Umsetzung und Vergleich verschiedener BCI gesteuerter Bildschirmtastaturen

Eingereicht von:	Marcel Glab
Matrikelnummer:	4878540
Am:	28.09.2022

1. Gutachter: Prof. Dr. Lutz Prechelt
   Freie Universität Berlin, Fachbereich Informatik
   Leiter der Forschungsgruppe Software Engineering
2. Gutachterin: Prof. Dr. Claudia Müller-Birn
   Freie Universität Berlin, Fachbereich Informatik
   Leiterin der Forschungsgruppe Human-Centered Computing

Selbstständigkeitserklärung

Ich versichere hiermit an Eides Statt, dass diese Arbeit von niemand anderem als meiner Person verfasst worden ist. Alle verwendeten Hilfsmittel wie Berichte, Bücher, Internetseiten oder ähnliches sind im Literaturverzeichnis angegeben, Zitate aus fremden Arbeiten sind als solche kenntlich gemacht. Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungskommission vorgelegt und auch nicht veröffentlicht.

Montag, 26. September 2022

X _____

Marcel Glab

Zusammenfassung:

Aufgrund ihrer körperlichen Behinderungen sind viele Menschen nicht in der Lage, über Text oder Sprache zu kommunizieren. Ziel dieser Arbeit ist es, Möglichkeiten zur Texteingabe zu untersuchen, die diese Nachteile ausgleichen oder wenigstens mindern können.

Dafür wurden drei Bildschirmtastaturen entwickelt und mittels einer kleinen Feldstudie auf Tauglichkeit hin überprüft. Die Bachelorarbeit entstand im Sommersemester 2022 während meiner Projektarbeit in Kooperation mit Jakob Schmock bei der munevo GmbH.

Während der Testphase konnte von den drei entwickelten Prototypen eine Tastatur besonders überzeugen.

Die Ergebnisse der Studie zeigen, dass eine Texteingabe mittels des Brain-Computer-Interfaces NextMind im jetzigen Entwicklungszustand des Systems grundsätzlich möglich ist, jedoch nur für Menschen sinnvoll erscheint, bei denen alle anderen Systeme versagen.

Summary:

Due to their physical disabilities, many people are unable to communicate either by text or speech. The goal of this work is to find possibilities for text input, which compensate or at least reduce this disadvantage.

For this purpose, three on-screen keyboards were developed and tested by means of a small field study. This bachelor thesis was written during summer semester 2022 based on my project work in cooperation with Jakob Schmock at munevo GmbH.

During the test phase of the three prototypes developed, one keyboard was particularly convincing

The results of the study show that text input using the brain-computer interface NextMind in its current state of development is possible in principle – still, the system appears to be useful only for people other systems fail to support accordingly.

Inhaltsverzeichnis

I. Einführung

Die Fähigkeit, Texte schreiben zu können, ist in der heutigen Gesellschaft unabdingbar. Egal ob im privaten oder beruflichen Umfeld: für Menschen mit Behinderungen kann diese Tätigkeit oft gar nicht mehr, und wenn doch, nur mit sehr viel Aufwand und Anstrengung ausgeführt werden. Häufig sind Menschen mit Behinderungen auch beim Schreiben von Texten auf fremde Hilfe angewiesen. Um diese Menschen in die Gesellschaft zu integrieren, ist es wesentlich, Möglichkeiten zur Texteingabe zu finden, die den Nachteil der Behinderung ausgleichen oder zumindest mindern.

In diesem Projekt wurden deshalb verschiedene Benutzeroberflächen entworfen, implementiert sowie mittels einer kleinen Feldstudie auf Realisierbarkeit, Nützlichkeit und Leistungspotential untersucht. Für die Umsetzung dieser Arbeit kam hauptsächlich das Brain-Computer-Interface *NextMind* zum Einsatz. Dieses Produkt basiert auf der Technologie der Elektroenzephalographie.

1. Allgemeines über Elektroenzephalographie

Die Elektroenzephalografie (EEG) ist eine Methode zur Messung der Gehirnaktivität. Die Messung erfolgt in der Regel mit kleinen, scheibenförmigen Metallobjekten, den Elektroden, die mit Kappen, Netzen oder Stirnbändern auf der Kopfhaut befestigt werden [1].

Die Neuronen des Gehirns kommunizieren untereinander über einen Ionenstrom, den die Elektroden messen und als Spannungsschwankungen aufzeichnen. Die elektrische Aktivität im Gehirn wird als Aktionspotenzial bezeichnet. Da ein Gehirn über sehr viele Neuronen verfügt, ist das EEG nur in der Lage, Messungen von kleinen Bereichen des Gehirns zu erfassen. Neuronen leiten mittels elektrischer Impulse Botschaften aneinander weiter, so dass Verhalten und Gedanken entstehen können [2].

Die Messung durch ein EEG erfolgt über einen bestimmten Zeitraum. Sie wird grafisch als Wellenlinien mit Spitzen und Tälern dargestellt. Auf der x-Achse sieht man die Zeit, während auf der y-Achse das Spannungspotential der empfangenen Hirnaktivität aufgezeichnet wird.

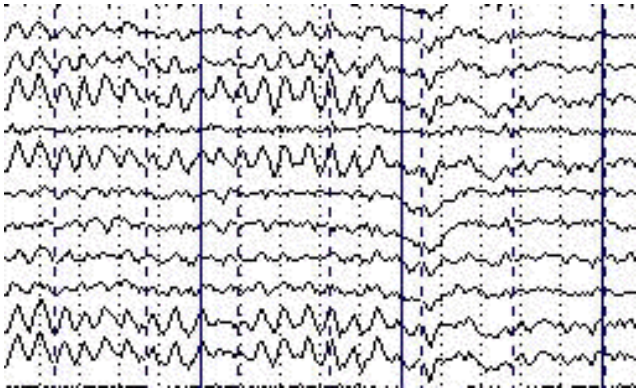

Abbildung 1: Die Ausgabe einer Elektroenzephalografie

In einer solchen Aufzeichnung sind in der Regel fünf Arten von Wellen zu finden: Alphawellen, Betawellen, Gammawellen, Thetawellen und Deltawellen [2].

Diese Wellen unterscheiden sich in zwei Parametern:

1. der Amplitude, die im Wesentlichen die Intensität der Hirnaktivität angibt, Einheit: Volt [V]
2. der Frequenz, die die Menge der Hirnaktivitäten angibt, Einheit: Hertz [Hz]

Das EEG wird hauptsächlich dazu verwendet, das Vorhandensein oder Fehlen spezifischer Hirnaktivitäten in bestimmten Bereichen des Gehirns festzustellen, so z.B. in der Medizin, um eine Vielzahl von Krankheiten und Hirnstörungen zu diagnostizieren: [3]

- Epilepsie
- Erkennung von Hirntumoren
- Schlafstörungen
- Schlaganfällen
- Demenz

Darüber hinaus wird das EEG zu Überwachungszwecken in der Medizin eingesetzt, z.B. bei Patienten im komatösen Zustand oder bei Operationen, um sicher zu sein, dass es dem Patienten gut geht. Da es sich bei der Elektroenzephalographie um ein nicht-invasives Verfahren handelt, das EEG also nicht in den menschlichen Körper implantiert wird und demzufolge jederzeit wieder entfernt werden kann, handelt es sich um eine schmerzfreie und sehr sichere Methode zur Messung der Gehirnaktivität.

Ein Nachteil dieser Methode besteht jedoch darin, dass die Bandbreite, beziehungsweise die Signalqualität von Übertragungen aufgrund von Entfernung der Elektroden zu den Neuronen, stark verringert ist [4].

2. Was ist *NextMind*?

NextMind ist ein im Jahr 2017 gegründetes Startup im Bereich Neurotechnologie aus Frankreich, das im Jahr 2020 ein gleichnamiges Developer-Kit veröffentlichte. Das Developer-Kit beinhaltet das sogenannte Brain-Computer-Interface (BCI), ein Stirnband, zum Befestigen des Gerätes am Kopf sowie ein Ladekabel. Besagtes BCI wird im Rahmen dieser Bachelorarbeit als Eingabesystem benutzt.

Das Brain-Computer-Interface ist ein nicht invasives EEG, das an der Rückseite des Kopfes mit Hilfe des Stirnbandes befestigt wird, um die Gehirnwellen des visuellen Kortex zu messen. Der visuelle Kortex ist ein Teil der Großhirnrinde im menschlichen Gehirn, der die visuelle Wahrnehmung ermöglicht.

Abbildung 2: Die Hardware des Projektes - das NextMind

Die Hardware des *NextMind* besteht aus einem kreisförmigen Körper, der die Hauptelektronik beinhaltet, sowie aus einer bogenförmigen Halterung, auf der neun Sensoren zur Messung der Gehirnwellen platziert sind (Abb. 2). Durch das Gewicht von lediglich 60 Gramm ist der Sensor als tragbares Gerät geeignet. Ebenfalls bietet das Gerät eine Akkulaufzeit von bis zu acht Stunden, bei einer Ladezeit von zwei Stunden. Durch das Aufstecksystem ist es ebenfalls

möglich, NextMind zusammen mit Brillen für VR (Virtual Reality) oder AR (Augmented Reality) wie zum Beispiel der *Oculus Quest 2* oder ähnlicher Hardware zu benutzen.

Das Ziel von *NextMind* ist es, Gehirnströme aus dem visuellen Kortex zu interpretieren und in digitale Befehle umzuwandeln, die es erlauben, Benutzeroberflächen in Echtzeit zu bedienen [5].

Dafür wird ein Software Developer-Kit (SDK) von NextMind angeboten. Dieses SDK findet Verwendung in der *Unity Engine*. *Unity* ist eine Cross Platform Game Engine, also eine Laufzeit- und Entwicklungsumgebung für Computerspiele. Die Laufzeitumgebung ist in C++ geschrieben und besitzt eine API (Application Programming Interface) für C#.

Durch den Einsatz der *Unity Engine* sowie die Verwendung von VR- und AR-Brillen ist das ursprüngliche Anwendungsszenario in der Spieleentwicklung gesetzt. Nichtsdestotrotz eignet sich die *Unity Engine* ebenfalls zur Erstellung anderer Software, die nicht ausschließlich für Unterhaltungszwecke gedacht ist.

Bevor Applikationen mit dem NextMind bedient werden können, muss das Gerät erst kalibriert werden. Die Kalibrierung dauert ungefähr 45 Sekunden und besteht darin, sich auf einen in der Mitte des Bildschirms platzierten *NeuroTag* wiederholt zu fokussieren. Nach der Kalibrierungsphase bewertet die Software auf einer Skala von 1 bis 5, wie gut das Gerät kalibriert wurde. Je höher das Ergebnis ist, desto schneller und präziser funktioniert das Betätigen der Benutzeroberflächen mit dem Gerät. Aufgrund der SDK ist es ebenfalls möglich, diesen Kalibrierungsprozess zu modifizieren oder ein komplett neues Konzept zu implementieren.

Die Bedienung der Benutzeroberfläche erfolgt über die wichtigste Softwarekomponente des Systems: *NeuroTag*. Es handelt sich hierbei um ein Grafik-Overlay, ein durch *NextMind* entwickeltes Blinkmuster, das den visuellen Bereich des Gehirns so stimuliert, sodass man mit Hilfe des EEG erkennen kann, was für ein Blinkmuster betrachtet wurde [6].

