HAW HAMBURG

BACHELOR THESIS
Fin Ole Engelbart

Text oder Sprache - Eine vergleichende Studie zur Texteingabe in Virtual Reality

FAKULTÄT TECHNIK UND INFORMATIK
Department Informatik

Faculty of Engineering and Computer Science
Department Computer Science

HOCHSCHULE FÜR ANGEWANDTE WISSENSCHAFTEN HAMBURG
Hamburg University of Applied Sciences

Fin Ole Engelbart

Text oder Sprache - Eine vergleichende Studie zur Texteingabe in Virtual Reality

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung
im Studiengang *Bachelor of Science Angewandte Informatik*
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Birgit Wendholt
Zweitgutachter: M.Sc. Iwer Petersen

Eingereicht am: 17. August 2023

Fin Ole Engelbart

Thema der Arbeit

Text oder Sprache - Eine vergleichende Studie zur Texteingabe in Virtual Reality

Stichworte

Virtual Reality, Texteingabe, Spracherkennung

Kurzzusammenfassung

Der Einsatz von Virtual Reality (VR) in den unterschiedlichsten Anwendungsbereichen erfordert zunehmend die Eingabe von Freitexten. Jedoch lassen sich die traditionellen Eingabeverfahren nur bedingt in den virtuellen Kontext übertragen. In dieser Arbeit wird eine Lösung für die Texteingabe in VR entwickelt, welche die effiziente und ergonomische Eingabe über Sprache mit einem Drumkeyboard kombiniert, das besser für die Fehlerkorrektur geeignet ist. Die Lösung wird anschließend einem Benutzertest unterzogen und hinsichtlich verschiedener Kriterien bewertet.

Fin Ole Engelbart

Title of Thesis

Text or speech - A comparative study regarding text entry in virtual reality

Keywords

Virtual Reality, Text Entry, Speech Recognition

Abstract

The use of Virtual Reality (VR) in various application areas increasingly demands text input. However, traditional input methods have limited applicability in the virtual context. This thesis presents a solution for text input in VR that combines efficient and ergonomic voice input with a drum keyboard, which is better suited for error correction. The proposed solution is subsequently subjected to a user test, and its performance is evaluated based on various criteria.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungen

CER Corrected Error Rate.

EKS ER Erroneous Keystroke Error Rate.

ER Error Rate.

FA Funktionale Anforderung.

HMD Head Mounted Display.

KSPC Keystroke per Character.

MSD Minimum String Distance.

MSD ER Minimum String Distance Error Rate.

NCER Not Corrected Error Rate.

NFA Nicht-funktionale Anforderung.

UE4 Unreal Engine 4.

VR Virtual Reality.

WER Word Error Rate.

wpm Wörter pro Minute.

1 Einleitung

Im Rahmen des Projekts *Open Citizen Soundwalks* an der HAW Hamburg soll die Durchführung virtueller Soundwalks untersucht werden. Bei einem Soundwalk erkundet eine Gruppe von Teilnehmern die Akustik einer bestimmten Umgebung, indem gemeinsam eine Route abgelaufen und an mehreren Stationen innegehalten wird, um auf die dortigen Geräusche zu lauschen. An jeder Station füllen die Teilnehmer nach dem Zuhören einen Fragebogen aus, in dem sie ihre Empfindungen festhalten und verschiedene auditive Aspekte der Umgebung bewerten. Dies ist in der Realität eine etablierte Methode, die nun in den digitalen Raum überführt werden soll, denn die Durchführung von Soundwalks in Virtual Reality (VR) würde es ermöglichen, neben bereits existierenden und als Simulation rekonstruierten Orten auch solche zu erkunden, die sich noch in der Planungsphase befinden und rein virtuell vorliegen. Dadurch ließe sich beispielweise ein Windpark in einer beliebigen Landschaft platzieren und die damit einhergehende Geräuschentwicklung schon frühzeitig beurteilen, was die Akzeptanz der Windkraftanlagen fördern könnte, da Anwohner und Betroffene sich mittels einer realistischen Nachbildung einen eigenen Eindruck verschaffen können.

Um den Realismus der virtuellen Soundwalks zu validieren und ihre Anwendung in der Praxis zu legitimieren, sollen die Ergebnisse mit denen traditioneller Soundwalks verglichen werden. Dafür wurde der herkömmliche Papierfragebogen zunächst in eine Mobile App überführt, die in realen Szenarien zum Einsatz kommt und das Ausfüllen und spätere Auswerten der Daten vereinfacht. Diese wurde anschließend in VR nachgebildet, sodass die Fragen für ein möglichst immersives Erlebnis direkt in der virtuellen Realität beantwortet werden können. Die App verwendet verschiedene Fragetypen, von denen sich zwei problemlos in VR umsetzen lassen: Die Angabe eines Wertes auf einer Skala mittels eines Sliders und die Auswahl einer Antwort aus mehreren Optionen über ein Dropdown-Menü. Die übrigen Fragen, welche die Eingabe eines Freitextes erfordern, stellen jedoch eine Herausforderung dar. Während ein Großteil der Menschen heutzutage geübt darin ist, Texte auf dem Smartphone einzutippen oder über eine Tastatur am PC einzugeben,

eignet sich beides nur bedingt für den Einsatz in VR, da es zumindest gelegentlichen Sichtkontakt zur Tastatur und den eigenen Händen erfordert, welcher durch das Head Mounted Display (HMD) blockiert wird. Darüber hinaus werden zur Steuerung oft spezielle Controller in den Händen gehalten, die nicht auf das Eingeben von Text ausgelegt sind. Es bedarf demnach anderer Möglichkeiten der Texteingabe in VR. Das Thema ist nach wie vor Gegenstand der Forschung.

1.1 Ziel

Zur Lösung des beschriebenen Problems sollen die zahlreichen in der Literatur vorgeschlagenen Methoden zur Texteingabe in VR untersucht und hinsichtlich verschiedener relevanter Faktoren miteinander verglichen werden. Neben Geschwindigkeit und Fehleranfälligkeit spielen hier beispielsweise auch Ergonomie und Benutzerfreundlichkeit eine Rolle, insbesondere im Hinblick auf Personen, die wenig bis gar keine Erfahrung mit VR haben. Auf Basis der Recherche sollen die Methoden ausgewählt werden, die sich für das bestehende Projekt am besten eignen, d.h. folgende Kriterien erfüllen:

- Sie sind möglichst effizient, ergonomisch und einsteigertauglich.
- Sie passen zum konkreten Anwendungsfall, was insbesondere bedeutet, dass sie im Stehen durchgeführt werden können und nicht die Immersion beeinträchtigen.
- Sie entsprechen den technischen Rahmenbedingungen, die sich aus Hardware und Software des bestehenden Projekts ergeben.

Diese Methoden sollen im Anschluss implementiert und in Benutzertests miteinander verglichen werden, um das bestgeeignete Verfahren für die Texteingabe in VR zu bestimmen. Darüber hinaus soll die Lösung in das bestehende System integriert werden, sodass der Soundwalk-Fragebogen vollständig in VR beantwortet werden kann.

1.2 Gliederung der Arbeit

In Kapitel 2 wird der aktuelle Stand von Wissenschaft und Technik bezüglich der Texteingabe in VR untersucht. Die in der Literatur vorgeschlagenen Eingabeverfahren werden hinsichtlicher verschiedener Kriterien miteinander verglichen. In Kapitel 3 findet die

Analyse der zu entwickelnden Lösung statt, indem zunächst das Ist-System aus fachlicher Sicht beschrieben und die Auswahl der umzusetzenden Texteingabeverfahren konkretisiert wird. Anschließend werden die genauen funktionalen und nicht-funktionalen Anforderungen formuliert. Kapitel 4 widmet sich dem Entwurf der Lösung und geht dabei auf den technischen Kontext, die Umsetzung der Spracherkennung und das fachliche Design der Implementierung ein. Außerdem wird die zugrundeliegende Softwarearchitektur beschrieben. Kapitel 5 erläutert die Durchführung der Benutzertests, nimmt eine Auswertung der Ergebnisse vor und führt mögliche Kritikpunkte auf. Kapitel 6 fasst die Arbeit zusammen und gibt einen Ausblick.

2 Stand von Wissenschaft und Technik

Verschiedene Texteingabeverfahren eignen sich für verschiedene Einsatzzwecke und die jeweiligen Vor- und Nachteile müssen je nach Anwendungsfall gegeneinander abgewogen werden. In Kapitel 2.1 werden zunächst die wichtigsten Bewertungskriterien vorgestellt und im Anschluss in Kapitel 2.2 konkrete Methoden der Texteingabe in VR untersucht. Kapitel 2.3 fasst die wichtigsten Ergebnisse zusammen.

2.1 Bewertungskriterien

Die vorgestellten Bewertungskriterien sind Eingabegeschwindigkeit (Kapitel 2.1.1), Fehleranfälligkeit (Kapitel 2.1.2), Ergonomie (Kapitel 2.1.3), Mobilität (Kapitel 2.1.4), Immersion (Kapitel 2.1.5), Einsteigertauglichkeit (Kapitel 2.1.6) und Hardwarebedarf (Kapitel 2.1.7).

2.1.1 Eingabegeschwindigkeit

Grundsätzlich gilt selbstverständlich: Je schneller ein Text eingegeben werden kann, desto besser. Die Geschwindigkeit wird meist in Wörtern pro Minute (wpm) gemessen. Anders als der Name vermuten lässt, wird dabei nicht die tatsächliche Anzahl an Wörtern gezählt, sondern die Anzahl der eingegebenen Zeichen durch den festen Wert 5 geteilt, der auf der durchschnittlichen Wortlänge der englischen Sprache beruht, aus Gründen der Vergleichbarkeit aber auch in deutschsprachigen Studien verwendet wird [3, 27]. Auch wenn die Eingaberate auf den ersten Blick der entscheidende Faktor sein mag und zusätzlich den Vorteil hat, dass sie sehr leicht zu erfassen ist, handelt es sich dabei nur um eines von vielen relevanten Kriterien für ein gutes und je nach Anwendungsszenario geeignetes Texteingabeverfahren, wie sich im Folgenden zeigen wird.

