal im Überblick zusammengefasst. Danach werden die Schwierigkeiten und Erfahrungen erläutert, welche bei der Entwicklung auftraten. Danach wird noch ein kurzer Ausblick gewährt, welcher Verbesserungsmöglichkeiten sowie Erweiterungen des Programms genauer beschreibt.

6.1 Zusammenfassung

Im Laufe dieses Projekts ist ein Spiel für mobile Endgeräte entstanden, welches darauf abzielt die Lokalisierungsfähigkeit des Gehörs zu trainieren. Die Zielgruppe sind hierbei Menschen, die ihr Gehör verbessern wollen, aber auch Hörgeschädigte oder Tinnitusbetroffene. Das Training kann in zwei unterschiedlichen Modi, in je drei Schwierigkeitsstufen mit verschiedenen Geräuschschemata erfolgen. Beim Zeitmodus muss der Nutzer, so schnell und fehlerlos es ihm möglich ist, die Richtung einer Schallquelle lokalisieren indem er einen Zeiger auf dem Bildschirm darauf ausrichtet. Beim Wellenmodus hingegen muss der Spieler auf ihn zukommende Geräuschquellen durch Betätigen des, für die Richtung entsprechenden, Buttons finden bevor sie ihn erreichen.

Die Anwendung sammelt während dem Spielen erhobene Daten wie Statistiken und Highscores um sie permanent zu speichern. Diese Daten können als CSV-Datei im Tabellenformat exportiert werden. Dies wurde mit dem Hintergedanken umgesetzt, dass die Daten für spätere Auswertungen benutzt werden können.

6.2 Kritikpunkte

Die Arbeit mit Unity3D war im Großen und Ganzen sehr angenehm. Erleichtert wurde die Arbeit durch eine sehr gute Dokumentation und viel hilfreichen Content wie Tutorials und Ähnliches direkt von deren Website.

Leider war Unity genauso Fluch wie Segen. Während des Entwicklungsverlaufs wurden mehrere Updates veröffentlicht, welche grundlegendere Dinge in der Funktionaliät selbst sowie Coding veränderten. Dadurch sind manche Methoden, welche schon im Programm verwendet wurden, obsolet geworden und mussten ersetzt werden. Ein weiteres Problem war, dass sich nach manchen Updates die UI-Funktionalität verändert hat und damit die komplette Benutzeroberfläche des Programms unbrauchbar wurde (Abb. 6.1)

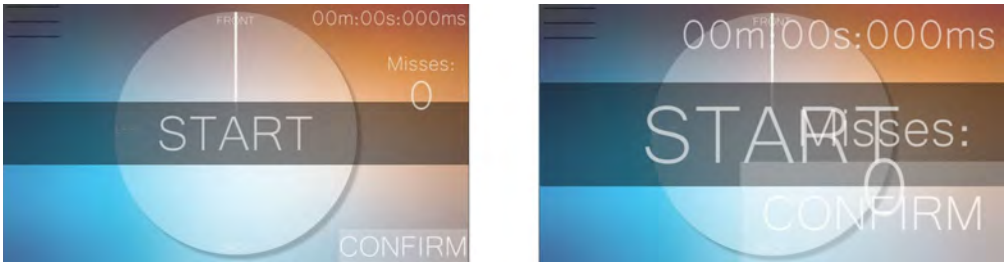

Abbildung 6.1: Vorher-Nachher: Unity Update

Ein schwerwiegenderes Problem war, dass der standardmäßige generierte 3D-Sound von Unity keinen sehr differenzierten Klang bietet. Dadurch wird es schwieriger die Soundquelle vor dem Spieler von derjenigen zu unterscheiden, welche hinter ihm ist.

Ebenso schwierig gestaltete es sich das bauen des Projektes für die verschiedenen Betriebssysteme. Unity erlaubt es zwar für alle Systeme zu schreiben und umzuwandeln, eine letztendlich installierbare Version kann aber beispielsweise für iOS nicht exportiert werden. Um dies zu erreichen muss das Projekt zuerst in XCode umgewandelt und dann auf einem Mac gebuildet werden. Da bei der Entwicklung nur ein Windows-PC verfügbar war konnte dieser Schritt also nicht durchgeführt werden. Das Programm ist also aktuell nur auf Android- und Windowsgeräten installierbar.

6.3 Ausblick

Mit dem aktuellen Umfang des Programms wurde ein solider Grundstein gesetzt um die Lokalisierungsfähigkeit spielerisch zu verbessern. Durch einige Zusätze könnte aber der Anwendungsbereich sowie der Langzeit-Spaßfaktor erweitert werden.

Durch das zusätzliche Abspielen von Störgeräuschen während des Lokalisierens könnte die auditive Stimulation noch intensiver gestaltet werden. Dadurch wäre die Anwendung beispielsweise effektiver für die Tinnitus-Retraining-Therapie.