3. Limitationen der Arbeit

Da es sich bei der Funktionsweise des *NextMind* um einen nicht-invasiven Prozess handelt, muss in diesem Projekt mit Einschränkungen bei der Übertragung der Signale gerechnet werden. Diese werden z.B. bei der Auslösezeit der Schaltflächen spürbar. Durch die geringere Übertragungsrate ist die Auslösezeit verhältnismäßig hoch. Ein weiterer limitierender Faktor ist, dass maximal 10 Schaltflächen gleichzeitig beobachtet werden können. Zudem ist die Größe der Schaltflächen beschränkt, da es sich um ein visuelles System handelt.

Diese Einschränkungen könnten durchaus kompensiert werden - was jedoch den Rahmen dieser Arbeit sprengen würde. Optimiert wurde hier lediglich die Größe der Benutzeroberfläche durch verschiedene Umsetzungen. Die entwickelten Tastaturiterationen können zudem in Punkto Geschwindigkeit und Zuverlässigkeit nicht mit einer regulären Tastatur oder ähnlichen Eingabegeräten mithalten. Daher konzentriert sich diese Arbeit darauf, eine Eingabemöglichkeit speziell für Menschen zu schaffen, die konventionelle Eingabegeräte beispielsweise wegen einer muskulär-körperlichen Behinderung nicht bedienen können.

II. Stand der Wissenschaft und Technik

Laut Definition des *Sozialgesetzbuch V* bezeichnet man als medizinische Hilfsmittel Gegenstände, die erforderlich sind, um den Erfolg einer Krankheitsbehandlung zu sichern, einer drohenden Behinderung vorzubeugen oder eine Behinderung auszugleichen, soweit sie nicht als allgemeine Gebrauchsgegenstände des täglichen Lebens anzusehen sind (§ 33 Absatz 1 SGB V). Nachfolgend soll genauer auf den Status Quo in diesem Bereich eingegangen werden.

1. Körperliche Behinderungen

Unter körperlichen Behinderungen versteht man physiologische Einschränkungen des menschlichen Körpers, die durch verschiedenste Ursachen begründet sind – so zum Beispiel folgenschwere Krankheiten und/oder Verletzungen sowie angeborene chronische Erkrankungen.

Körperliche Behinderungen zeigen sich in unterschiedlichster Art und Weise:

Häufige Erscheinungsformen sind [7]:

- Querschnittslähmungen, bei denen die betroffene Person durch Lähmungen von Extremitäten beeinträchtigt sind
- zerebrale Bewegungsstörungen, wie Spastik, Arthrose oder Ataxie - hier haben Betroffene eine beeinträchtigte Kontrolle über gewisse Muskelgruppen und verfügen somit über eine deutlich eingeschränkte Feinmotorik
- neuromuskuläre Erkrankungen wie zum Beispiel Spinale Muskelatrophie oder Amyotrophe Lateralsklerose, aufgrund derer betroffene Menschen nach und nach wegen zunehmender Muskelschwäche ihre motorischen Fähigkeiten verlieren
- Amputationen

Das Ziel dieses Abschnittes ist es, einen aktuellen Überblick über Möglichkeiten zur Mensch-Computer-Interaktion (Human-Computer-Interaction, abgekürzt HCI) in Bezug auf medizinische Hilfsmittel zu bieten, die als Eingabegeräte dienen und/oder in Form von speziell konfigurierten Bedienoberflächen für Computersystemen zum Einsatz kommen.

2. Eingabegeräte

Die momentan am meisten genutzten Technologien in Bezug auf Eingabegeräte für Menschen mit Behinderung sollen nachfolgend vorgestellt und vor allem im Hinblick auf ihre Nutzerfreundlichkeit näher erörtert werden.

a) Sip-and-puff (SNP)

SNP ist eine Hilfsmitteltechnologie, die Steuersignale an Geräte senden kann, wenn die betroffene Person durch ein Röhrchen ein- und ausatmet. Da viele Behinderungen auch die Atemmuskulatur oder das Lungenvolumen einschränken, ist eine individuelle Anpassung dieser Geräte an die jeweilige Person notwendig. Zudem muss eine passende Kalibrierung erfolgen. Grundsätzlich können per SNP zwei verschiedene Signale gesendet werden: eines beim Einatmen und eines beim Ausatmen. Obendrein besteht die Möglichkeit, weitere Signale wie zum Beispiel besonders starkes Ein- und Ausatmen, zu erkennen und für die Steuerung anzuwenden [8].

b) Spracherkennung

Bei stark ausgeprägten körperlichen Behinderungen, die der betroffenen Person nichtdestotrotz das Sprechen erlauben, ist Spracherkennung eine passende Unterstützungstechnologie zur Texteingabe. Durch die rasante Entwicklung der Leistungsfähigkeit von Computern im Heimbereich sowie den wissenschaftlichen Fortschritt im Bereich der künstlichen Intelligenz erreichen aktuelle Systeme beim Diktieren von Fließtexten eine Erfolgsquote von annähernd 99% [9].

Diese Erfolgsrate wird allerdings nur erzielt, wenn die verwendende Person ein qualitativ gutes Mikrofon besitzt und die Technologie in einer ruhigen Umgebung Anwendung findet. Zudem sollte die Stimme der Person keine markanten Auffälligkeiten aufweisen, muss also einer gewissen Norm entsprechen, da viele Systeme auf bestimmte Stimmlagen trainiert sind [10]. Bei diversen körperlichen Einschränkungen ist das Sprechen zwar möglich und für einen Mitmenschen durchaus verständlich - dies lässt sich jedoch bisher nicht 1:1 auf Computersysteme mit Spracherkennung übertragen.

c) Augensteuerung

Augensteuerung ist eine Technologie, mit der man feststellen kann, wohin eine Person blickt. Augensteuerungsmodule bestehen gemeinhin aus Kameras, Lichtquellen und einer Recheneinheit. Das von dem Augensteuerungsmodul ausgesendete Licht wird von den Augen reflektiert. Dadurch kann die eingebaute Kamera die Pupillen erfassen und die Reflektionen auf der Iris oder auf dem Pupillenrand erkennen. Basierend auf dem Verhältnis zwischen Pupillenmitte und Reflektionspunkt, ist ein solches Steuerungsmodul in der Lage, zu berechnen, wo eine Person ungefähr hinschaut. Mit dieser Information können dann Benutzeroberflächen erstellt werden, die durch längeres Hinschauen ermöglichen, Schaltflächen zu aktivieren und/oder die Maus in Blickrichtung zu bewegen [11]. Diese Eingabemöglichkeit benötigt sehr viel Konzentration und ist dadurch auf Dauer sehr anstrengend und fordernd für die Person, die sie verwendet.

d) Brain-Computer-Interface (BCI)

Die bereits erwähnten BCI-Systeme sind Geräte, mit denen man Befehle an einem Computer ausführen kann, indem man Gehirnströme der betroffenen Person misst und analysiert. In den letzten Jahren wurde viel in diese Richtung geforscht, was zur Entwicklung höchst innovativer Projekte, wie beispielsweise einem gedankengesteuerten Rollstuhl, geführt hat [12]. Des Weiteren hat sich die Technik so weit verbessert, dass mittlerweile Geräte in der Verbraucherklasse existieren – so zum Beispiel das *NextMind*, das im Rahmen dieser Bachelorarbeit Verwendung findet.

III. Gewählter Lösungsansatz, Alternativen und Abwägungen

1. Problemstellungen

Während der Arbeit an diesem Projekt gab es an einigen Stellen Probleme. Aufgrund der nur minimal vorhandenen Dokumentation der *NextMind*-SDK, dauerte die Entwicklung der ersten Iteration deutlich länger als erwartet. Die Funktionsweise musste durch ausgiebiges Testen erst erschlossen werden. Aufgrund der geringen Nutzerzahlen konnte auch nicht auf vorherige Projekte oder Nutzererfahrungen zurückgegriffen werden. Eine Kommunikation mit dem Unternehmen *NextMind* selbst, um diese Probleme zu beheben war nicht möglich, da *NextMind* von *Snap* akquiriert wurde [13].

Nach der Übernahme von *NextMind* wurden die im Rahmen der Projektarbeit verwendeten Geräte nicht mehr zum Verkauf angeboten. Dadurch stellte sich die Frage, ob die Entwicklung der Tastaturen mit dem *NextMind* überhaupt noch sinnvoll ist. Es fiel die Entscheidung das Projekt dennoch weiterzuführen, da man Funktionsweise der Tastaturen (leider mit einigem Mehraufwand) auf andere BCI-Systeme übertragbar ist und *NextMind* immerhin bereits zur Verfügung stand. Daher erschien eine Umsetzung mittels dieses Geräts trotzdem zweckmäßig - zumindest für die erste Iteration.

2. Entwicklungsmodell

Die Software entstand im Rahmen eines agilen Softwareentwicklungsprozesses. Dementsprechend wurden zunächst nur die wesentlichen Anforderungen erhoben (siehe Anhang), um diese der Methodik agiler Softwareentwicklung folgend im Verlauf des Projekts zu ergänzen und weiter anzupassen. Hierfür wurde das Framework Scrum eingesetzt. Scrum ist ein agiles Vorgehensmodell, das sich deutlich von klassischen Projektmanagementmodellen wie beispielsweise dem Wasserfallmodell unterscheidet. Es gibt weder strikte Projektphasen noch eine Projektleitung vor, die Aufgaben verteilt. Stattdessen wird in sogenannten Sprints gearbeitet - bestimmte Zeitabschnitte, innerhalb derer zuvor definierte Ziele erreicht werden sollen [14]. In diesem Projekt lag die Dauer eines Sprints bei zwei Wochen.

3. Dokumentation

Für jedes der Teilprojekte wurde eine Inline-Dokumentation zur Erläuterung benutzter APIs, Funktionen und Methoden sowie aller relevanten globalen Variablen erstellt, um zukünftig an diesem Projekt arbeitenden Personen den Einstieg so einfach wie möglich zu gestalten. Darüber hinaus beinhaltete die Planungsphase jeder Iteration die Erstellung von Mockups, die grundlegende Elemente der Benutzeroberfläche aufzeigen. Des Weiteren wurde für Iteration I ein UML-Aktivitätsdiagramm erstellt, welches die Funktionsweise der Bildschirmtastatur zusätzlich erklärt.

IV. Funktionalität der Bildschirmtastaturen

1. Iteration I

Die Hauptlimitation von *NextMind* ist wie bereits erläutert die Beschränkung auf zehn gleichzeitig aktive *NeuroTags*. Aus diesem Grund schien ein unkonventionelles Tastaturlayout sinnvoll. Da das Ziel dieser Tastatur das Verfassen von Texten und nicht die Bedienung des Computers mittels Tastenkombinationen ist, konnte auf viele Elemente wie zum Beispiel Funktionstasten oder Pfeiltasten verzichtet werden. Um die Größe der Tastatur weiter zu minimieren, wurde die Anzahl an Tasten so weit wie möglich reduziert.