2.1.2 Fehleranfälligkeit

Beim Eingeben von Texten können Fehler auftreten. Die meisten Eingabeverfahren bieten deshalb einen oder mehrere Mechanismen zur Fehlerkorrektur. Ist dies der Fall, lassen sich zwei Typen von Fehlern unterscheiden: Solche, die der Nutzer bemerkt und anschließend korrigiert, und solche, die entweder unbemerkt blieben oder die der Nutzer nicht korrigieren wollte und die daher im fertigen Text vorhanden sind. Eine hohe Fehleranfälligkeit wirkt sich sowohl negativ auf die Eingabegeschwindigkeit aus, da das Korrigieren von Fehlern Zeit kostet, als auch auf die Benutzerfreundlichkeit, denn häufig etwas korrigieren zu müssen oder einen fehlerhaften Text zu verfassen kann zu Frustration führen. Im Allgemeinen sollte deshalb eine möglichst niedrige Fehlerrate angestrebt werden, doch gibt es auch hier Trade-offs: Man stelle sich z.B. ein Spracherkennungsprogramm vor, dass 99,99% aller eingesprochenen Wörter korrekt erkennt, dafür aber pro Wort eine Verarbeitungszeit von zehn Sekunden benötigt. Hier wäre trotz erhöhtem Korrekturaufwand ein anderes Programm mit etwas geringerer Erfolgsquote vermutlich vorzuziehen, wenn es im Gegenzug deutlich schneller ist.

Das Messen der Fehlerrate ist komplexer als das der Eingabegeschwindigkeit und es gibt verschiedene Metriken mit unterschiedlichen Vor- und Nachteilen. Die am häufigsten verwendeten werden in [33] und [3] beschrieben:

Die *Error Rate* (ER) misst die Fehlerhaftigkeit des fertig eingegebenen Textes T als den prozentualen Anteil inkorrekter Zeichen in T zur Gesamtlänge von T. Als fast äquivalent betrachtet werden kann die *Minimum String Distance Error Rate* (MSD ER), welche die *Minimum String Distance* (MSD) zwischen T und dem einzugebenden Text P ermittelt und durch das Maximum der Längen beider Texte teilt. Die MSD entspricht der minimalen Anzahl an Operationen aus Einfügen, Löschen und Ersetzen eines Zeichens, die nötig sind, um T in P zu überführen. Wenn die Operationen nicht für einzelne Zeichen sondern auf der Ebene ganzer Wörter betrachtet werden, erhält man die *Word Error Rate* (WER), die standardmäßig zur Bewertung von Spracherkennungssystemen herangezogen wird [24]. Sowohl ER als auch MSD ER und WER haben den Nachteil, dass der Aufwand von Korrekturen, die evtl. während der Eingabe stattgefunden haben, nicht beachtet wird. So können sie beispielsweise bei einem fehlerfreien Text nicht unterscheiden, ob während der Eingabe viele Fehler auftraten, die mühsam korrigiert wurden, oder ob der Text von vornherein korrekt eingegeben wurde - obwohl dies natürlich einen großen Unterschied für die Benutzererfahrung macht.

Zu einem gewissen Grad Aufschluss über den Korrekturaufwand gibt die *Keystroke per Character* (KSPC) Metrik. Sie berechnet, wie viele Tastendrücke durchschnittlich für die Eingabe eines Zeichens nötig sind und liefert damit einen besonders hohen Wert, wenn beispielsweise oft die Backspace-Taste zur Korrektur gedrückt wurde. KSPC ist aber im herkömmlichen Sinne nur bei Verwendung einer Tastatur anwendbar und kann selbst dann nicht sinnvoll für den Vergleich von one-tap und multi-tap Tastaturen herangezogen werden. (Bei multi-tap Tastaturen sind zur Eingabe eines Zeichens ggf. mehrere Tastendrücke nötig, wie es bei alten Handys der Fall war. Sie liefern deshalb unabhängig von der Korrekturzahl einen deutlich höheren KSPC-Wert als one-tap Tastaturen.) Darüber hinaus kann KSPC nicht unterscheiden, ob wenige Fehler auftraten, die durch viele Tastendrücke korrigiert werden mussten, oder viele Fehler, deren Korrektur nur wenige Tastendrücke erforderte. Es besteht eine näherungsweise inverse Beziehung zwischen ER bzw. MSD ER und KSPC (je mehr Fehler korrigiert wurden, desto höher der KSPC-Wert und niedriger der ER/MSD ER-Wert, und umgekehrt), sie lassen sich jedoch nicht auf sinnvolle Weise zu einem einheitlichen Fehlermaß zusammenführen.

Die *Erroneous Keystroke Error Rate* (EKS ER) ist ein weiterer Ansatz und wird berechnet als Quotient aus *IC*, der Anzahl aller inkorrekten Zeichen (korrigiert und nicht korrigiert, inklusive fehlender Zeichen), und der Länge des einzugebenden Textes. Da sie jedoch keine Trennung zwischen korrigierten und nicht korrigierten Fehlern macht, wird sie oft nur verwendet, wenn die Art der Texteingabe das Korrigieren von Fehlern erzwingt und der eingegebene Text dadurch fehlerfrei ist.

Die mächtigste und in vielen neueren Studien zur Texteingabe in VR eingesetzte Fehlermetrik ist die *Total Error Rate* (Total ER). Sie gibt das Verhältnis von *IC* zur Anzahl aller eingegebenen Zeichen an (korrekt und inkorrekt). Neben dem Vorteil, dass sich damit ein gesamtheitliches Bild des Nutzerverhaltens bezüglich fehlerhafter Eingaben zeichnen lässt, kann die Total ER auch problemlos in die *Corrected Error Rate* (CER) für korrigierte Fehler und die *Not Corrected Error Rate* (NCER) für nicht korrigierte Fehler aufgeteilt werden, um beide Typen in einer detaillierteren Analyse getrennt voneinander zu betrachten. Es gilt CER + NCER = Total ER. Alle drei können im Gegensatz zur gerätespezifischen KSPC-Metrik auch für einen geräteübergreifenden Vergleich von Fehlerraten herangezogen werden.

2.1.3 Ergonomie

Eine ergonomische Bedienung ist zentral für alle Arten der Mensch-Computer-Interaktion, so auch für die Texteingabe in VR. Sie zeichnet sich zum einen durch eine intuitive Benutzerschnittstelle aus, die auf unseren Erfahrungen im Umgang mit Objekten der realen Welt beruht und von vielen unserer natürlichen Fähigkeiten Gebrauch macht. Zum anderen ermöglicht sie eine Interaktion mit geringem kognitiven und motorischen Aufwand, um sowohl geistige als auch körperliche Erschöpfung zu vermeiden. Dies ist vor allem dann wichtig, wenn längere Texte eingegeben werden sollen. Speziell für VR ergeben sich weitere Anforderungen an die Ergonomie, da die Bildschirme des HMD das gesamte Sichtfeld einnehmen und somit kein visueller Bezug zur realen Umgebung mehr besteht. Phänomene wie Cybersickness (Kopfschmerzen, Übelkeit, Schwindel usw.) können u.a. durch Diskrepanzen verschiedener Sinneseindrücke entstehen, wenn beispielsweise die Augen eine Bewegung des Körpers wahrnehmen, die der Gleichgewichtssinn nicht verspürt. Auch Probleme in der Raum- und Tiefenwahrnehmung können auftreten, falls nicht hinreichend visuelle Hilfestellungen gegeben sind. Diese und weitere Effekte sollten bei der Entwicklung der Texteingabesysteme beachtet werden, um eine gute Ergonomie zu gewährleisten. Ob und zu welchem Grad dies erreicht wird, kann z.B. mithilfe von Interviews oder Fragebögen ermittelt werden [11].

2.1.4 Mobilität

Insbesondere für VR-Erlebnisse, bei denen sich der Benutzer frei in einem Trackingbereich bewegen kann, ist es wichtig, dass die Mobilität durch die Art der Texteingabe nicht eingeschränkt wird. Manche Verfahren erfordern eine sitzende Position oder benötigen eine feste Unterlage, um beispielsweise eine PC-Tastatur zu bedienen. Dies ist für Szenarien, in denen auf natürliche Weise eine Umgebung erkundet werden soll, weniger tauglich. Findet das VR-Erlebnis hingegen ohnehin nur stationär im Stehen oder im Sitzen statt, ist dieser Aspekt kaum bis gar nicht relevant.

Im weiteren Verlauf der Arbeit wird ein Eingabeverfahren als *mobil* oder *für die mobile Anwendung geeignet* bezeichnet, wenn es die Bewegungsfreiheit des Nutzers nicht einschränkt.

2.1.5 Immersion

Für viele VR-Erlebnisse, beispielsweise beim Gaming oder in realistischen Simulationen, ist die Immersion ein zentraler Aspekt und sollte durch die Art der Texteingabe möglichst wenig beeinträchtigt werden. So würde das Einblenden von Kamerabildern immer wieder den Bezug zur realen Umgebung herstellen und das Eintauchen in die virtuelle Welt erschweren. Es gibt jedoch auch Anwendungsfälle, bei denen dies nur eine untergeordnete Rolle spielt, z.B. durch Virtual Reality unterstützte Büroarbeitsplätze oder Kollaborationsplattformen.

2.1.6 Einsteigertauglichkeit

Manche Art der Texteingabe erfordert ein längeres, zum Teil mental anstrengendes Training, bevor sie effizient eingesetzt werden kann. Dies ist vor allem der Fall, wenn es sich um eine neue, speziell für VR entwickelte Methode handelt, wie z.B. das Formen von Gesten mit den Fingern, die zunächst auswendiggelernt und eingeübt werden müssen. Hier kann es von Vorteil sein, auf bereits bestehendes Wissen zurückzugreifen, indem im Alltag verbreitete Bedienkonzepte wie das Tippen auf der Tastatur oder das Schreiben mit einem Stift auf Papier in den virtuellen Kontext übertragen werden, wenn auch eventuell in abgewandelter Form.

2.1.7 Hardwarebedarf

Benötigt ein Texteingabeverfahren teure oder sehr spezielle Hardware oder ein komplexes Setup, schränkt dies die Breite der Einsatzmöglichkeiten ein. (Beispiel: Präzises Tracking der einzelnen Finger durch eine Vielzahl an Motion-Capture-Sensoren). Wenn die erforderliche Technik aber ohnehin durch den konkreten Anwendungsfall gegeben ist, kann es sich durchaus lohnen, diese auch zu nutzen, um besonders gute Resultate zu erzielen.