Eine sinnvolle Erweiterung wäre auch ein Story-Modus. Dieser wäre, wie ursprünglich angedacht, an das Spiel „A Blind Legend “ angelehnt (Leider hätte die Umsetzung den Rahmen einer Bachelor-Arbeit gesprengt). Hierbei spielt der Nutzer durch eine Geschichte mit verschiedenen Begegnungen und Aufgaben. Diese sollen ihm nur durch Audio nähergebracht werden. Um die verschiedenen Aufgaben zu bestehen muss der Spieler durch das Identifizieren und Lokalisieren verschiedener Geräuschquellen mit der Welt interagieren. Der *WaveMode* wäre hierfür eine optimale Grundlage als Aufbau dieses Abenteuermodus. Um den Wiederspielwert zu erhöhen könnten die verschiedenen Begegnungen und Aufgaben zufällig erfolgen, indem sie aus einem größeren Pool von Möglichkeiten angeordnet werden. Durch mehrere verschiedene Modi könnte das Spiel mehr Nutzer ansprechen und die Vorhandenen länger am Training halten.

Verbesserungspotential hat im aktuellen Status auch die 3D-Audiowiedergabe. Wie schon in Abschnitt 6.2 beschrieben funktioniert diese zwar, aber die Richtungen lassen sich nicht immer eindeutig differenzieren.

Ein Punkt welcher die Mobilität der Anwendung nochmal verbessern würde wäre die Möglichkeit noch zusätzlich eine Einstellung zur Bildschirmorientierung einzubauen. Im Moment ist nur die Anzeige im Querformat möglich. Eine Hochformat-Einstellung könnte in Situationen von Vorteil sein, in denen der Nutzer nur eine Hand frei hat um das Gerät zu bedienen.

Literaturverzeichnis

[1] Schickler, M., Pryss, R., Reichert, M., Schobel, J., Langguth, B., Schlee, W.: Using Mobile Serious Games in the Context of Chronic Disorders - A Mobile Game Concept for the Treatment of Tinnitus. In: 29th IEEE Int'l Symposium on Computer-Based Medical Systems (CBMS 2016). (2016) 343–348

[2] Dowino: A Blind Legend. http://www.ablindlegend.com (Zugriff am 12.02.2019)

[3] Blauert, J.: Räumliches Hören. Hirzel Stuttgart (1974)

[4] Association, A.S.L.H., et al.: (central) auditory processing disorders. (2005)

[5] Ptok, M., Berger, R., von Deuster, C., Gross, M., Lamprecht-Dinnesen, A., Nickisch, A., Radü, H., Uttenweiler, V.: Auditive Verarbeitungs-und Wahrnehmungsstörungen Konsensus-Statement der Deutschen Gesellschaft für Phoniatrie und Pädaudiologie. HNO **48** (2000) 357–360

[6] Einfacher Hoeren: Hörwiki - Die Auditive Verarbeitungs- und Wahrnehmungsstörung(AVWS). https://www.einfacher-hoeren.de/de/avws-auditive-wahrnehmungsstoerung (Zugriff am 25.04.2019)

[7] Rehabilitationsklinik Werscherberg: Rehabilitation bei Auditiven Verarbeitungs- und Wahrnehmungsstörungen. https://www.rehaklinik-werscherberg.de/avws.html (Zugriff am 25.04.2019)

[8] Berssenbrügge, H.: Diagnostik auditiver Verarbeitungs-und Wahrnehmungsstörungen: anamnestische Befunde und ihr Zusammenhang mit auditiven Verarbeitungs- und Wahrnehmungsstörungen. Inauguraldissertation. Universitätsklinikum Münster (2004)

[9] Hören.at: Binaurales Hören: Wie wir Geräusche orten. http://www.hoeren.at/binaurales-hoeren/ (Zugriff am 25.04.2019)

[10] Dr. Stefanie Sperlich: Altersschwerhörigkeit. https://www.netdoktor.at/krankheit/altersschwerhoerigkeit-7876 (Zugriff am 25.04.2019)

[11] Tinitracks: Tinnitus-Symptome. http://www.tinnitracks.com/de/tinnitus/symptome (Zugriff am 25.04.2019)

[12] Schickler, M., Pryss, R., Reichert, M., Heinzelmann, M., Schobel, J., Langguth, B., Probst, T., Schlee, W.: Using Wearables in the Context of Chronic Disorders - Results of a Pre-Study. In: 29th IEEE Int'l Symposium on Computer-Based Medical Systems. (2016) 68–69

[13] Forum HNO: Auditive Verarbeitungs- und Wahrnehmungsstörung (AVWS). http://www.forumhno.de/ihre-beschwerden/auditive-wahrnehmungsstoerung-avws.html (Zugriff am 25.04.2019)