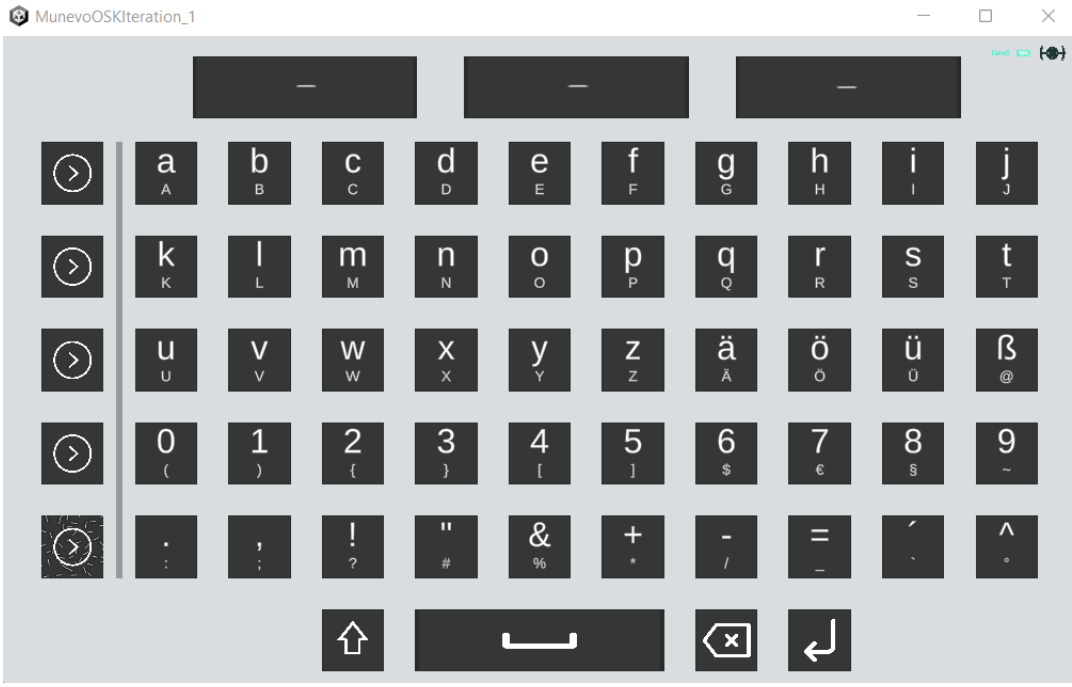

Abbildung 3: Screenshot - Iteration I

Das Layout der ersten Iteration der Bildschirmtastatur besteht aus 59 Tasten. Über diesen Tasten findet man drei Textfelder, in denen Wortvorschläge angezeigt werden. In dieser Version der Tastatur wird zwischen drei verschiedenen Schaltflächen unterschieden:

- Allgemeine Tasten zum Simulieren von Zeichen (Generic Buttons)
- Tasten zur Auswahl einer Reihe (Row Chooser)
- Spezielle Tasten, wie Umschalttaste, Leertaste, Rücktaste und Eingabetaste (Special Buttons)

Abbildung 4: Generic Button ohne und mit Fokus

Die für die Tasten verwendeten Sprites sind 2D-Grafikobjekte von *Unity*. Jeder *Generic Button* besteht aus einem dunkelgrauen, quadratischen Sprite und einem orangenen, quadratischen Sprite. Letzterer liegt hinter der Taste und vergrößert sich beim Fokussieren der jeweiligen Taste. Dies erzeugt einen visuellen Effekt, der dem Benutzer signalisiert, welche Schaltfläche momentan bedient wird.

Außerdem beinhaltet jeder *Generic Button* zwei Textkomponenten, die vertikal übereinander angeordnet sind. Die obere Textkomponente beinhaltet das Symbol, das beim Fokussieren der Schaltfläche simuliert wird (Primärwert), während die untere Komponente das Symbol anzeigt, das entsteht, wenn man zuvor die Umschalttaste gedrückt hat (Sekundärwert).

Es gibt fünf Reihen mit jeweils zehn Generic Buttons, sodass das Tippen von 100 verschiedenen Symbolen möglich ist. Die Schaltflächen in der äußeren linken Spalte dienen zur Auswahl der jeweiligen Reihe von Tasten (*RowChooser*). Zu Beginn sind die *NeuroTags* der Elemente in der linken Spalte sowie der Schaltfläche für das Betätigen der automatischen Vervollständigung aktiviert. Betätigt man einen der fünf Knöpfe in der linken Spalte, werden vorerst alle aktiven *NeuroTags* deaktiviert und dann die NeuroTags der Schaltflächen in der gewählten Reihe aktiviert.

Unter den oben genannten Schaltflächen findet man vier weitere Tasten: die Umschalt-, Leer-, Zurück- und die Eingabetaste (Special Buttons). Dabei handelt es sich nach eigener Erfahrung um die besonders oft benötigten Tasten. Daher wurden die

NeuroTags dieser Tasten so implementiert, dass sie zur selben Zeit wie die Row Chooser aktiv sind und somit direkt mit einem Tastendruck bedient werden können.

Um die Tastatureingaben zu simulieren, wurde die *SendInput* Funktion aus der *user32.dll*-Bibliothek benutzt. *User32.dll* ist eine Bibliothek, die die von Windows zur Verfügung gestellte Komponente *USER* implementiert, mit der Standardelemente der Benutzeroberfläche wie Fenster und Menüs erstellt und bearbeitet werden können. Programme wie zum Beispiel auch diese Bildschirmtastatur können Funktionen von *USER* aufrufen, um das Empfangen von Fenstermeldungen bei Tastatur- oder Mausereignissen zu realisieren.

Um Tastenanschläge simulieren zu können, benötigt *SendInput* drei Parameter:

- die Anzahl der Eingaben
- ein Array vom Typ *INPUT*, das die zu sendenden Eingaben beinhaltet
- die Größe des *INPUT* structs

Hierbei finden *virtual keycodes* (VK) Verwendung, die ebenfalls in der *user32.dll*-Bibliothek in Form von hexadezimalen Zahlen definiert sind. Es existieren Keycodes für jede Taste auf einer Tastatur, die man physisch bedienen kann, ohne Tastenkombinationen zu benutzen:

- a-z
- 0-9
- einige Satzzeichen
- Sonderzeichen
- Modifikationstasten – wie z.B. shift, alt und strg

Damit die Tastenanschläge auch im richtigen Programm ankommen, muss sichergestellt sein, dass das Zielprogramm beim Betätigen der Tasten im Fokus steht. Üblicherweise wird ein Fenster in den Fokus gesetzt, wenn die Person in die Fläche des Programms klickt.

Dieses Verhalten bringt zwei Probleme mit sich: zum einen verschwindet die Tastatur im Hintergrund, sobald in ein Textfeld eines anderen Fensters geklickt wird. Zum anderen verliert das Fenster mit dem Textfeld den Fokus, wenn eine Schaltfläche der Tastatur bedient wird,

wodurch das Betriebssystem nicht mehr sicherstellen kann, an welches Fenster der Tastenanschlag geschickt werden soll.

Um diese Probleme zu lösen, kam die *SetWindowLong* Funktion aus *user32.dll*-Bibliothek zum Einsatz. In dieser Funktion wurden zwei Flags als Parameter gesetzt: **GWL_STYLE = -20**, was die Tastatur immer an oberster Stelle erscheinen lässt, und **WS_DISABLED = 0x08000000L**, das den inhärenten Focus der Tastatur unterbindet.

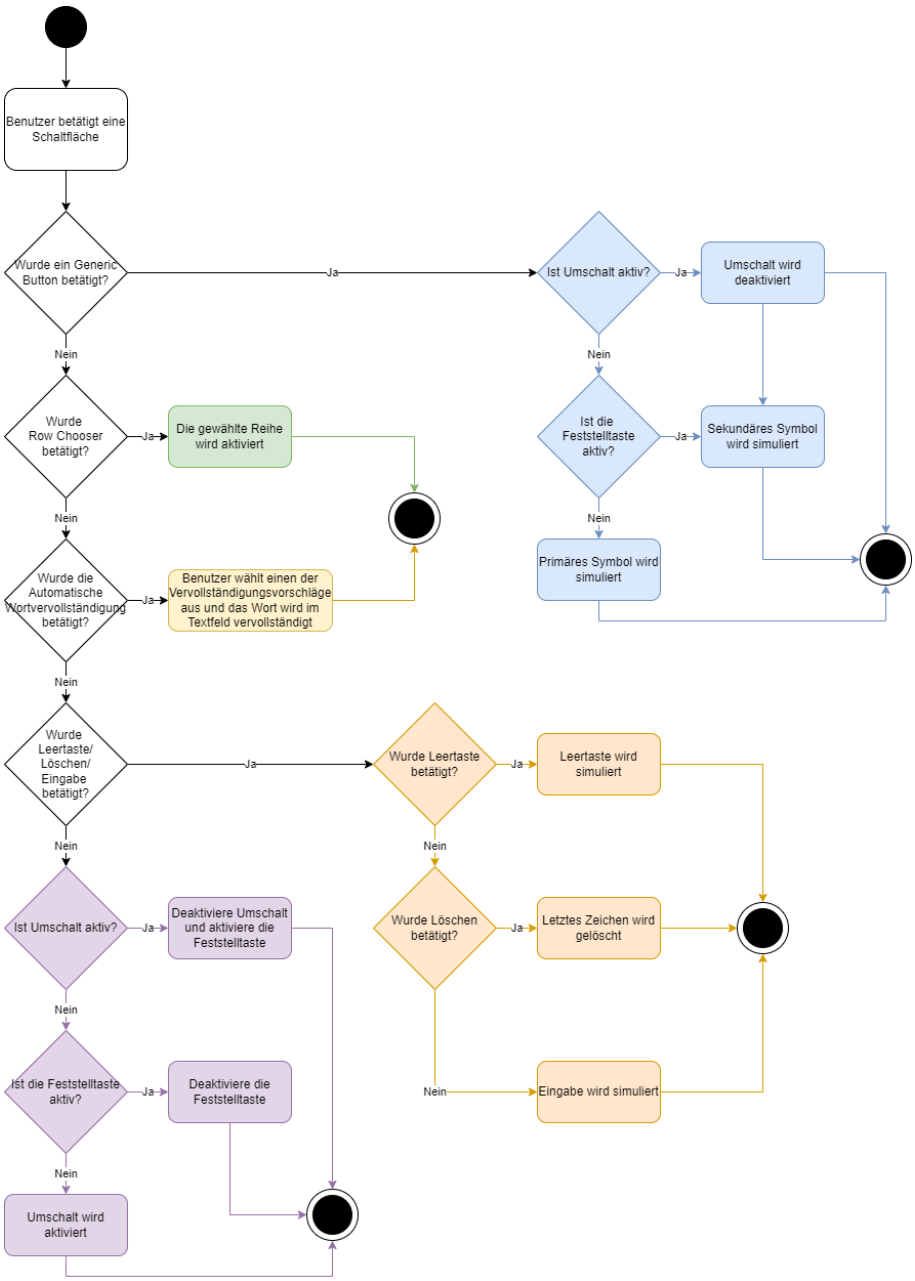

Abbildung 5: Funktionalität der ersten Iteration

In Abbildung 4 ist der Programmablauf beim Aktivieren einer Schaltfläche durch eine Person zu sehen. War bereits eine Reihe aktiviert, muss die Person einen *Generic Button* betätigt haben. Da diese Iteration der Tastatur nicht dem konventionellen Layout entspricht und die Umschalttaste ebenfalls die Funktion der a*ltGr* Taste übernimmt, benötigt jeder *Generic Button* vier Variablen für VK:

- einen Primärwert
- einen Sekundärwert
- zwei Variablen zum Modifizieren der beiden Werte

Letztere können den VK für *shift*, *altGr* oder einen Wert, der keine Eingabe simuliert, enthalten. Wird zum Beispiel die Taste zum Simulieren des Zeichens 'a' gedrückt, gleichen der Primärwert sowie der Sekundärwert dem **VK = 65**, welcher der physischen 'A'-Taste einer konventionellen Tastatur entspricht. Die Variable zum Modifizieren des Primärwertes, der beim Senden ans Betriebssystem zu keinen Veränderungen führt, ist **VK = 255**, während die Variable zum Modifizieren des Sekundärwertes (**VK = 165**) dem Betriebssystem das Drücken der Umschalttaste signalisiert. Die Variable zum Modifizieren des Primärwertes ist nötig, wenn die Person eine Taste betätigt, die vom konventionellen Layout einer Tastatur abweicht, wie zum Beispiel die '!'-Taste.