2.2 Vorstellung konkreter Methoden und ihrer Ergebnisse

Nach mehreren Jahrzehnten der Forschung gibt es ein breites Spektrum an vorgeschlagenen Methoden zur Texteingabe in VR. Vor allem in den letzten Jahren sind einige neuartige und kreative Ideen hinzugekommen, die zum Teil auch durch den stetigen

technischen Fortschritt erst ermöglicht wurden. Einige dieser Verfahren werden in diesem Kapitel vorgestellt und hinsichtlich der soeben beschriebenen Bewertungskriterien diskutiert. Kapitel 2.2.1 behandelt physische und Kapitel 2.2.2 virtuelle Tastaturen. In Kapitel 2.2.3 wird die Eingabe mittels Handschrift betrachtet. Kapitel 2.2.4 untersucht die Spracheingabe.

2.2.1 Physische Tastaturen

Zu den physischen Tastaturen zählen neben der Standard-PC-Tastatur auch andere tastenbasierte Eingabegeräte sowie Smartphones und Tablets. Der folgende Abschnitt behandelt die verschiedenen Typen.

PC-Tastatur

Der allgegenwärtige Gebrauch von PC-Tastaturen im Alltag vieler Menschen macht sie zu offensichtlichen Kandidaten für die Texteingabe in VR. Ihre Adaption birgt jedoch die Schwierigkeit, dass sowohl Anfänger als auch erfahrene Nutzer Sichtkontakt zur Tastatur und den eigenen Händen benötigen, um die korrekte Position der Finger sicherzustellen. Je ungeübter eine Person, desto häufiger wird zur Verifikation des nächsten Tastendrucks ein Blick auf die Tastatur geworfen [4, 31]. Dies ist aufgrund des getragenen HMDs nicht ohne Weiteres möglich (Abb. 2.1). Um das Problem zu adressieren, setzen einige Anwendungen externe Tiefenkameras zum Tracken von Tastatur und Händen ein. Dadurch können ihre virtuellen Repräsentationen in VR dargestellt werden (Abb. 2.2) [12, 7, 16]. In einer Studie [7] konnten Teilnehmer mit dieser Methode eine Eingaberate von 34 wpm erzielen, was sich mit etwas Übung auf 44 wpm steigern ließ. Vergleichbare Ergebnisse liefert [16] mit 31 wpm ohne vorheriges Training, was 71% der normalen Tippgeschwindigkeit der Testpersonen außerhalb von VR entsprach. Die Error Rate stieg von 6,58 auf 13,43 im Vergleich zur Baseline. Damit waren die Ergebnisse entgegen den Erwartungen etwas schlechter als ohne eine virtuelle Darstellung der Hände (34 wpm und 12% ER). Probleme beim Tracking, wodurch die Hände nur zu durchschnittlich 70% der Zeit angezeigt werden konnten, haben möglicherweise dazu beigetragen. Andere Untersuchungen bestätigen jedoch diesen Fund und ermittelten ähnliche Daten, wenn nur die aktuellen Positionen der Fingerspitzen auf der Tastatur in VR angezeigt werden [15]. Ein weiterer Ansatz ist das Einblenden eines live Videostreams von Tastatur und Händen, der einen Ausschnitt der virtuellen Welt überdeckt (Abb. 2.2). Damit sind etwas höhere

Eingabegeschwindigkeiten möglich (ca. 39 wpm) und die Fehlerrate ist deutlich niedriger. Allerdings wird die Immersion erheblich gestört und es sind zusätzliche Kameras erforderlich [15, 12].

Abbildung 2.1: Nutzung einer PC-Tastatur in VR [12].

Abbildung 2.2: Benutzersicht bei der Eingabe über eine PC-Tastatur mit virtueller Repräsentation der Hände (links) und eingeblendetem Kamerabild (rechts) [15].

Neben dem Problem der blockierten Sicht bedarf es auch einer festen Unterlage für die Tastatur sowie zur bequemen Bedienung einer Stütze für die Handgelenke. Dies schränkt die Mobilität ein und erschwert den Einsatz in VR-Anwendungen, bei denen sich die Nutzer frei umherbewegen können. Die Flexibilität kann durch kabellose Tastaturen in gewissem Rahmen erhöht werden, da sie umhergetragen und so an verschiedenen Orten

eingesetzt werden können, aber die Grundproblematik bleibt. Als Alternative wurde die Möglichkeit einer am Körper befestigten Unterlage für die Tastatur untersucht, welche an Gurten um Nacken und Hüfte getragen wird, genannt *HawKEY* (Abb. 2.3) [26]. Dies ermöglicht einem Nutzer, umherzugehen und dabei jederzeit die Tastatur bedienen zu können. Auch hier wurde die Performance für verschiedene Darstellungen von Händen und Tastatur miteinander verglichen. Das Einblenden von Kamerabildern erzielte erneut die besten Ergebnisse mit 32 wpm und einer Total ER von etwas über 10%. Es besteht somit nur ein geringer Performanceverlust gegenüber der stationären Anwendung. Das Tragen des Gestells wurde als weitgehend komfortabel eingestuft, wenn auch die Gurte mit der Zeit etwas unbequem würden. Eine vollständig natürliche Bewegung ist aufgrund des zusätzlichen Gewichts und der Notwendigkeit, durchgehend aufrecht zu stehen, jedoch auch mit dieser Methode nicht möglich. Da es sich dabei um eine Spezialanfertigung handelt, sind die Einsatzmöglichkeiten ohnehin beschränkt.

Abbildung 2.3: HawKEY [26].

Das exakte haptische Feedback von physischen PC-Tastaturen verbessert sowohl Performance als auch Benutzerfreundlichkeit im Vergleich zu anderen Verfahren und erhöht zudem die Ergonomie [16]. Die mentale Belastung ist in VR ein wenig höher als in der Desktopanwendung, aber in den Punkten Komfort und Erschöpfung wurde diese Art der Texteingabe insgesamt positiv bewertet. Sie erwies sich zudem als einsteigerfreundlich, vorausgesetzt es besteht bereits Erfahrung im herkömmlichen Umgang mit der Tastatur,

denn die erworbenen Fähigkeiten scheinen sich in relativ hohem Maße auf den Einsatz in VR übertragen zu lassen.

Mobile Tastaturen

Um die Mobilitätseinschränkungen physischer PC-Tastaturen zu umgehen, wurde der Einsatz mobiler Tastaturen vorgeschlagen. Ein bekanntes Beispiel ist Twiddler, das einer kleinen Fernbedienung mit zwölf Tasten gleicht (Abb. 2.4) [22]. Es wird in einer Hand gehalten und Zeichen werden durch das Drücken einer oder mehrerer Tasten mit Zeige-, Mittel-, Ring- und kleinem Finger eingegeben. Damit konnten in einer Benutzerstudie, in der ein virtuelles Abbild der Tastatur in VR dargestellt wurde, 3 wpm und eine Genauigkeit von 82% erzielt werden. Das Auswendiglernen der Tastenkombinationen erforderte intensives Training. Etwas bessere Ergebnisse lieferte der Test einer mobilen multi-tap Tastatur, bei der über jede der neun Tasten mehrere Zeichen eingegeben werden können, indem die zugehörige Taste ein- oder mehrmals gedrückt wird. Um beispielsweise ein „b“ einzugeben, muss die Taste für „a“, „b“ und „c“ zweimal gedrückt werden. Damit kam man auf 12,1 wpm und 95% Genauigkeit. Der Gewinn an Mobilität geht demnach in beiden Fällen mit einer deutlich schlechteren Performance sowie einer geringen Einsteigertauglichkeit einher. Diese Art der Texteingabe ist somit höchstens für das Eingeben kurzer Phrasen wie Passwörtern oder Suchbegriffen geeignet. Darüber hinaus können die speziellen Geräte ausschließlich als Tastatur verwendet werden. Erfolgt die übrige Interaktion beispielsweise über Controller, muss zwischen beiden umständlich hin- und hergewechselt werden [12].

Abbildung 2.4: Twiddler [22].

Smartphone- und Tablet-Tastaturen

Ebenfalls im Alltag sehr verbreitet ist das Tippen auf einer Smartphone-Tastatur. Bei der Übertragung dieser Methode in VR wiegt das Problem des fehlenden Blickkontakts noch um Einiges schwerer als bei PC-Tastaturen, da der flache Touchscreen keinerlei haptische Orientierung bietet. HoVR-Type [19] machte deshalb von der Hover-Funktionalität moderner Smartphones Gebrauch, um die aktuelle Position der Finger über dem Touchscreen zu ermitteln und auf dem in VR dargestellten virtuellen Abbild anzuzeigen (Abb. 2.5). Das tatsächliche Smartphone war für die Benutzer nicht sichtbar. Die Auswahl einer Taste erfolgte entweder durch Antippen oder Loslassen des Touchscreens. In einer Studie lieferte die erste Variante 7,8 wpm und eine Genauigkeit von 79,5%. Die Nutzer wirkten gestresst beim ständigen Versuch den Touchscreen nicht versehentlich zu berühren und ein falsches Zeichen einzugeben, das anschließend korrigiert werden musste. Bei der zweiten Variante entfiel diese mentale Belastung, da sich die Auswahl noch anpassen ließ, indem der Finger vor dem Loslassen zur richtigen Taste bewegt wurde. Entsprechend besser war die Performance mit 9,0 wpm und 92,6% Genauigkeit.

Abbildung 2.5: HoVR-Type [19].

Eine andere Studie [10] untersuchte die Texteingabe auf einem Grafiktablet mittels eines Stylus (Abb. 2.6). Die Eingabe fand im Sitzen statt und das Tablet lag vor den Nutzern auf einem Tisch. Auch hier sahen Benutzer nur ein virtuelles Abbild des Tablets, auf dem eine Tastatur dargestellt wurde. Die über die eingebauten Sensoren bestimmte Position des Stylus wurde als Cursor auf der Tastatur angezeigt und die Eingabe eines Zeichens erfolgte durch Antippen der entsprechenden Taste. 12,79 wpm bei 6,4% MSD ER konnten so erzielt werden. Benutzerfreundlichkeit und Ergonomie wurden eher schlecht bewertet: Um auf das Tablet hinabzusehen, muss der Kopf für längere Zeit nach unten gebeugt sein, wobei der Nacken durch das Gewicht des HMDs zusätzlich belastet wird. Hinzu kamen

Probleme beim Tracking des Stylus, was zu gelegentlichem Verschwinden des Cursors führte und die korrekte Eingabe erschwerte. Da ein Tisch als Unterlage für das Tablet benötigt wird, ist die Mobilität bei dieser Methode ebenfalls eingeschränkt.