[14] Schickler, M., Reichert, M., Pryss, R., Schobel, J., Schlee, W., Langguth, B.: Entwicklung mobiler Apps: Konzepte, Anwendungsbausteine und Werkzeuge im Business und E-Health. eXamen.press. Springer Vieweg (2015)

[15] Therapie am Main: Auditive Wahrnehmungsstörungen. http://www.therapie-am-main.de/4c33919768147100b/4c33919768147f525/index.html (Zugriff am 25.04.2019)

[16] Wikipedia: Spiel-Engines. https://de.wikipedia.org/wiki/Spiel-Engine (Zugriff am 17.03.2019)

[17] Unity: Learn More. https://unity3d.com (Zugriff am 17.03.2019)

[18] Unity: Documentation, Scenes. https://docs.unity3d.com/Manual/CreatingScenes.html (Zugriff am 17.03.2019)

[19] Unity: Documentation, Prefabs. https://docs.unity3d.com/Manual/Prefabs.html (Zugriff am 17.03.2019)

A

Quelltexte

In diesem Anhang sind einige wichtige Quelltexte aufgeführt.

```
1  public void LoadLevel(string name)
2  {
3          menuClickSound.Play();
4          SceneManager.LoadScene(name);
5  }
```

Listing A.1: LoadLevel Methode

```
1  string[][] output = new string[rowData.Count][];
2
3  for(int i = 0; i < output.Length; i++)
4  {
5  output[i] = rowData[i];
6  }
7
8  int length = output.GetLength(0);
9  string delimiter = ";";
10
11 StringBuilder sb = new StringBuilder();
12
13 for (int i = 0; i < length; i++)
```

```
14 {
15 sb.AppendLine(string.Join(delimiter, output[i]));
16 }
17
18 string filePath = getPath();
19
20 StreamWriter outStream = System.IO.File.CreateText(filePath);
21 outStream.WriteLine(sb);
22 outStream.Close();
```

Listing A.2: Ausschnitt aus der DataExport Klasse, nachdem die zu exporierenden Daten bereits im Array rowData gespeichert wurden

```
1  void Start(){
2          float time = PlayerPrefs.GetFloat("Time", 99999999999f);
3          System.TimeSpan t = System.TimeSpan.FromSeconds(time);
4          string timerFormatted =
5           string.Format("{0:D2}m:{1:D2}s:{2:D3}ms",
6             t.Minutes, t.Seconds, t.Milliseconds);
7          timeText.text = timerFormatted;
8
9          float misses = PlayerPrefs.GetInt("misses", 0);
10         float hits = PlayerPrefs.GetInt("hits", 0);
11         float hitrate = 999;
12         if (hits + misses != 0)
13         {
14         hitrate = (hits / (misses + hits)) * 100f;
15         }
16         hitrateText.text = Mathf.RoundToInt(hitrate) + "%";
17
18         int overallDeviation = Mathf.RoundToInt
```

```
19          (PlayerPrefs.GetFloat("OverallDeviation", 180));
20          int overallTries = PlayerPrefs.GetInt("OverallTries",0);
21          deviationForTries.text = "Your deviation is " +
22           overallDeviation + " in " + overallTries + " tries.";
23
24
25          int score = PlayerPrefs.GetInt("bestScore", 0);
26          highscore.text = score.ToString();
27  }
```

Listing A.3: Statistics Methode

```
1  float ang = Random.value * 360;
2  Vector3 pos;
3  pos.x = center.x + radius * Mathf.Sin(ang * Mathf.Deg2Rad);
4  pos.y = center.y + radius * Mathf.Cos(ang * Mathf.Deg2Rad);
5  pos.z = 0f;
6  Instantiate(enemyPrefab, pos,
7          Quaternion.AngleAxis(90f, new Vector3(1f, 0f, 0f)));
```

Listing A.4: Positionsbestimmung des Gegners und Instanziierung

```
1  private void Start()
2  {
3  // Player lokalisieren und den Gegner darauf ausrichten
4  playerTransform = GameObject.FindWithTag("Player").transform;
5  transform.LookAt(playerTransform);
6  }
7
8  void Update()
9  {
```

```
10 // Nach vorne bewegen mit festgelegter Geschwindigkeit
11 if (isFound == false)
12 {
13 transform.position += transform.forward *
14         MoveSpeed * Time.deltaTime;
15 }
16 }
17
18 public void TargetFound()
19 {
20 Instantiate(text, this.transform.position, Quaternion.identity);
21 isFound = true;
22 enemyAttacking.Stop();
23 enemyIsHit.Play();
24 Destroy(gameObject);
25 }
```

Listing A.5: Funktionen innerhalt der Gegner-Klasse

Abbildungsverzeichnis

Tabellenverzeichnis

Name: Alexander Böck Matrikelnummer: 834584

Erklärung

Ich erkläre, dass ich die Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Ulm, den 03.06.2019 .....

Alexander Böck