In er vorliegenden Iteration kann dieses Zeichen ohne Modifikatoren ausgegeben werden, jedoch wird beim Standardlayout die Tastenkombination aus Umschalt- und '1'-Taste benötigt, um zum selben Ergebnis zu gelangen. Die Specialbuttons beinhalten jeweils den VK für ihre Funktion. Eine Ausnahme bildet die Umschalttaste, die hier zwei Flags berücksichtigt und dem Programm signalisiert, ob die Umschalt- beziehungsweise Feststelltaste aktiv ist. Diese werden von den *Generic Buttons* überprüft, um festzustellen, ob der Primärwert oder der Sekundärwert ausgegeben werden soll.

Da eine hohe Tippgeschwindigkeit erheblich die User Experience steigert, wurde ein simpler Autovervollständigungsalgorithmus implementiert. Hat die Person mehr als zwei Zeichen geschrieben, gibt der Algorithmus drei Wörter zurück, die den bereits getippten Wortstamm enthalten und oft in deutschen Texten vorkommen. Hierfür fanden die folgenden externen Quellen Verwendung:

https://dumps.wikimedia.org/ ist eine Website, die eine vollständige Kopie aller Wikimedia Wikis in verschiedenen Sprachen bereitstellt [15]. Mit diesen Daten und einem in Kombination mit einem Python Script (GitHub - gambolputty/dewiki-wordrank: Tab-delimited word frequency list compiled from the German Wikipedia) wurde ein Datensatz erstellt, der in jeder Zeile einer Textdatei ein Wort sowie seine Frequenz des Vorkommens speichert [16].

Die Basis der Autovervollständigung bildet ein Trie. Ein Trie (retrieval tree) ist eine Art k - artiger Suchbaum also eine Baumdatenstruktur, die zur Suche nach einem bestimmten Schlüssel innerhalb einer Menge verwendet wird. Er ist speziell für das schnelle Auffinden von Daten entlang eines Pfades konzipiert.

Jeder Knoten des Baumes enthält einen Boolean **isEndNode**. Dieser signalisiert, ob der Knoten das Ende eines Wortes beschreibt. Etwas unintuitiv kann jeder Knoten innerhalb des Baumes diesen Boolean als **Wahr** gesetzt haben: das Wort „Baum" ist nach dem 'm' zu Ende, allerdings gibt es im Deutschen viele Wörter mit demselben Wortstamm („Baumstamm", „Baumkrone", usw.). Außerdem hat jeder Knoten ein Wörterbuch, in dem seine Kinder gespeichert sind, jeweils mit einem im Wörterbuch einzigartigen Buchstaben. Jeder Knoten, der als Ende markiert ist, besitzt einen zusätzlichen Eintrag in seinem Wörterbuch, der als Schlüssel nur eine Zahl enthält, welche beschreibt, mit die Häufigkeit das gefundene Wort in Wikipedia-Artikeln auftritt.

Für die automatische Wortvervollständigung werden Funktionen zum Einfügen sowie Suchen nach Schlüsselwörtern im Baum benötigt. Das Einfügen eines Wortes im Baum verläuft folgendermaßen: die Wurzel wird als Startknoten designiert und das einzufügende Wort Zeichen für Zeichen betrachtet. Ist das erste Zeichen im Wörterbuch der Wurzel nicht als Schlüssel vorhanden, erfolgt dafür ein Eintrag. Danach, oder wenn das Zeichen bereits im Wörterbuch zu finden war, wird der Knoten mit dem korrekten Zeichen als nächster zu betrachtender Knoten gewählt und das zweite Zeichen im Wort betrachtet. Dieser Prozess wiederholt sich bis zum letzten Zeichen des Wortes. Dort angelangt, wird dem vermutlich neu erstellten Knoten der Boolean **isEndNode** auf **Wahr** gesetzt und ein zusätzlicher Eintrag im Wörterbuch für die Häufigkeit auf Wikipedia angelegt.

Um herauszufinden, ob ein Wort im Baum vorhanden ist, erfolgt ein Neustart von der Wurzel aus und das Wort wird Zeichen für Zeichen betrachtet. Wenn das derzeitige Zeichen im Wörterbuch des aktuellen Knotens enthalten ist, wird das darauffolgende Zeichen mit dem gefundenen Kind überprüft. Ist an irgendeinem Knoten nicht das gesuchte Zeichen im Wörterbuch, dann ist das Wort nicht im Baum vorhanden. Wurde das letzte Zeichen des Wortes gefunden, aber der zugehörige Knoten ist kein Endknoten, ist das Wort ebenso nicht im Baum als Wort gespeichert: Es könnte zum Beispiel nach „Baumhau" gesucht worden sein, eine Zeichenfolge, die im Baum enthalten, aber nur ein Teilwort ist. Ist der letzte Knoten jedoch ein Endknoten, dann gilt das Wort als gefunden.

Zusätzlich gibt es eine Funktion, die, ab einem ausgewählten Knoten im Baum, alle Wörter findet und zurückgibt. So können alle Wörter mit demselben Wortstamm gefunden werden. Hat der bestimmte Knoten Kinder, wird für jedes dieser Kinder der zugehörige Buchstabe an den Wortstamm angehängt und überprüft, ob damit ein Wortende gefunden ist. Falls dies der Fall sein sollte, wird das gefundene Wort, zusammen mit seiner Häufigkeit, in einer Liste gespeichert. Nun erfolgt der rekursive Aufruf der Funktion, bis keine Kinder mehr zum Prüfen vorhanden sind.

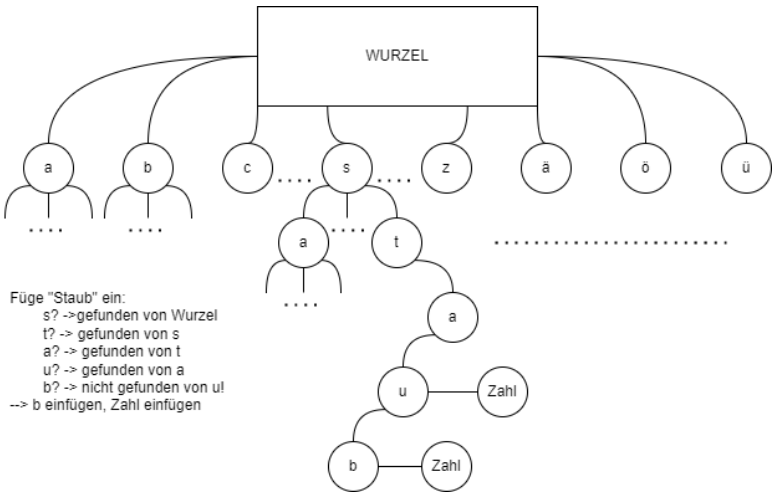

Abbildung 6: Prozess des Suchens und des Einfügens in einen Trie

Während der Testphase hat sich ein grundlegendes Problem dieses Layouts herausgestellt. Die Programmfläche der Tastatur nimmt einen Großteil des Bildschirms ein, um eine optimale Funktionalität zu erreichen. Die Notwendigkeit entsteht durch die hohe Anzahl der Schaltflächen, die ihre minimale Größe haben müssen, um bei den Personen akkurat die visuellen Stimuli auszulösen.

2. Iteration II

Das Ziel der zweiten Iteration ist es, die Größe stark zu verringern, um das oben genannte Problem der Benutzerfreundlichkeit aus Iteration I zu beheben. Um dies zu erreichen, bot sich ein hierarchiebasiertes UI-Design an.

Ausgehend vom Hauptmenü wird die Tastatur gesteuert. Dabei sind im Hauptmenü zunächst die drei Kategorien „Buchstaben und Zahlen“, „mathematische Zeichen”, „Navigation und Tastenkürzel” verfügbar. Durch die Auswahl eines dieser Menüpunkte wechselt die Sicht auf ein Untermenü, in dem spezifische Buchstaben, Zahlen und Symbole ausgewählt werden können. Nach der erfolgreichen Eingabe, zum Beispiel eines Buchstabens, gelangt man zurück in das vorherige, hierarchisch höhere Menü. Eine Ausnahme hiervon sind das Leerzeichen sowie die Löschtaste, da es nach deren Benutzung sinnvoller ist, im Buchstabenmenü zu bleiben. Eine Autovervollständigung hilft zudem beim Tippen und beschleunigt den Prozess.

Um die Zahlen schneller zu erreichen und den Schreibprozess fließend zu gestalten, wurde das Zahlenmenü an mehreren Stellen implementiert. Somit ist es sowohl über den Kontext „mathematische Zeichen“ als auch im Buchstabenmenü über das Segment „Satzzeichen und Zahlen” erreichbar. Im letzten vom Hauptmenü erreichbaren Unterpunkt „Einstellungen” kann man beispielsweise die Auslösegeschwindigkeit individuell anpassen. Die Einstellungsmöglichkeiten wurden bisher noch nicht implementiert, da diese für die Fragestellung der Bachelorarbeit nicht von Bedeutung sind.

Dadurch konnten mehrere Schaltflächen zusammengeführt und somit im Programmfenster die Anzahl der Tasten reduziert werden. Zum einen konnte so die Gesamtgröße des Programms deutlich verringert werden. Zum anderen ist es möglich, mehr Untermenüs für weitere Zeichen und Funktionen anzulegen, da eine solche Hierarchie beliebig tief konstruiert werden kann.