Abbildung 2.6: Eingabe mit Tablet und Stylus [10].

2.2.2 Virtuelle Tastaturen

Bei virtuellen Tastaturen handelt es sich um Tastaturen, die allein in VR existieren und kein physisches Gegenstück besitzen. Oft benutzen sie das Layout der Standard-PC-Tastatur, um auf bekanntem Wissen aufzubauen. Es finden sich aber auch andere Modelle, da die Hardwareunabhängigkeit große Freiheiten im Design mit sich bringt. Die verschiedenen Techniken und zahlreichen Interaktionsmethoden werden im Folgenden betrachtet. Wenn nicht anders erwähnt, wurde das Standardlayout der PC-Tastatur verwendet.

Head-Pointing

Beim Head-Pointing wird ausgehend vom Gesicht des Benutzers ein Strahl in die VR-Welt projiziert, der seiner Blickrichtung folgt und auf eine vor ihm schwebende virtuelle Tastatur trifft (Abb. 2.7). Um ein Zeichen einzugeben, wird durch Bewegen des Kopfes die Position des Strahls auf der Tastatur verändert, sodass die korrekte Taste anvisiert wird; anschließend wird diese durch kurzzeitiges Verweilen, Drücken eines Controllerbuttons oder auf andere Weise ausgewählt [12]. In [38] erfolgte die Auswahl einer Taste durch eine Verweildauer von 400 ms. Damit war eine Eingaberate von 10,6 wpm sowie eine Genauigkeit von 95,8% möglich. Wurde die Auswahl stattdessen durch Drücken eines Controllerbuttons getroffen, konnte dies auf 15,6 wpm und eine 98% Trefferrate erhöht werden. [34] liefert sehr ähnliche Resultate für die controllerbasierte Auswahl (15,4 wpm und 1% Error Rate), womit diese für Head-Pointing als performanter eingestuft werden

kann. Allerdings wird ein zusätzlicher Controller benötigt und es bleibt zu untersuchen, ob eine pro Benutzer konfigurierbare Verweildauer die Geschwindigkeit für jene Technik erhöhen könnte. Eine etwas schlechtere Performance (11,2 bis 13,5 wpm) ermittelte [21]. Dort wurde die anvisierte Taste durch Blinzeln oder eine kurze Vorwärtsbewegung des Kopfes ausgewählt. Hinsichtlich der Benutzerfreundlichkeit und Ergonomie gab es keine nennenswerten Unterschiede in der Bewertung der verschiedenen Auswahlmethoden. Sie wurden alle als einfach zu erlernen und die Erschöpfung als akzeptabel eingestuft. Potenziell können die häufigen Kopfbewegungen jedoch zu Unwohlsein führen [12].

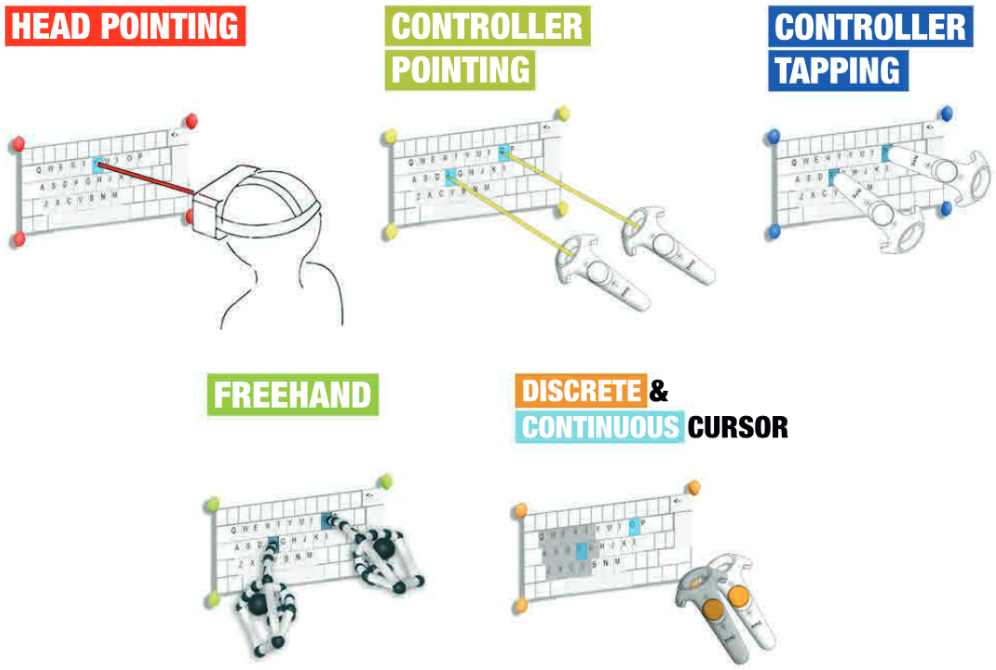


Abbildung 2.7: Verschiedene Bedienarten einer virtuellen Tastatur [34].

Controller-Pointing

Auch beim Controller-Pointing wird ein Strahl in die virtuelle Welt geworfen, der anders als beim Head-Pointing aber vom Controller ausgeht, sodass die Tasten der virtuellen Tastatur wie mit einem Laserpointer anvisiert werden können (Abb. 2.7). Zur Eingabe des Zeichens wird ein Button des Controllers gedrückt. Sowohl [34] als auch [5] verwendeten beide Controller für zweihändigen Input. In [34] lag die Geschwindigkeit bei 15,4 wpm und die Error Rate bei 1%. [5] berichtet von 16,7 wpm und einer Total ER von 11,5%.

Die Performance ist damit ähnlich zu der des Head-Pointing. Controller-Pointing wurde jedoch als noch benutzerfreundlicher und weniger körperlich anstrengend bewertet. [37] vergleicht ein flaches (2D) und ein gekrümmtes (3D) Tastaturlayout und kommt zu dem Schluss, dass das flache Layout, welches auch in [34] und [5] verwendet wurde, eine leicht höhere Eingabegeschwindigkeit und niedrigere Fehlerrate mit sich bringt.

Tapping

Wenn Tracking von Händen oder Controllern eingesetzt wird, kann die Texteingabe durch Tapping erfolgen, indem die virtuellen Tasten mit der Controllerspitze oder einem Finger angetippt werden (Abb. 2.7) [34, 18, 2, 1]. Dies ermöglicht eine sehr präzise Eingabe und dementsprechend niedrige ER im Bereich von 1%, ist durch das häufige Ausstrecken der Arme aber auch körperlich anspruchsvoll. Die Geschwindigkeit liegt je nach Studie bei ca. 10 bis 15 wpm. Laut [34] ist das Controller-Tapping etwas schneller und genauer, was möglicherweise auf eine höhere Tracking-Genauigkeit zurückzuführen ist. Das Tapping per Finger wurde deshalb auch als weniger benutzerfreundlich eingestuft.

Drumkeyboard

Drumkeyboards machen von einer Schlagzeug-Metapher Gebrauch. Beide Controller werden wie Drumsticks verwendet und die Tasten fungieren als Trommeln, die durch Abwärtsbewegungen in Handgelenken und Unterarmen angeschlagen werden (Abb. 2.8). Damit sind signifikant bessere Eingaberaten als mit den Pointing-Methoden möglich: 24,6 wpm wurden in [6] gemessen und 21 wpm in [5]. Die Total ER betrug 7% bzw. 12% und war damit etwas höher. Interviews mit den Testpersonen deuten auf eine benutzerfreundliche und unterhaltsame Bedienung sowie eine geringe körperliche und moderate mentale Belastung hin. Eine spätere Studie [32] erweiterte die Technik zur sog. FLIK-Methode, bei der die Tasten auch von der Unterseite angeschlagen werden können, wodurch die auf eine Abwärtsbewegung folgende Aufwärtsbewegung ebenfalls zur Eingabe eines Zeichens genutzt werden kann. Somit wird die Anzahl benötigter Bewegungen halbiert. Die erzielten 23 wpm lassen aber auf keinen nennenswerten Performancegewinn schließen.

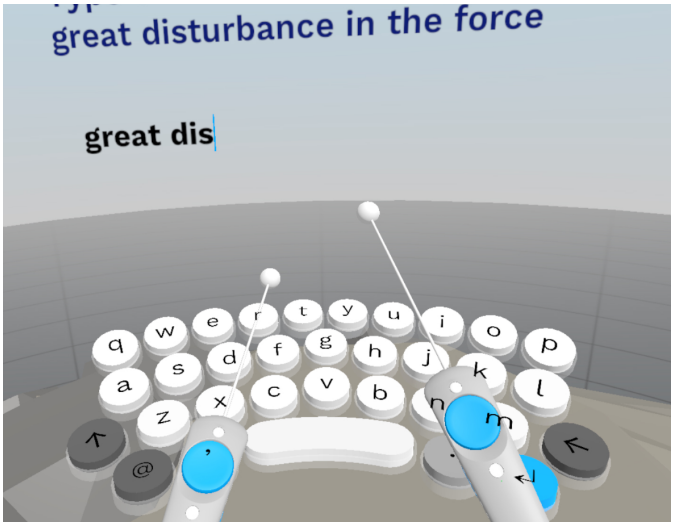


Abbildung 2.8: Drumkeyboard [6].

Cursor

Weit verbreitet auf Spielekonsolen ist die Texteingabe über einen Cursor, der mithilfe der Richtungstasten oder Analog-Sticks über die Tastatur bewegt wird. Dies lässt sich auch mit VR-Controllern umsetzen, die entsprechende Interaktionsmöglichkeiten bieten (Abb. 2.7). [5] und [34] verwendeten die Trackpads der *HTC Vive*-Controller, welche mit dem Daumen bedient werden und sowohl touchbasierten zweidimensionalen Input ermöglichen, um die Position des Cursors zu bestimmen, als auch zur Eingabe eines Zeichens wie ein Button gedrückt werden können. Die virtuelle Tastatur wurde in einen linken und rechten Teil aufgetrennt (Split-Keyboard), die beide einen eigenen Cursor besitzen, welcher mit der jeweiligen Hand gesteuert wird. In [5] ließ sich der Cursor kontinuierlich bewegen und wählte stets die Taste aus, über der er sich gerade befand. [34] untersuchte zusätzlich einen diskreten Cursor. Beide Varianten schneiden in den Punkten Eingabegeschwindigkeit und Benutzerfreundlichkeit schlechter ab als alle bisher vorgestellten Bedienarten virtueller Tastaturen. Die physische Belastung hingegen ist hier am geringsten und die Fehleranfälligkeit ebenfalls niedrig.