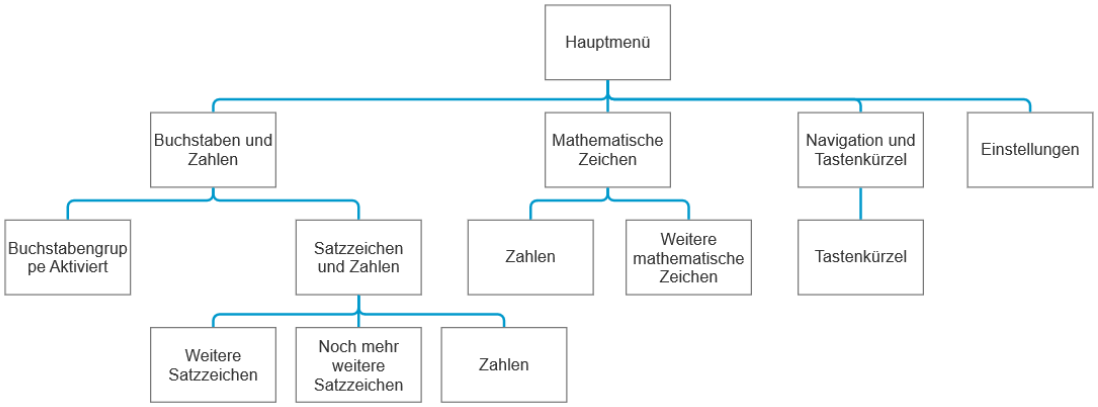

Abbildung 7: Aufbau der Hierarchie der zweiten Iteration

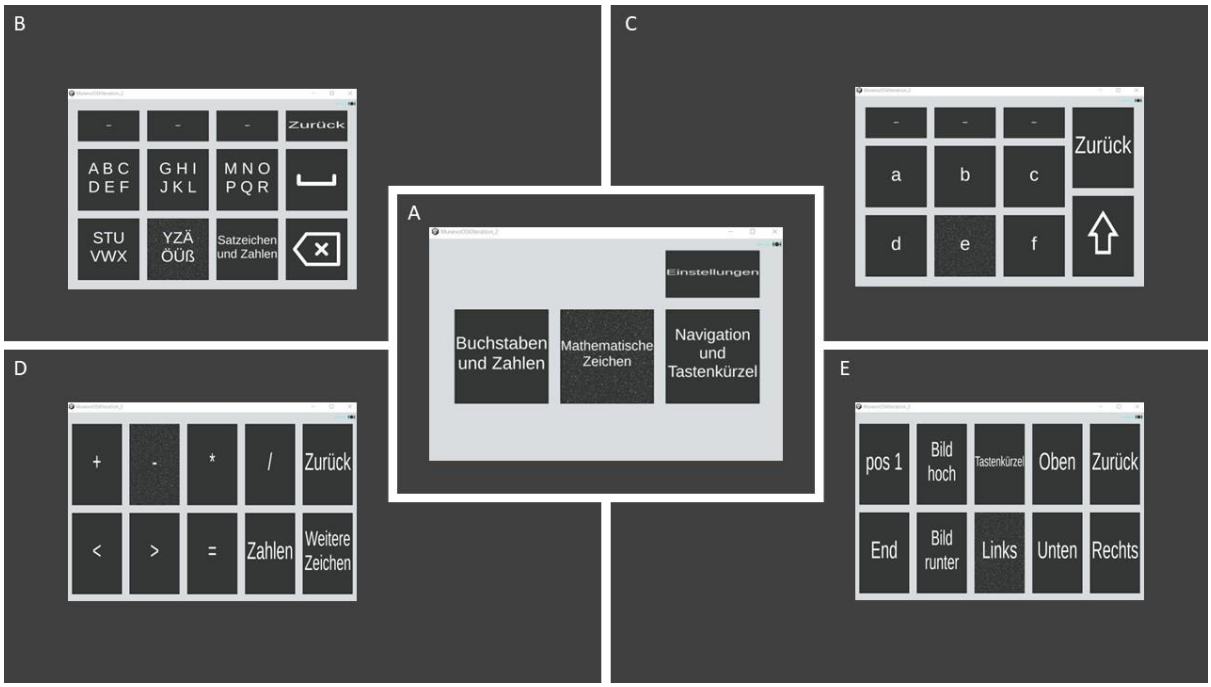

Abbildung 8: Ausschnitte aus der Bedienoberfläche der Iteration II. A) Hauptmenü B) Buchstaben und Zahlen C) Buchstabengruppe a bis f D) mathematische Zeichen E) Tastenkürzel

Allerdings benötigt die Person mit wachsender Tiefe der Hierarchie immer mehr Klicks, um an die gewünschte Schaltfläche zu gelangen. Daher ermöglicht dieses Design eine Priorisierung von wichtigen Schaltflächen in einem oben gelegenen Teil der Hierarchie. Weniger frequent genutzte Schaltflächen sind tiefer in der Hierarchie gelegen. Zum Beispiel können alle Buchstaben des Alphabets mit zwei Auslösungen erreicht werden. Aus eigener Erfahrung seltener genutzte Standardzeichen hingegen benötigen bis zu drei Kontextwechsel in der Hierarchie. Somit lässt sich dieser Nachteil zu einem gewissen Teil ausgleichen.

Durch eine klare Strukturierung des Projekts in einzelne Module, konnte der Arbeitsaufwand bei Iteration II deutlich geringer gehalten werden, da große Teile aus Iteration I

wiederverwendet wurden. Die grundlegende Funktionsweise der Tasten aus Iteration I unterscheidet sich nur in wenigen Details:

Einerseits wurden die *RowChooser* Buttons komplett weggelassen, da das Bedienkonzept dieser Iteration ein gänzlich anderes ist. Andererseits er folgte die Erweiterung der Special Buttons um Navigationskomponenten.

Die automatische Wortvervollständigung konnte gleichwertig aus Iteration I übernommen werden. Der größte Entwicklungsaufwand dieser Iteration bestand darin, die Logik für die Navigation durch die Hierarchie zu implementieren. Jeder Teil der Hierarchie ist hier als View definiert. Zum Anfang des Programms wird ein Stapel initialisiert, der das Hauptmenü beinhaltet. Navigiert die Person erfolgreich in das nächste Menü, wird dieses dem Stapel hinzugefügt. So ist immer sichergestellt, dass der momentane Menüpunkt immer ganz oben auf dem Stack liegt. Umgekehrt wird das oberste Element des Stapels gelöscht, wenn man zurück navigiert.

3. Iteration III

Abbildung 9: Screenshot - Iteration III

Die zweite Iteration hat den Nachteil, dass der Aufbau unintuitiv ist. Bei der dritten Iteration wurde daher das deutsche Standardtastaturlayout als Basis verwendet. Dadurch nähert es sich von der Größe wieder eher an Iteration I an, sollte aber intuitiver zu bedienen sein. Da nur zehn NeuroTags simultan aktiv sein können, ist das Layout in farblich differenzierte Unterbereiche der Tastatur aufgeteilt. Nach der Aktivierung eines Bereiches können die darin vorhandenen neun Tasten ausgewählt werden.

Es wurde zusätzlich eine Zurücktaste eingeführt, die fokussiert werden kann, wenn sich die Person in der Auswahl eines Unterbereichs befindet. Eine fehlerhafte Auswahl eines Bereiches kann mit dieser Taste rückgängig gemacht werden: man gelangt zum Startpunkt zurück. Nach der erfolgreichen Aktivierung einer Taste werden die farblich markierten Zonen auf der Tastatur wieder sichtbar. Neben den farbigen Unterkategorien sind in diesem Layout noch drei weitere oft verwendete Knöpfe auswählbar:

- Leertaste
- Löschtaste
- Möglichkeit zur Auswahl für die Autovervollständigung

V. Studiendesign

Zur Evaluation der drei konstruierten Iterationen erfolgte die Konstruktion eines Versuchsaufbaus. Hierfür wurden fünf Personen mit einem Alter zwischen 20 und 50 Jahren ausgewählt. 40% diese Personen hatten eine Sehbeeinträchtigung.

Sowohl Menschen mit körperlicher Beeinträchtigung wie SMA (Spinale Muskelatrophie) als auch Testpersonen ohne körperliche Einschränkungen waren Teil der Studie. Allerdings konnte eine Testperson nicht inkludiert werden, da diese Person aufgrund ihrer stark fortgeschrittenen körperlichen Behinderung dauerhaft auf eine Kopfstütze angewiesen ist. Somit war das Platzieren der *NextMind*-Hardware am Hinterkopf nicht möglich.

Das Ziel dieser Mini-Studie war es, erste Einblicke in das Verhalten und Empfinden der nutzenden Personen zu gewinnen, um eine initiale Einschätzung zur Bedienfreundlichkeit der einzelnen Iterationen vornehmen zu können. Eine Testperson war mit den Layouts bereits vertraut. Hier galt es herauszufinden, ob durch eine längere Benutzung der Layouts eine Verbesserung in der Tippgeschwindigkeit und/oder -genauigkeit eintritt.

Tab. 1: Testpersonen der Studie

	Alter	Sehbeeinträchtigung	Körperliche Beeinträchtigungen	Vorerfahrung
Testperson 1	24	Nein	Nein	Nein
Testperson 2	26	Ja	Ja	Ja
Testperson 3	48	Ja	Nein	Nein
Testperson 4*	21	Nein	Ja	Nein
Testperson 5	34	Nein	Nein	Nein

*konnte nicht eingeschlossen werden

1. Testbedingungen

Für die optimale Benutzung wurde das *NextMind*-System einmalig pro Person kalibriert. Dabei musste sich die Testperson wiederholt auf einen Kreis mit einem NeuroTag fokussieren und erhielt nach ca. 45 Sekunden einen Wert zwischen 1-5. Eine 5 bedeutet hierbei eine perfekte Kalibrierung, ab 3 ist das Gerät nutzbar. Darunter sollte man eine Neukalibrierung vornehmen. Wenn das Gerät während der Testphase verrutschte oder abgenommen wird, muss eine komplette Neukalibrierung durchgeführt werden. Zur Vergleichbarkeit wurde der Test nur bei einem Wert von 4 oder 5 durchgeführt.

Die Testzeit betrug ca. 90 Minuten, wobei die Reihenfolge zufällig ausgewählt und nach jedem Test einer Iteration eine Pause eingelegt wurde. Dies sollte dazu beitragen, den Einfluss von Ermüdungserscheinungen auf das Resultat zu verringern. Während der gesamten Test- und Pausenzeit wurde das Gerät nicht abgelegt, um eine erneute Kalibrierung zu vermeiden und zusätzlich den Tragekomfort über einen längeren Zeitraum in die Studie mit einzubeziehen.

Um bestmögliche und einheitliche Voraussetzungen zu schaffen, haben alle Testpersonen den Test im selben Raum und am selben Bildschirm durchgeführt. Dabei wurde auf einen festen Abstand zum Bildschirm und eine vorher definierte Größe von 27" des Bildschirms geachtet. Um ein optimales Ergebnis zu erzielen, erfolgte der Test unter gleichbleibenden Lichtverhältnissen, da das System auf visueller Basis funktioniert.

Für jedes der drei Layouts gab es eine kurze Eingewöhnungszeit von zwei Minuten. Danach musste jede Testperson den gleichen Satz fehlerfrei tippen „Zwölf Boxkämpfer jagen Viktor quer über den großen Sylter Deich. 123!". Dieser Satz enthält alle Buchstaben des deutschen Alphabets inklusive einiger Umlaute, Zahlen und Sonderzeichen. So konnte die Tippgeschwindigkeit und die Korrektheit der ausgelösten Tasten zwischen den verschiedenen Teilnehmern und den Tastaturen verglichen werden. Korrekturen während des Schreibens waren erlaubt.