Gaze-Typing

Viele neuere VR-Headsets unterstützen Eyetracking und ermöglichen damit die Steuerung über Augenbewegungen [12]. Dieses sog. Gaze-Typing wurde in einer Studie für

flache und gewölbte virtuelle Tastaturen untersucht [29]. Zur Eingabe fokussierten Benutzer ihren Blick auf eine Taste und verweilten dort für 550 ms oder drückten einen Controllerbutton. Sie erreichten damit zwischen 8 und 10 wpm. Die Genauigkeit lag stets bei über 99% und es wurden keine signifikanten Unterschiede zwischen flacher und gewölbter Tastatur festgestellt. Ein anderer Ansatz [23] kombinierte Eyetracking mit einem Brain-Computer-Interface zur Bestimmung der Blickrichtung, indem über ein EEG elektrische Impulse im Gehirn aufgezeichnet wurden. Die Tasten waren dabei alphabetisch sortiert in einem 5×8 großen Raster angeordnet. Eine informelle Evaluation ergab ebenfalls eine relativ geringe Eingaberate von 10 wpm. Ungeachtet dessen könnte diese Methode für Nutzer mit körperlichen Einschränkungen sinnvoll sein [12].

Wortgesten

Die bisherigen Methoden basieren auf der Eingabe einzelner Zeichen. Um ein Wort einzugeben, werden nacheinander die entsprechenden Tasten ausgewählt. Bei Wortgesten-Tastaturen hingegen können ganze Wörter am Stück eingegeben werden, indem ein Pfad durch die entsprechenden Tasten gezogen wird (Abb. 2.9). Ein Beispiel für ihre Anwendung in VR ist Vulture [25], das Hand- bzw. Fingertracking einsetzt. Durch Bewegen der Hand platziert der Benutzer zunächst einen Cursor auf der Taste, bei der der Pfad beginnen soll. Dann führt er mit Daumen und Zeigefinger eine Kneifgeste aus und bewegt die Hand von einer Taste zur nächsten, womit er eine Linie zeichnet, die durch die Zeichen des Wortes verläuft. Durch Öffnen der Kneifgeste wird das Zeichnen beendet. Der entstandene Pfad wird anschließend dekodiert und über der Tastatur werden die fünf wahrscheinlichsten Wörter angezeigt, von denen durch Antippen eines ausgewählt werden kann. Eine Auswertung dieser Methode ergab 20,6 wpm. Wie alle Wortgestenverfahren hängt ihre Effizienz stark von der Genauigkeit des Dekodierers ab. Darüber hinaus ist Vulture auf sehr präzises Tracking angewiesen.

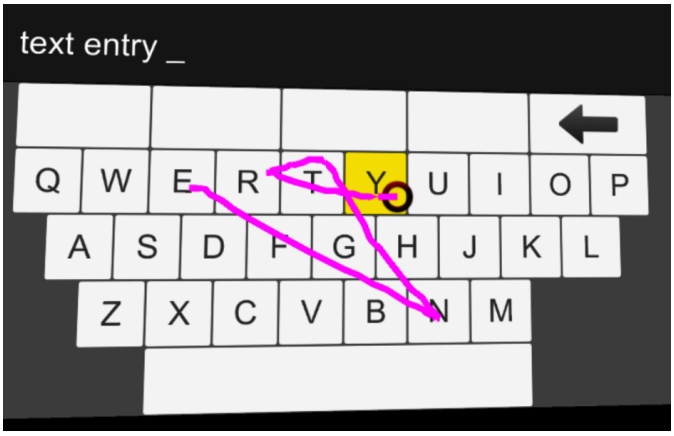


Abbildung 2.9: Eingabe einer Wortgeste [38].

[38] untersuchte das Zeichnen von Wortgesten durch Kopfbewegungen, analog zum Head-Pointing. Start und Ende eines Wortes wurden durch Drücken und Loslassen eines Controllerbuttons signalisiert. Benutzer erreichten damit 24,7 wpm bei geringer Fehlerrate. Wie erwartet, konnte die Performance im Vergleich zum Head-Pointing also stark verbessert werden, da nicht jede Taste einzeln bestätigt werden muss. Die hohe körperliche Belastung durch die vielen Kopfbewegungen ist jedoch nach wie vor vorhanden.

Zwei weitere Verfahren werden in [8] vorgestellt: Das Zeichnen von Wortgesten unter Verwendung der Controller als virtuelle Laserpointer (siehe Controller-Pointing) und mittels eines Smartphone-Touchscreens. Die erzielte Geschwindigkeit war mit 16,4 bzw. 9,6 wpm aber deutlich geringer und die controllerbasierte Variante scheint keinen Vorteil gegenüber dem Controller-Pointing zu bieten.

Handbasierte Methoden

Im Zuge immer robusterer Bilderkennungstechnologien wurden Texteingabeverfahren vorgeschlagen, die speziell für Freihand-Szenarien geeignet sind. Bei BlueTap [9] berühren Nutzer mit dem Daumen bestimmte Stellen der übrigen Finger, auf die jeweils ein Zeichen abgebildet wird. Erkannt werden die Gesten über eine am Handgelenk getragene Kamera. DigiTap [28] verwendet dieselbe Technik in der Multi-tap-Variante (Abb. 2.10). Hier stehen die Knöchel und Fingerspitzen für jeweils ein bis vier Zeichen, zwischen denen durch mehrfaches Berühren gewechselt werden kann, was eine durch natürliche Haptik

unterstützte Eingabe mit 10 wpm erlaubt. Die effiziente Nutzung dieser Verfahren ist mit einem nicht unwesentlichen Lernaufwand verbunden.

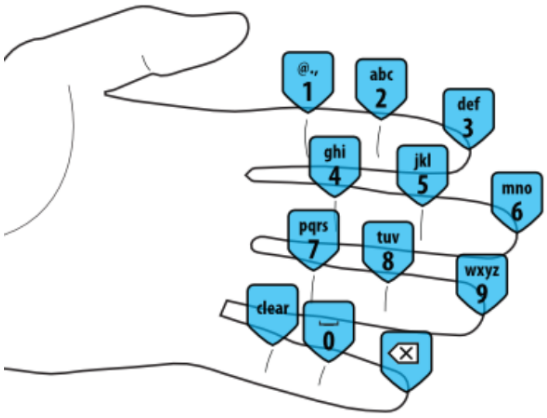


Abbildung 2.10: DigiTap [28].

Auch bei PinchType [14] wird zur Eingabe eines Zeichens der Daumen mit einem anderen Finger derselben Hand zusammengeführt. Im Unterschied zu BlueTap und DigiTap kommen beide Hände zum Einsatz, sodass acht verschiedene Gesten ausgeführt werden können. Jede davon steht für einen farblich hervorgehobenen Bereich auf einer virtuellen Tastatur. Um auf dem vorhandenen Muskelgedächtnis aufzubauen, sind einem Finger dabei genau die Tasten zugeordnet, die er standardmäßig auf der Tastatur tippen würde. Zusätzlich haben die Fingerspitzen der dargestellten Handrepräsentationen dieselbe Farbe wie die zugehörigen Tasten. Da jede Geste für mehrere Zeichen steht und deshalb uneindeutig ist, wird ein Algorithmus verwendet, der basierend auf der bisherigen Eingabe die möglichen Wörter ermittelt und nach ihrer Häufigkeit in der jeweiligen Sprache sortiert. Neben der Tastatur wird eine Liste der wahrscheinlichsten Kandidaten angezeigt, aus der das gewünschte Wort ausgewählt werden kann. Die genauen Details werden in [14] erläutert. Teilnehmer der Studie konnten mit dieser Methode 12,5 wpm erzielen, wobei fast die Hälfte der Zeit zur Fehlerkorrektur aufgewandt wurde. Die Bedienung wurde im Durchschnitt als mäßig komfortabel empfunden.

Kreisförmige Layouts

Einige vorgeschlagene Eingabeverfahren, die von einer virtuellen Nachbildung der Standard-PC-Tastatur abweichen, verwenden ein kreisförmiges Layout. HiPad [17] eignet sich für Controller mit rundem Touchpad oder einem vergleichbaren Bedienelement und ist auf

die einhändige Texteingabe ausgelegt. Die Buchstaben sind ringförmig in alphabetischer Reihenfolge angeordnet und in sechs Bereiche zu je vier oder fünf Zeichen unterteilt (Abb. 2.11). Zur Eingabe eines Wortes werden nacheinander die Bereiche ausgewählt, die die gewünschten Buchstaben beinhalten. Wie schon bei DigiTap [28] gibt es somit für jede Eingabefolge immer mehrere mögliche Wörter, die algorithmisch ermittelt werden. Die wahrscheinlichsten Optionen werden in einer Liste angezeigt, die durch Drücken eines Buttons oder Triggers in den Fokus des Inputs versetzt werden kann. Daraufhin kann über das Touchpad zum gewünschten Wort navigiert und dieses durch erneutes Drücken ausgewählt werden. Mechanismen zur Eingabe einzelner Buchstaben oder Leerzeichen sowie zum Löschen von Zeichen werden in [17] beschrieben. Anfänger konnten mit dieser Methode 13,6 wpm erzielen (4,4% Total ER) und gaben eine gute Bewertung der Nutzerfreundlichkeit ab.

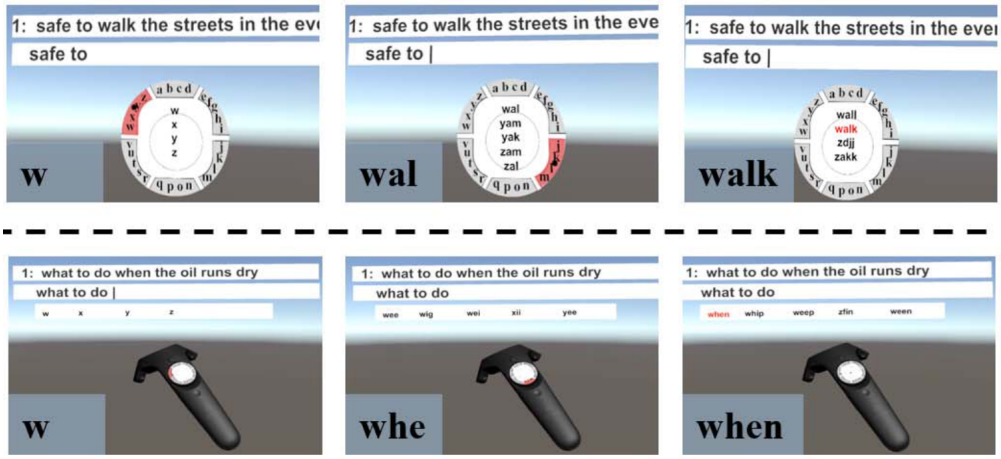


Abbildung 2.11: HiPad [17].

Auch bei PizzaText [39] erfolgt eine Aufteilung des Alphabets, in dem Fall in sieben Regionen zu je vier Zeichen (26 Buchstaben plus Leerzeichen und Backspace), deren Anordnung an die Stücke einer Pizza erinnert (Abb. 2.12). Innerhalb einer Region sind die Zeichen wiederum an festen Positionen angeordnet (oben, unten, links, rechts). Ursprünglich zur Steuerung über die beiden Joysticks eines Gamecontrollers konzipiert, ließe sich PizzaText auch mit zwei VR-Controllern umsetzen, die Joysticks, Trackpads oder Ähnliches besitzen. Einer der Joysticks dient zur Selektion der Region und mit dem anderen wird im Anschluss das gewünschte Zeichen innerhalb der Region ausgewählt. Eine anfängliche Eingabegeschwindigkeit von 8,6 wpm ließ sich nach einem zweistündigen Training beinahe verdoppeln.

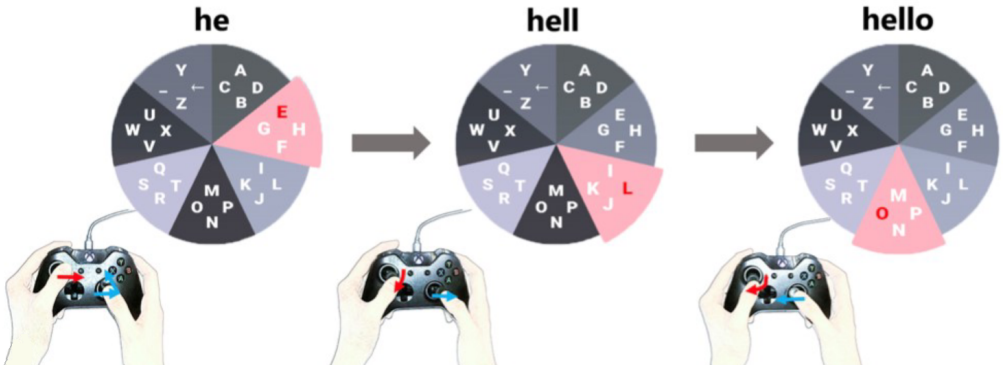


Abbildung 2.12: PizzaText [39].

Flower Text Entry [20] teilt die QUERTY-Tastatur in sechs Reihen nebeneinanderliegender Buchstaben auf, die sternförmig angeordnet werden (Abb. 2.13). In der Mitte befindet sich die Leertaste. Der Aufbau wurde so gewählt, dass möglichst viel Erfahrung mit der vertrauten QUERTY-Tastatur auf das neue System übertragen werden kann. Die Auswahl einer Taste erfolgt, indem eine Bewegung des Controllers im Raum auf die zweidimensionale virtuelle Tastatur projiziert wird. Ausgehend vom Mittelpunkt wird der Controller in eine bestimmte Richtung radial nach außen bewegt, um sowohl die Buchstabenreihe als auch die Position innerhalb der Reihe zu bestimmen. Die genaue Berechnung kann in [20] nachgelesen werden. Wenn die richtige Taste erreicht wurde, wird zur Bestätigung ein Button gedrückt. Wurde der Auswahlprozess zusätzlich durch auditives und haptisches Feedback unterstützt, erreichten Benutzer 12 wpm bei einer Total ER von 6,5%.

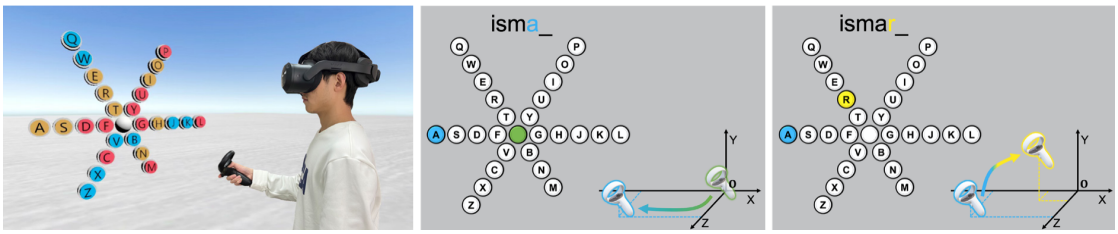


Abbildung 2.13: Flower Text Entry [20].

3D Layouts

VR-Erlebnisse finden in einer dreidimensionalen Umgebung statt, dennoch handelt es sich bei allen bisher vorgestellten Methoden um 2D Layouts. Das Cubic Keyboard [36]

verwendet stattdessen eine $3 \times 3 \times 3$ Matrix aus würfelförmigen Tasten (Abb. 2.14). Zur Eingabe eines Zeichens wird der Controller zu der entsprechenden Taste bewegt und dort kurz gehalten. In einer vorläufigen Studie ließ sich Text so mit 22 wpm eingeben. Weitere Bewertungskriterien wurden noch nicht untersucht.

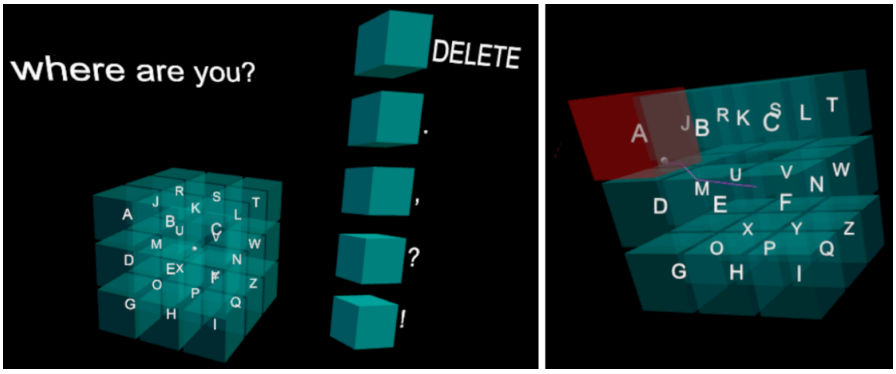


Abbildung 2.14: 3D-Tastatur [36].

2.2.3 Handschrift

Vielleicht noch vertrauter als das Tippen auf PC- und Smartphonetastatur ist uns das Schreiben mit einem Stift, sei es altmodisch auf Papier oder digital auf einem Grafiktablet. Trotzdem gibt es nur wenige Studien, die Handschrift als Texteingabeverfahren in einem modernen VR-Setup untersuchen [35]. Experimente zum Schreiben mit Stylus und Tablet ergaben, dass die Performance in VR deutlich schlechter ist als außerhalb [35]. Eine andere Möglichkeit ist die *on-air* Handschrift, bei der die Schreibunterlage rein virtuell existiert, d.h. man schreibt „in der Luft“. Motiviert wird dieser Ansatz u.a. durch die Unabhängigkeit von spezieller, teurer Hardware wie Grafiktablets; stattdessen werden die weit verbreiteten VR-Controller oder Handtracking eingesetzt [35], was auch eine mobile Anwendung zulässt.

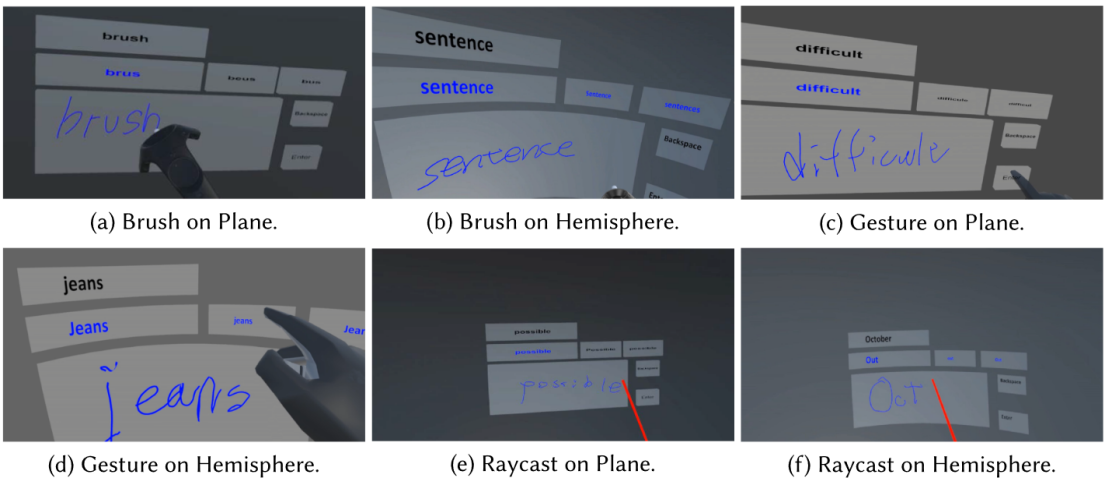
In [13] wird zur on-air Handschrift der Controller als Laserpointer verwendet und auf eine etwas entfernte virtuelle Leinwand gerichtet (Abb. 2.15). Dort, wo der Strahl auf die Leinwand trifft, ändert sich ihre Farbe, vorausgesetzt der designierte Button wird gedrückt gehalten. So lässt sich eine beliebige Linie zeichnen. Um ein Zeichen zu schreiben, das aus mehreren Linien besteht (z.B. ein „A“), muss der Button nach dem Loslassen innerhalb von 0,6 Sekunden erneut gedrückt werden. Ansonsten wird die Eingabe als fertig angesehen und die Zeichenerkennung angewandt. Für das nächste Zeichen wird eine

neue, leere Leinwand an derselben Stelle erzeugt und die alte nach links verschoben. Über der aktiven Leinwand ist der bisher eingegebene Text zu sehen. Testpersonen konnten einen einfachen englischen Satz mit durchschnittlich 3 wpm und einer ER von 34% eingeben. Diese äußerst schlechte Performance lässt sich vermutlich darauf zurückzuführen, dass eine kleine Bewegung des Controllers eine viel größere Bewegung des Strahls auf der entfernten Leinwand zur Folge hatte. Ein leichtes Zittern der Hand wird so erheblich verstärkt und es bedarf einer sehr guten Feinmotorik.