Innerhalb der Bildschirmpause zwischen den jeweiligen Iterationen wurde der Fragebogen zur Tastatur ausgefüllt. Die Fragen wurden vorgelesen und zusammen mit den Testpersonen ausgefüllt.

2. Auswertung der Studie

Der Fragebogen zur anschließenden Auswertung der Studie ist in verschiedene Abschnitte unterteilt (s. Anhang). Dabei wurde zuerst eine kurze allgemeine Befragung zum *NextMind*-System und den jeweiligen Testpersonen durchgeführt. Zudem er folgte zu jeder Iteration die individuelle Evaluation der Benutzerfreundlichkeit. Die Tippgeschwindigkeit und -genauigkeit wurden manuell während des Testprozesses aufgenommen.

Der erste zu betrachtende Faktor war die Geschwindigkeit. Diese wurde ab dem ersten Tastenanschlag bis zum letzten getippten Zeichen des Satzes gemessen. Hierfür kam eine Stoppuhr zum Einsatz. Daraus konnten im Anschluss sowohl die Anzahl der Anschläge pro Minute berechnet werden, als auch die Durchschnittsgeschwindigkeit über alle Testpersonen.

Der zweite relevante Messwert ist die Genauigkeit. Dabei wurde während des Schreibprozesses jeder Tippfehler erfasst und die Anzahl addiert - auch wenn der jeweilige Fehler korrigiert wurde. Abschließend erfolgte eine zusätzliche Überprüfung des geschriebenen Satzes auf Fehlerfreiheit. Hierbei wurde die Anzahl der jeweils von der Person nichtkorrigierten Fehler notiert. Daraus ergaben sich die Werte zur Bestimmung der effizientesten Tastatur.

VI. Evaluation

Folgende Ergebnisse wurden von den Testpersonen erfragt und entsprechend ausgewertet:

1. Ablauf der Kalibrierung

Der erste Schritt im Versuchsablauf war die selbstständige Kalibrierung des *NextMind*-Systems. Die Testpersonen konnten die Schwierigkeit der Kalibrierung bewerten (Abb. 9). Dabei zeigte sich, dass die Kalibrierung für die Testpersonen intuitiv war.

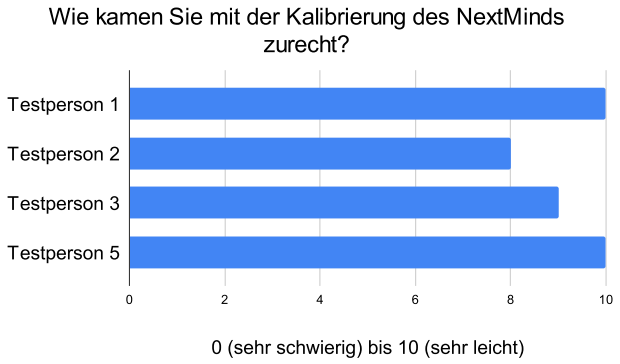

Abbildung 10: Bewertung des Kalibrierungsprozesses durch die Testpersonen

Die Einschätzung der Testpersonen bezüglich des Komforts beim längeren Tragen des Gerätes lieferte kein eindeutiges Ergebnis (Abb. 10). Eine der Testpersonen bewertete das Tragen als stark unangenehm (2 auf der Skala), während eine andere das Gerät als nicht wahrnehmbar (10 auf der Skala) angab. Der Rest der Testpersonen verteilt sich im mittleren Bereich der Skala.

Eine weitere Frage zum Gerät, die den Testpersonen gestellt wurde, bezog sich auf die Alltagstauglichkeit des *NextMind*-Systems (Abb. 10). Zwei Testpersonen können sich vorstellen, *NextMind* auf der Arbeit oder in der Universität zu verwenden. Andererseits tendierten zwei Testpersonen eher gegen eine Benutzung im Alltag.

2. Bewertung der drei Iterationen durch die Testpersonen

Der Fragebogen bot den Testpersonen die Möglichkeit, die verschiedenen Tastaturen zu bewerten in den folgenden Kategorien:

- intuitive Bedienung
- Design
- allgemeine Benutzerfreundlichkeit

- Anstrengung bei der Verwendung

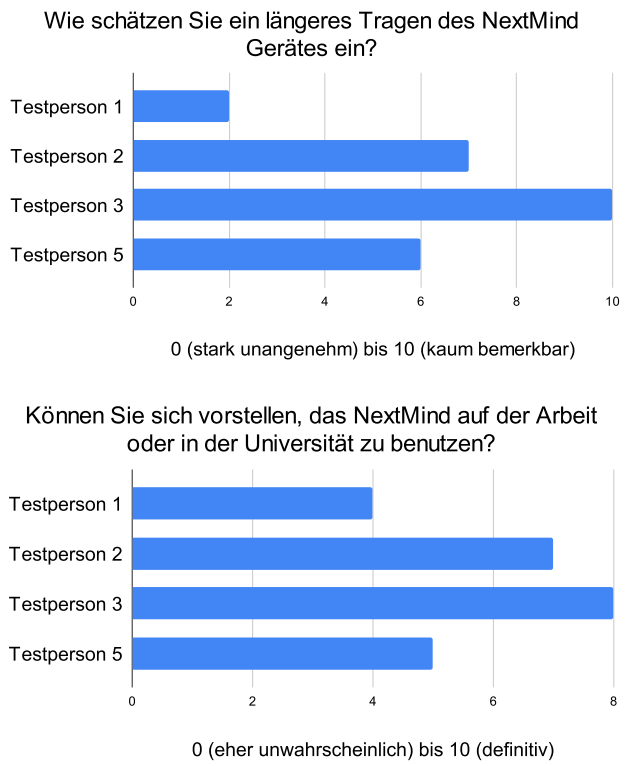

Abbildung 11: Bewertung des Komforts und der Alltagstauglichkeit durch die Testpersonen

Mit Iteration I kamen die Testpersonen am schlechtesten zurecht. Die beste Bewertung bekam diese Iteration in Bezug auf ihre intuitive Bedienung. Diese lag trotzdem deutlich unter dem Durchschnitt der anderen beiden Iterationen. Zudem wurde das Design nur als mittelmäßig gut eingeschätzt. Die negativen Bewertungen bezüglich der intuitiven Bedienung werden dadurch begründet, dass die Testpersonen während der Verwendung dieser Iteration durchgängig Anstrengung und zum Großteil Frustration während der Benutzung dieser Iteration erfuhren. Deshalb wurde die Bedienung dieser Tastatur als stark belastend wahrgenommen (Abb. 12).

Die zweite Iteration zeigte hingegen eine deutliche Verbesserung in allen Bereichen (Abb. 13). Die intuitive Bedienbarkeit dieser Tastatur wurde fast ausschließlich mit Höchstnoten bewertet. Nur eine Testperson hat diese Iteration mit einer 6 von 10 auf der Skala bewertet. Ähnliche Ergebnisse zeigten sich auch bei der Benutzerfreundlichkeit der Tastatur. Im Schnitt wurden 9 von 10 Punkten vergeben, mit 8 von 10 Punkten als niedrigste Bewertung und 10 als höchste. Mit exakt denselben Bewertungen wurde auch das Design dieser Tastatur bedacht.

Da von allen Testpersonen Iteration II als nicht anstrengend klassifiziert wurde, kann die eindeutige Angabe von Zufriedenheit aller Testpersonen mit Iteration II erklärt werden.