Abbildung 2.15: On-air Handschrift basierend auf Controller-Pointing [13].

In [35] konnten mehrere Zeichen hintereinander geschrieben werden, um ein ganzes Wort oder einen Teil davon einzugeben und über eine virtuelle Enter-Taste zu bestätigen (Abb. 2.16). Anschließend wurde das Geschriebene von der Leinwand gelöscht und die erkannten Zeichen in einem darüberstehenden Textfeld angezeigt. Die Laserpointer-Methode war hier deutlich performanter mit 8,21 wpm und einer ER von nur 0,87%. Ob dies u.a. an einem geringeren Abstand zur Leinwand liegen könnte, lässt sich nicht sagen, da wie auch in [13] keine genauen Angaben zur Entfernung gegeben sind. Zwei weitere Bedienkonzepte wurden untersucht: Am präzisesten war es, die Leinwand direkt mit der Spitze des Controllers zu berühren und auf diese Weise wie mit einem Pinsel zu malen. 10,33 wpm und 0,5% ER konnten damit erreicht werden. Wurde stattdessen via Handtracking mit dem Zeigerfinger auf der Leinwand geschrieben, lag die Performance bei 7,66 wpm und 1,57% ER. Alle Varianten wurden jeweils auf einer flachen und einer gewölbten Leinwand getestet. Ersteres lieferte in allen Fällen etwas bessere Ergebnisse. In Interviews gaben Benutzer an, dass sie die Pinsel-Metapher bevorzugten, weil sie am intuitivsten und einfachsten zu erlernen sei. Das Schreiben mit dem Laserpointer sei aufgrund der hohen Sensitivität eher schwierig und erfordere eine Menge Konzentration; allerdings würde einem dabei weniger schwindelig, da die Leinwand weiter entfernt und dadurch immer komplett sichtbar war, was Augen und Nacken entlaste. Des Weiteren

gab ein Großteil der Befragten an, dass die Bedienung per Zeigefinger ermüdend sei und dass Tracking-Ungenauigkeiten und Verzögerungen ein Problem darstellten.



(a) Brush on Plane. (b) Brush on Hemisphere. (c) Gesture on Plane.

(d) Gesture on Hemisphere. (e) Raycast on Plane. (f) Raycast on Hemisphere.

Abbildung 2.16: Verschiedene Varianten der on-air Handschrift [35].

Handschrift als Texteingabe für VR ist somit weniger effizient als andere vorgestellte Methoden. Für stationäre Anwendungen sind physische Tastaturen besser geeignet als das Schreiben auf einem Grafiktablet. Die on-air Variante erzielt eine schlechtere Performance als die ebenfalls mobilen virtuellen Tastaturen, könnte sich aber als vorteilhaft erweisen, wenn mehr Symbole als nur die 26 Buchstaben des Alphabets unterstützt werden sollen (z.B. Ziffern, Sonderzeichen, andere Alphabete etc.). Eine Tastatur wird mit jedem zusätzlichen Zeichen unübersichtlicher und komplizierter zu bedienen; bei der Eingabe per Handschrift hingegen bleibt der Aufwand konstant [35].

2.2.4 Sprache

Sprache ist die natürlichste, effizienteste und bevorzugte Art menschlicher Kommunikation [24]. Moderne Spracherkennungssysteme erlauben eine dreimal so schnelle Eingabe eines englischen Textes auf dem Smartphone (mehr als 160 wpm) wie das Tippen auf der Tastatur und zugleich eine 20% niedrigere Fehlerrate [30]. Die Bedienung ist äußerst einfach und ergonomisch, da lediglich ein Button zum Starten und Beenden der Aufnahme gedrückt werden muss. Gängige HMDs sind standardmäßig mit einem Mikrofon ausgestattet, womit die Hardwarevoraussetzungen ebenfalls erfüllt sind. Man kann sich zudem während des Sprechens frei bewegen und benötigt keine weiteren Hilfsmittel.

Die Verwendung von Sprache zur Texteingabe in VR ist demnach sehr vielversprechend. Dennoch finden sich auch zu diesem Thema nur wenig aktuelle Studien.

In SWIFTER [27] verwendeten Benutzer einen Controller als Laserpointer, um ein virtuelles Interface zu bedienen, über das die Spracheingabe gestartet und Korrekturen vorgenommen werden konnten (Abb. 2.17). Das Interface zeigte den zuletzt eingesprochenen Satz, in dem sich sowohl die einzelnen Wörter zur Korrektur auswählen ließen als auch dazwischen eingefügte „+“-Symbole, um neue Wörter hinzuzufügen. Die Auswahl öffnete ein weiteres Menü mit einer Liste vorgeschlagener Wörter sowie Buttons zum Löschen und zum Aktivieren der Spracheingabe, um ein einzelnes Wort einzusprechen oder laut zu buchstabieren. Über einen weiteren Button neben dem Text wurde die Korrektur abgeschlossen und der Satz bestätigt. Auf diese Weise konnten Testpersonen ohne vorheriges Training einen deutschen Text mit 23,6 wpm eingeben. Die WER betrug 0,56%. Zum Vergleich wurde derselbe Text über ein Smartphone eingegeben, was 21 wpm und 0,23% WER lieferte. Eine Auswertung der Benutzerfreundlichkeit in Form von Fragebögen ergab eine klare Präferenz für SWIFTER. Auch die Wahrscheinlichkeit, die Verfahren in realen Szenarien einzusetzen, wurde für SWIFTER als signifikant höher eingestuft. Ein Teil der Testpersonen merkte jedoch an, dass repetitive Korrekturen viel Zeit kosteten, und es wurden Bedenken hinsichtlich der Eingabe beliebiger Wörter geäußert.

Abbildung 2.17: SWIFTER [27].

Selbst bei hoher Qualität der Spracherkennung kann es vorkommen, dass sich Fehler durch erneutes Einsprechen nur teilweise korrigieren lassen oder die Korrektur zu neuen Fehlern führt. Vor allem bei schwierigen Wörtern und in geräuschvoller Umgebung stellt

dies ein Problem dar. Es beeinträchtigt die Performance und kann zu Frustration führen, wenn mehrere Korrekturversuche nötig sind oder sich bestimmte Fehler gar nicht korrigieren lassen. Deshalb wird an dieser Stelle oft zu anderen Inputmodalitäten gewechselt [1]. Der in [1] präsentierte Ansatz verwendete Fingertapping zur Bedienung eines Korrekturinterfaces (Abb. 2.18). Angelehnt an SWIFTER, ließen sich die Wörter des zuletzt eingesprochenen Satzes sowie der Leerraum zwischen ihnen einzeln durch Antippen auswählen. Anstatt aber erneut von der Spracheingabe Gebrauch zu machen, wurde das neue bzw. korrigierte Wort daraufhin über eine virtuelle Tastatur eingetippt. Zusätzlich wurden unter jedem Wort ein oder mehrere Vorschläge angezeigt, die direkt ausgewählt werden konnten, ebenso wie ein Button zum Entfernen des Wortes. Eine Auswertung dieser Methode ergab 28 wpm und eine CER von 0,5% für einen Text, indem die Hälfte der Sätze ein *out-of-vocabulary* Wort enthielten, das nicht im Trainingsdatensatz des Sprachmodells vorhanden war (z.B. Eigennamen). Fast jedes dritte dieser unbekannten Wörter wurde nicht korrekt erkannt, was einer viermal so hohen Fehlerrate wie bei den *in-vocabulary* Wörtern entsprach. Dennoch konnte die Hälfte der Teilnehmer einen fehlerfreien Text verfassen. In einem weiteren Experiment wurde untersucht, wie gut sich Fehler durch erneutes Einsprechen oder Buchstabieren korrigieren ließen. Man fand, dass 43,5% der Korrekturversuche scheiterten. Dieses Ergebnis unterstreicht den Vorteil eines nicht sprachbasierten Korrekturmechanismus.

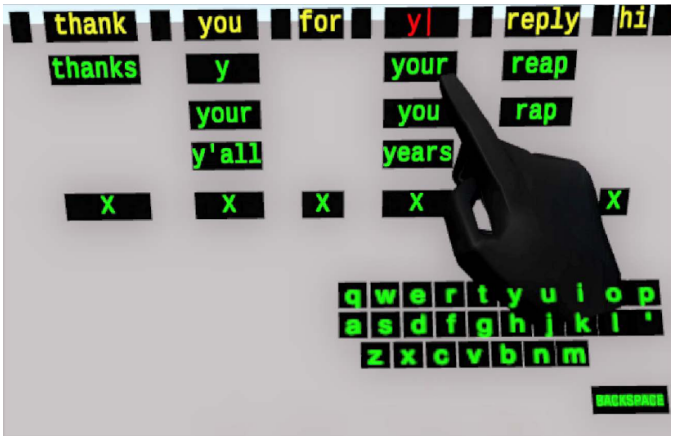


Abbildung 2.18: Spracheingabe unterstützt durch ein Korrekturinterface mit virtueller Tastatur [1].

2.3 Zusammenfassung

Ein Verfahren zur Texteingabe in VR lässt sich anhand verschiedener Kriterien bewerten. Dazu zählen neben Eingabegeschwindigkeit und Fehleranfälligkeit auch Ergonomie, Einsteigertauglichkeit, Mobilität, Immersion und Hardwarebedarf. Die in der neueren Literatur am häufigsten beschriebenen Texteingabeverfahren lassen sich in physische Tastaturen, virtuelle Tastaturen, Handschrift und Spracheingabe unterteilen, wobei es insbesondere bei den virtuellen Tastaturen eine Vielzahl unterschiedlicher Varianten und Bedienkonzepte gibt. Tabelle 2.1 fasst die Bewertung dieser Methoden hinsichtlich der wichtigsten Kriterien zusammen.