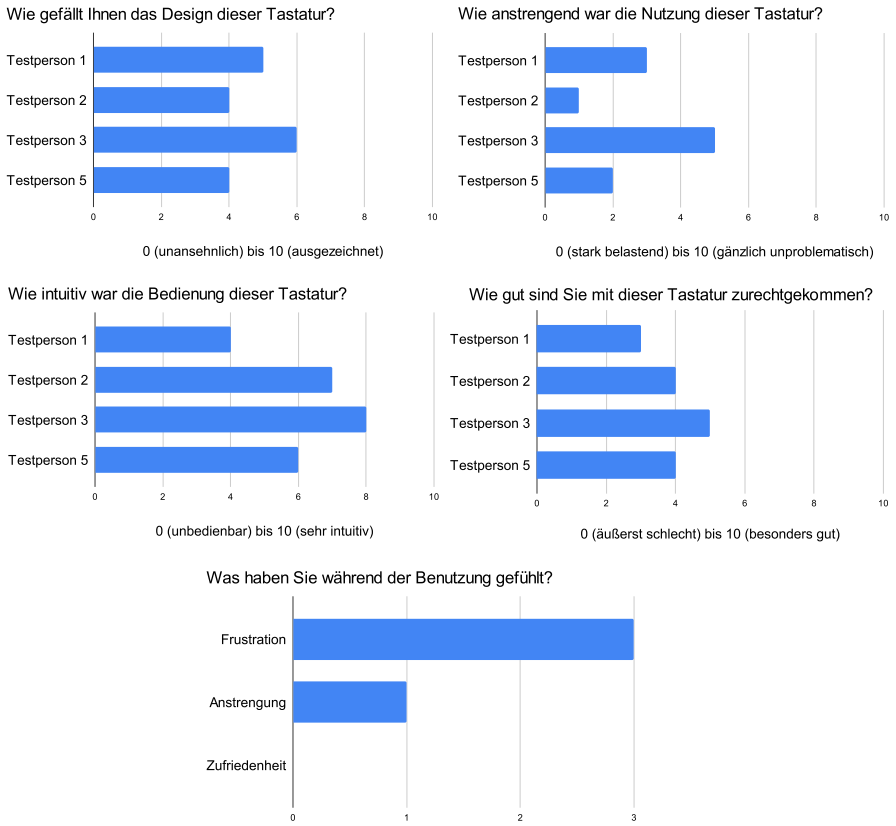

Abbildung 12: Ergebnisse aus den Umfragen für Iteration I

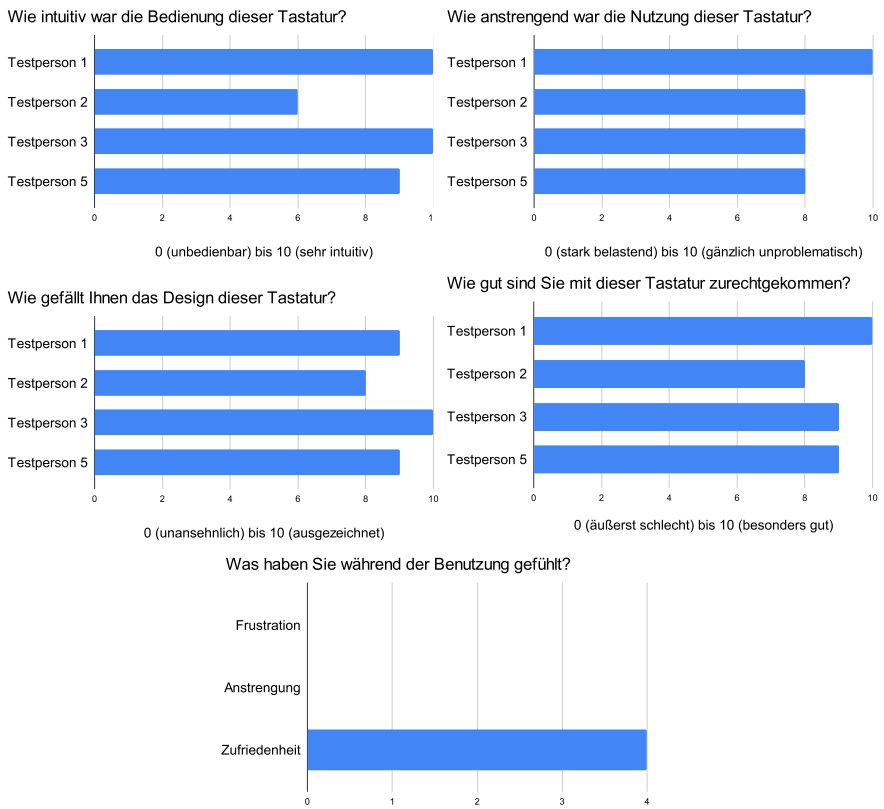

Abbildung 13: Ergebnisse aus den Umfragen für Iteration II

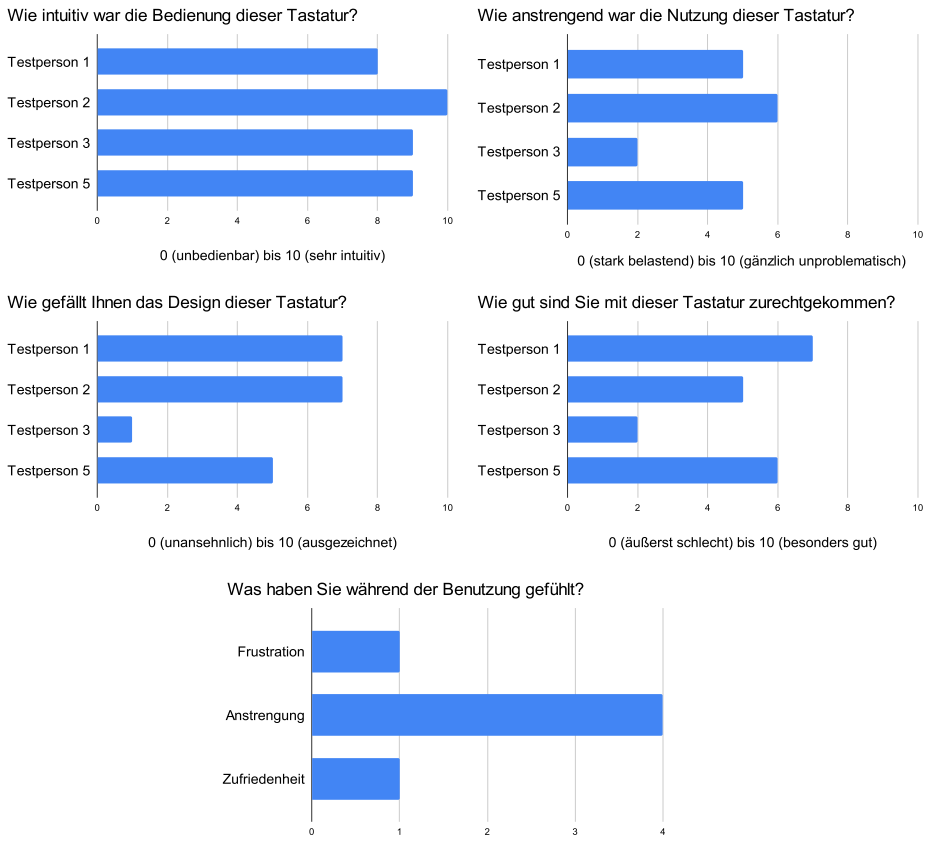

Abbildung 14:Ergebnisse aus den Umfragen für Iteration III

Das letzte von den Testpersonen bewertete Programm war die dritte Iteration (Abb. 14). Sie erhielt die besten Bewertungen von allen drei Tastaturen in puncto intuitive Bedienung. Allerdings wurde das Design mit einer Höchstwertung von 7 von 10 möglichen Punkten jedoch eher als mittelmäßig eingeschätzt. Eine Testperson bewertete das Design sogar lediglich mit einer 2 von 10 Punkten. Im Vergleich zu Iteration I wurde dieses Layout im Hinblick auf seine Bedienfreundlichkeit zwar besser bewertet, schnitt aber deutlich schlechter ab als Iteration II. Alle Testpersonen empfanden die Benutzung dieser Iteration anstrengend, eine Testperson sogar sehr anstrengend (2 von 10 möglichen Punkten). Die Bewertungen bezüglich der allgemeinen Zufriedenheit fielen verglichen mit Iteration I besser aus. Das spiegelt sich auch in der Bewertung der Anstrengung wider, die als eher mittel bis hoch angesehen wurde. Auch hier liegt die Bewertung durchschnittlich höher als bei Iteration I.

3. Analyse der ermittelten Daten

Die Analyse der empirisch erhobenen Daten bestätigt das Ergebnis der individuellen Bewertungen der Testpersonen. Für jede Iteration wurden die geschriebenen Buchstaben pro Minute, Präzision bzw. Fehlerfreiheit und korrigierte Fehler gemessen und jeweils der Durchschnitt aller Testpersonen ermittelt (Tab. 2-4).

Im Durchschnitt erreichte Iteration II die höchste Präzision. Mit diesem Layout schafften die Testpersonen auch die höchste Geschwindigkeit: 7,22 Buchstaben pro Minute. Somit war die Tippgeschwindigkeit um zirka drei Buchstaben pro Minute höher, lag also im Vergleich zu den Iterationen I und II fast doppelt so hoch. Der einzige Punkt, in dem die Iteration II nicht am besten abschnitt, war die Anzahl der behobenen Fehler: bei Iteration I lag diese um ca. 16% höher. Trotzdem erwies sich die Präzision in dieser Tastatur als am geringsten. Iteration III hatte die niedrigste Anzahl an behobenen Fehlern mit 68%. Die Testpersonen konnten dennoch etwas schneller und präziser mit Iteration III arbeiten als mit Iteration I.

Tab. 2: Testergebnisse Iteration I

Iteration I			
	Präzision in %	Behobene Fehler in %	Geschwindigkeit in Buchstaben pro Minute
Testperson 1	91,30%	83,30%	3,33
Testperson 2	86,96%	77,80%	3,18
Testperson 3	89,86%	100,00%	3
Testperson 5	88,41%	87,50%	3,15
---	---	---	---
Durchschnitt	89,13%	87,15%	3,17

Tab. 3: Testergebnisse Iteration II

Iteration II			
	Präzision in %	Behobene Fehler in %	Geschwindigkeit in Buchstaben pro Minute
Testperson 1	98,55%	100,00%	8,33
Testperson 2	95,65%	33,30%	6,62
Testperson 3	98,55%	100,00%	6,31
Testperson 5	97,10%	50,00%	7,62
---	---	---	---
Durchschnitt	97,46%	70,83%	7,22

Tab. 4: Testergebnisse Iteration III

Iteration III			
	Präzision in %	Behobene Fehler in %	Geschwindigkeit in Buchstaben pro Minute
Testperson 1	100,00%	66,70%	5,07
Testperson 2	88,41%	50,00%	4,09
Testperson 3	95,65%	100,00%	2,67
Testperson 5	89,86%	57,10%	4,03

Durchschnitt	93,48%	68,45%	3,97

4. Fazit

Die Erwartungshaltung vor der Durchführung der Studie bestand, dass Iteration III sich als die intuitivste Version erweisen würde, da diese Iteration dem deutschen Tastaturlayout gleicht und somit einem bereits gut bekannten Muster folgt. Diese Hypothese konnte während der Studie bestätigt werden. Entgegen der Erwartung schnitt Iteration II im Hinblick auf ihre Intuitive Bedienbarkeit besser ab als Iteration I, obwohl deren hierarchische Struktur und Benutzung keinem üblichen Muster entsprechen, wohingegen Iteration I auf der Struktur des Deutschen Alphabets basiert. Iteration I hatte aber den bereits antizipierten Nachteil der Unhandlichkeit durch die Größe des Programmfensters.

Iteration III irritierte die Testpersonen durch ihre Farbigkeit, was das fokussierte Arbeiten erschwerte. Aufgrund der insgesamt geringeren Anzahl der Bedienelemente bei Iteration II konnte deren Größe der Elemente jeweils erhöht werden – die Bedienung der Tasten wurde hierdurch vereinfacht.

Im Hinblick auf den Vergleich der empirischen Daten mit den Bewertungen der Testpersonen zeigte sich, dass diese zum Großteil übereinstimmen. Bei Iteration III ist die Abweichung zwischen den individuellen Bewertungen der einzelnen Testpersonen am deutlichsten. Dies spiegelt sich auch in der jeweiligen Tippgeschwindigkeit wider. Die Abweichungen lassen sich nicht durch eine etwaige Sehbeeinträchtigung erklären, da diese keinen relevanten Effekt auf die Ergebnisse hatte.

Aus den erhobenen Daten ist ersichtlich, dass Iteration II am effizientesten war, während Iteration I am wenigsten in Punkto Präzision und Tippgeschwindigkeit überzeugen konnte. Diese

Erkenntnisse lassen sich auch in der Gesamtbewertung durch die Testpersonen wieder finden (Abb. 15).

Die behobenen Fehler weichen von der Gesamtbewertung ab. Dies könnte auf die verschiedenen Bedienoberflächen zurückzuführen sein. Bei Iteration I sind alle Tasten ohne Kontextwechsel erreichbar. Wenn eine falsche Reihe ausgewählt wird, muss hier jedoch ein falscher Buchstabe gedrückt werden. Dadurch ist sowohl die Motivation höher als auch die Hürde geringer, eine falsche Eingabe zu korrigieren. Bedingt durch die hierarchische Struktur der Benutzeroberfläche von Iterationen II ist es hier komplizierter und zeitintensiver, auf die richtige Ebene zu navigieren: deshalb wurden aufgetretene Fehler nicht immer korrigiert. Im Hinblick auf die Bedienoberfläche ohne Kontextwechsel ähneln sich Iteration I und Iteration III. Das Hauptproblem hier war die Farben, die die Testpersonen irritiert und das Fokussieren schwieriger gemacht haben, weshalb Fehler wiederum nicht behoben wurden.

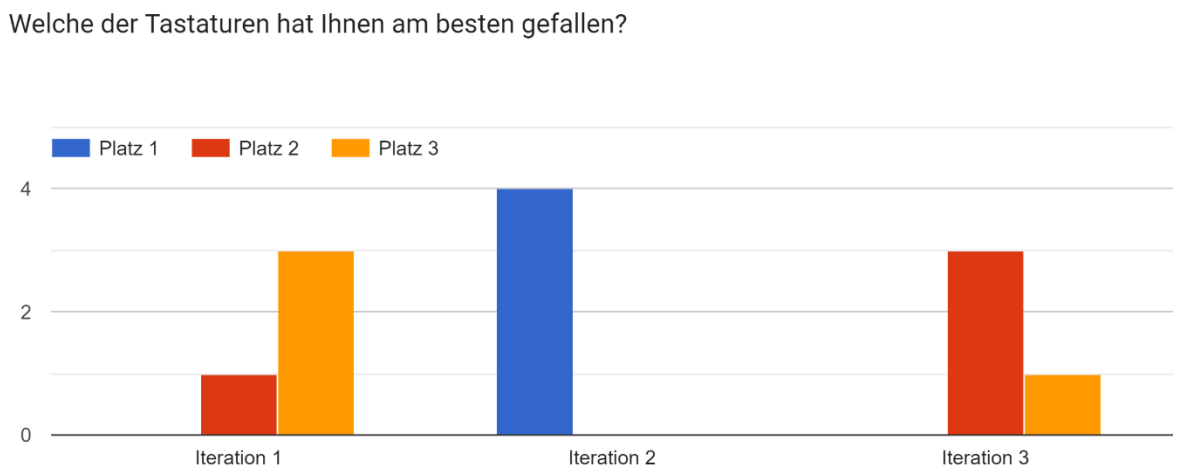

Abbildung 15: Allgemeine Bewertung der Iterationen durch die Testpersonen

Im Verlauf der Studie fiel auf, dass einige Faktoren keine Berücksichtigung gefunden hatten: beispielsweise wurden weder die Haarlänge noch -dichte der der Testpersonen beachtet. So zeigte sich während der Versuchsablaufs, dass diese Faktoren einen starken Einfluss insbesondere auf die Kalibrierung und den Tragekomfort der Hardware hatten. Jedoch schien es diesbezüglich keine Auswirkungen auf die Leistung im Test selbst zu geben.

Diese Arbeit zeigt auf, dass das Schreiben von Texten mittels einer ausschließlich BCI gesteuerten Lösung möglich ist. Es konnten drei unterschiedliche Tastaturen entwickelt und mit

unterschiedlichen Ergebnissen getestet werden. Iteration II erwies sich als die Option mit der höchsten allgemeinen Funktionalität. Sie könnte daher eine mögliche Alternative für Menschen mit Behinderung darstellen, bei denen andere Texteingabemöglichkeiten versagen. Dennoch stieß der gewählte Ansatz selbst unter kontrollierten Bedingungen schon an Grenzen: bereits in dieser kleinen Gruppe von Testpersonen gab es eine Person, der es aufgrund ihrer körperlichen Einschränkungen nicht möglich war, das *NextMind*-System am Kopf befestigen zu lassen. Diese Person konnte also den Schreibprozess von vornherein gar nicht ausführen.

VII. Ausblick

Aufgrund der Testergebnisse kann davon ausgegangen werden, dass eine zukünftige Weiterentwicklung der Bedienoberfläche in Richtung Iteration II als sinnvoll erscheint. Um die Bedienbarkeit weiter zu steigern, müssten die bisher lediglich als Platzhalter existierenden Funktionalitäten implementiert werden, wie zum Beispiel die in den Einstellungen anpassbare Auslöseschwelle der *NeuroTags*. Ideal wäre eine individuell anpassbare Gestaltung der Bedienoberfläche. Dies würde zusätzlich die User Experience signifikant verbessern. Da *NextMind* jedoch von *Snap* akquiriert wurde und demzufolge die Zukunft der Hardware unsicher ist, wäre es nun mehr notwendig, das Prinzip der *NeuroTags* auf andere BCIs zu übertragen.

Des Weiteren wäre es sinnvoll eine größere Anzahl an Testpersonen zu akquirieren, um herauszufinden, ob sich die hier gezeigten Tendenzen bestätigen.

Grundsätzlich ließen sich einige Probleme, wie die sich wiederholende Kalibrierung, der schlechte Tragekomfort oder die geringe Zuverlässigkeit und Geschwindigkeit der Auslösungen damit beheben, auf invasive Geräte wie Neuralink oder Synchron zurückzugreifen. Allerdings liegt diesem Ansatz ein deutlicher Mehraufwand sowie das Risiko einer Operation am Gehirn inne.

VIII. Literaturverzeichnis

[1] G. R. Müller-Putz, „Electroencephalography“, in *Handbook of Clinical Neurology*, Bd. 168, Elsevier B.V., 2020, S. 249–262. doi: 10.1016/B978-0-444-63934-9.00018-4.

[2] K. Blinowska und P. Durka, „ELECTROENCEPHALOGRAPHY (EEG)“.

[3] M. Teplan, „FUNDAMENTALS OF EEG MEASUREMENT“, *MEASUREMENT SCIENCE REVIEW*, Bd. 2, Nr. 2, 2002.

[4] T. Ball, M. Kern, I. Mutschler, A. Aertsen, und A. Schulze-Bonhage, „Signal quality of simultaneously recorded invasive and non-invasive EEG“, *Neuroimage*, Bd. 46, Nr. 3, S. 708–716, Juli 2009, doi: 10.1016/J.NEUROIMAGE.2009.02.028.

[5] „Overview – NextMind“. https://www.next-mind.com/documentation (zugegriffen Sep. 14, 2022).

[6] „Class NeuroTag“. https://www.next-mind.com/documentation/unity-sdk/api-reference/api/NextMind.NeuroTags.NeuroTag.html (zugegriffen Sep. 14, 2022).

[7] „Körperbehinderung: Erscheinungsformen und Schweregrade“. https://behinderung.org/gesetze/koerperbehinderung.htm (zugegriffen Sep. 20, 2022).

[8] H. S. Grewal, A. Matthews, R. Tea, V. Contractor, und K. George, „Sip-and-Puff Autonomous Wheelchair for Individuals with Severe Disabilities“, *2018 9th IEEE Annual Ubiquitous Computing, Electronics and Mobile Communication Conference, UEMCON 2018*, S. 705–710, Nov. 2018, doi: 10.1109/UEMCON.2018.8796679.

[9] „Dragon Professional Anywhere Speech Recognition Solution | Nuance“. https://www.nuance.com/dragon/business-solutions/dragon-professional-anywhere.html (zugegriffen Sep. 24, 2022).

[10] S. N. Kirillov, O. E. Shustikov, und V. T. Dmitriev, „Natural Language Voice Control Interface for People with Special Needs and Disabilities“, *Proceedings - 2019 1st International Conference on Control Systems, Mathematical Modelling, Automation and Energy Efficiency, SUMMA 2019*, S. 527–531, Nov. 2019, doi: 10.1109/SUMMA48161.2019.8947472.

[11] P. Majaranta und K.-J. R~ih~, „Twenty Years of Eye Typing: Systems and Design Issues“. [Online]. Available: www.prentrom.com/access/wivik.html

[12] D. W. K. Ng, Y. W. Soh, und S. Y. Goh, „Development of an autonomous BCI wheelchair“, *IEEE SSCI 2014 - 2014 IEEE Symposium Series on Computational Intelligence - CIBCI 2014: 2014 IEEE Symposium on Computational Intelligence in Brain Computer Interfaces, Proceedings*, S. 1–4, Jan. 2015, doi: 10.1109/CIBCI.2014.7007784.

[13] „Welcome NextMind! - Snap Newsroom“. https://newsroom.snap.com/de-DE/welcome-nextmind/ (zugegriffen Sep. 25, 2022).

[14] „What is Scrum? | Scrum.org“. https://www.scrum.org/resources/what-is-scrum (zugegriffen Sep. 24, 2022).

[15] „Wikimedia Downloads“. https://dumps.wikimedia.org/ (zugegriffen Sep. 20, 2022).

[16] „GitHub - gambolputty/dewiki-wordrank: Tab-delimited word frequency list compiled from the German Wikipedia“. https://github.com/gambolputty/dewiki-wordrank#2-extract-wikipedia-pages (zugegriffen Sep. 20, 2022).

IX. Abbildungsverzeichnis

X. Anhang

- Rohdaten der Studie:

Testperson 1 (24):		
Iteration I:	Iteration II:	Iteration III:
Geschwindigkeit in s	Geschwindigkeit in s	Geschwindigkeit in s
1243	497	817
6(5) Tippfehler(korrigiert)	1(1) Tippfehler(korrigiert)	3(2) Tippfehler(korrigiert)
----------	----------	----------
Buchstaben pro Minute:	Buchstaben pro Minute:	Buchstaben pro Minute:
3,330651649	8,329979879	5,067319461
----------	----------	----------
Präzision:	Präzision:	Präzision:
91,30%	98,55%	95,65%
83,30%	100,00%	66,70%

Testperson 2 (26):		
Iteration I:	Iteration II:	Iteration III:
Geschwindigkeit in s	Geschwindigkeit in s	Geschwindigkeit in s
1301	625	1012
9(7) Tippfehler(korrigiert)	3(1) Tippfehler(korrigiert)	8(4) Tippfehler(korrigiert)
----------	----------	----------
Buchstaben pro Minute:	Buchstaben pro Minute:	Buchstaben pro Minute:
3,182167563	6,624	4,090909091
----------	----------	----------
Präzision:	Präzision:	Präzision:
86,96%	95,65%	88,41%
77,80%	33,30%	50,00%

Testperson 3 (48):		
Iteration I:	Iteration II:	Iteration III:
Geschwindigkeit in s	Geschwindigkeit in s	Geschwindigkeit in s
1358	656	1548
7(7) Tippfehler(korrigiert)	1(1) Tippfehler(korrigiert)	3(3) Tippfehler(korrigiert)
----------	----------	----------
Buchstaben pro Minute:	Buchstaben pro Minute:	Buchstaben pro Minute:
3,048600884	6,31097561	2,674418605
----------	----------	----------
Präzision:	Präzision:	Präzision:
89,86%	98,55%	95,65%
100,00%	100,00%	100,00%

Testperson 4 (21, FEHLGESCHLAGEN):		
Iteration I:	Iteration II:	Iteration III:
Geschwindigkeit in s	Geschwindigkeit in s	Geschwindigkeit in s
N/A	N/A	N/A
N/A	N/A	N/A
----------	----------	----------
Buchstaben pro Minute:	Buchstaben pro Minute:	Buchstaben pro Minute:
N/A	N/A	N/A
----------	----------	----------
Präzision:	Präzision:	Präzision:
N/A	N/A	N/A
N/A	N/A	N/A

Testperson 5 (34):		
Iteration I:	Iteration II:	Iteration III:
Geschwindigkeit in s	Geschwindigkeit in s	Geschwindigkeit in s
1313	543	1028
8(7) Tippfehler(korrigiert)	2(1) Tippfehler(korrigiert)	7(4) Tippfehler(korrigiert)
----------	----------	----------
Buchstaben pro Minute:	Buchstaben pro Minute:	Buchstaben pro Minute:
3,153084539	7,624309392	4,027237354
----------	----------	----------
Präzision:	Präzision:	Präzision:
88,41%	97,10%	89,86%
87,50%	50,00%	57,10%

Quellcode/Dokumentation/konzeptionelle Mockups:

Iteration I: https://git.imp.fu-berlin.de/glab1996/bachelorarbeit_iteration_i/-/tree/7bc795612614314b92bc71080cf7224c843741b8

Iteration II: https://git.imp.fu-berlin.de/glab1996/bachelorarbeit_iteration_ii/-/tree/b5a0dafc8b9c9f181c880847fadb39e43b3c75fa

Iteration III: https://git.imp.fu-berlin.de/glab1996/bachelorarbeit_iteration_iii/-/tree/66546871575cd973c3acb2aa202c533f4f8a7c20