Die Literatur deutet darauf hin, dass von allen untersuchten Methoden die physische PC-Tastatur die schnellste Eingabe ermöglicht. Zugleich bringt diese aber auch eine große Einschränkung der Mobilität mit sich. Der Einsatz von speziellen mobilen Tastaturen oder Smartphones hat diesen Nachteil nicht, geht dafür laut Studien aber mit einer deutlich geringeren Geschwindigkeit und Nutzerfreundlichkeit einher. Für die mobile Anwendung scheinen virtuelle Tastaturen eine bessere Alternative darzustellen, insbesondere das Drumkeyboard ist hier hervorzuheben. Handschrift hat das Potenzial für eine natürliche Art der Interaktion, liefert in Form der on-air Handschrift in VR aber eine relativ schlechte Performance. Die besten Ergebnisse bezüglich Ergonomie und Einsteigertauglichkeit scheint die Spracheingabe zu liefern, die zudem als sehr performant eingestuft wird. In Kombination mit einer virtuellen Tastatur zur Korrektur scheint sie eine vielversprechende Methode zur mobilen Texteingabe in VR darzustellen.

Eingabeverfahren	**Geschwindigkeit (wpm)**	**Fehleranfälligkeit**	**Ergonomie**	**Einsteigertauglichkeit**	**mobil**
Physische Tastaturen					
PC-Tastatur	31-34	mittel	hoch	hoch	nein
Mobile Tastatur	3-12	hoch	hoch	gering	ja
Smartphone	9	hoch	mittel	mittel	ja
Tablet	13	mittel	gering	mittel	nein
Virtuelle Tastaturen					
Head-Pointing	11-15	mittel	mittel	hoch	ja
Controller-Pointing	15-17	mittel	hoch	hoch	ja
Tapping	10-15	gering	hoch	hoch	ja
Drumkeyboard	21-25	mittel	hoch	hoch	ja
Cursor	4-6	gering	hoch	mittel	ja
Gaze-Typing	8-10	gering	hoch	hoch	ja
Wortgesten	10-25	gering	mittel	mittel	ja
Handbasierte Methoden	10-12	hoch	mittel	gering	ja
Kreisförmige Layouts	7-13	mittel	hoch	mittel	ja
3D Layouts	22	-	-	-	ja
Handschrift					
On-air Handschrift	8-10	gering	mittel	hoch	ja
Sprache					
Korrektur durch erneutes Einsprechen	24	mittel	hoch	hoch	ja
Korrektur mithilfe virtueller Tastatur	28	gering	hoch	hoch	ja

Tabelle 2.1: Vergleich der verschiedenen Eingabeverfahren.

3 Analyse

Dieses Kapitel widmet sich der Anforderungsanalyse der zu entwickelnden Lösung. In Kapitel 3.1 wird zunächst das System aus fachlicher Sicht beschrieben. Kapitel 3.2 konkretisiert daraufhin die Verfahren zur Texteingabe. In Kapitel 3.3 werden die zu erfüllenden funktionalen und nicht-funktionalen Anforderungen spezifiziert und Kapitel 3.4 fasst die wichtigsten Punkte zusammen.

3.1 Fachliche Beschreibung des Systems

Kapitel 3.1.1 beschreibt das Ist-System und Kapitel 3.1.2 die gewünschte Erweiterung um die Möglichkeit der Texteingabe.

3.1.1 Ist-System

Bei dem bestehenden System handelt es sich um eine VR-Umgebung zur Durchführung virtueller Soundwalks. Der Nutzer trägt ein HMD, über das ihm die realistische Nachbildung eines Windparks präsentiert wird (Abb. 3.1). Neben dem visuellen Aspekt werden mittels der im HMD integrierten Kopfhörer auch einige Geräuschquellen in Form von 3D-Audio simuliert. Des Weiteren hält der Nutzer in beiden Händen einen Controller, welcher zur Interaktion dient. Position und Ausrichtung des HMD und der Controller im Raum werden in Echtzeit in die Simulation übertragen und dem Nutzer werden virtuelle Abbilder der Controller inklusive Bedienhinweisen angezeigt (Abb. 3.2).

Abbildung 3.1: Windparksimulation [Eigene Grafik].

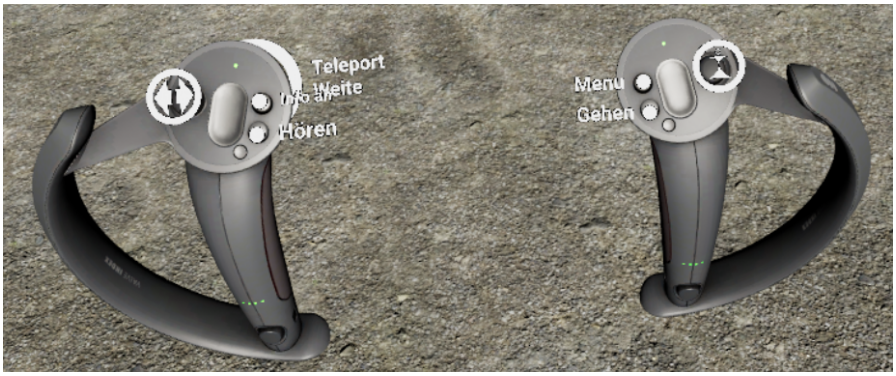


Abbildung 3.2: Virtuelle Controller mit Bedienhinweisen [Eigene Grafik].

Um eine natürliche Erkundung der virtuellen Umgebung zu ermöglichen, findet das Erlebnis im Stehen statt und der Nutzer kann sich innerhalb eines Trackingsbereichs frei bewegen. Da dieser aber auf wenige Quadratmeter beschränkt und damit deutlich kleiner als das Gebiet des Soundwalks ist, stehen zusätzlich zwei rein virtuelle Navigationsarten zur Verfügung, zwischen denen durch Drücken eines Buttons gewechselt werden kann: Beim „Spazierengehen“ wird durch Vorwärtsdrücken des rechten Joysticks eine kontinuierliche Bewegung innerhalb der Simulation veranlasst. Diese erfolgt entweder in Blickrichtung oder in die Richtung, in welche der rechte Controller zeigt. Alternativ dazu ist die Fortbewegung mittels Teleportation möglich. In diesem Modus bewirkt das Vorwärtsdrücken des Joysticks die Projektion eines Strahls in die virtuelle Welt, welcher ausgehend vom

rechten Controller entlang der Wurfbahn eines gedachten Objekts verläuft. Die Stelle, an der der Strahl auf den Boden trifft, wird durch einen Ring besonders hervorgehoben. Dies entspricht der Position, an die der Benutzer durch Loslassen des Joysticks teleportiert wird. Die Entfernung kann durch Drücken des linken Triggers erhöht werden. Auf diese Weise lassen sich auch größere Distanzen schnell überwinden.

Zu Beginn des Experiments wird der Nutzer mit der Steuerung vertraut gemacht und füllt (ebenfalls in VR) einen Fragebogen mit Angaben zu seiner Person aus (Abb. 3.3). Im Anschluss besucht er nacheinander die verschiedenen Stationen des Soundwalks. An jeder Station öffnet sich nach einer einminütigen Hörphase, in der auf die Umgebungsgeräusche gelauscht wird, ebenfalls ein Fragebogen. Die Beantwortung der Fragen erfolgt stets mithilfe des linken Controllers: Durch seitliches Drücken des Joysticks wird entweder ein Slider auf einer Skala verschoben oder durch eine diskrete Menge an Auswahlmöglichkeiten navigiert. Zum Bestätigen der Angabe wird ein Button gedrückt, woraufhin zur nächsten Frage gewechselt oder der Fragebogen geschlossen wird, falls er vollständig ausgefüllt wurde.

Abbildung 3.3: Ausschnitte aus dem Fragebogen [Eigene Grafik].

Das System kommuniziert über einen Server mit einer Datenbank. Vorab werden so die Koordinaten und Namen der Stationen und die anzuzeigenden Fragen ermittelt. Die eingegeben Antworten werden wiederum an den Server gesendet und in der Datenbank gespeichert.

3.1.2 Soll-System

Die Fragebögen, die im Rahmen des virtuellen Soundwalks direkt in VR ausgefüllt werden, sollen um zwei Arten von Fragen ergänzt werden, welche die Eingabe von Texten erfordern: Bei **Freitextfragen** können ein oder mehrere Sätze als Antwort formuliert werden. „Was geht Ihnen durch den Kopf?“ ist ein Beispiel für eine solche Frage. Der andere Fragetyp umfasst eine Aufzählung von Stichpunkten, deren Reihenfolge im Nachhinein angepasst werden kann, um eine Priorisierung vorzunehmen. Eine solche **Aufzählungsfrage** lautet z.B. „Bitte führen Sie die von Ihnen festgestellten Schallquellen in absteigender Reihenfolge auf, beginnend mit der auffälligsten Schallquelle“.

Wird eine Freitext- oder Aufzählungsfrage angezeigt, so sollen je nach Fragetyp die entsprechenden Bedienelemente zur Texteingabe aktiviert werden.

3.2 Auswahl des Texteingabeverfahrens

Aus der Recherche in Kapitel 2 geht hervor, dass für die mobile Texteingabe in VR das Drumkeyboard und die Spracherkennung am vielversprechendsten sind. Beide scheinen eine ähnlich gute Performance und Benutzerfreundlichkeit zu bieten. Die Ergebnisse deuten zudem darauf hin, dass es von Vorteil sein kann, die Spracherkennung durch eine andere Inputmodalität zu unterstützen, mit der sich Korrekturen zuverlässiger durchführen lassen. Da dies jedoch nicht eindeutig ist, sollen sowohl die Spracheingabe als auch das Drumkeyboard implementiert werden, sodass diese unabhängig voneinander und in Kombination eingesetzt werden können. In Benutzertests sollen die drei Varianten miteinander verglichen werden.

3.3 Anforderungen

Im Folgenden werden die genauen Anforderungen an die zu entwickelnde Lösung spezifiziert. Diese untergliedern sich in die funktionalen Anforderungen in Kapitel 3.3.1 und die nicht-funktionalen Anforderungen in Kapitel 3.3.2.

3.3.1 Funktionale Anforderungen

Die funktionalen Anforderungen (FAs) beschreiben die Funktionen, die das System dem Nutzer bereitstellen soll, und die technischen Voraussetzungen, die es erfüllen muss.

FA1 Freitextfragen: Der Nutzer kann Freitextfragen mit einem Text aus einem oder mehreren Sätzen inklusive Absätzen beantworten.

FA2 Aufzählungsfragen: Der Nutzer kann Aufzählungsfragen mit einem oder mehreren Stichpunkten beantworten, deren Reihenfolge anschließend angepasst werden kann, um eine Priorisierung vorzunehmen.

FA3 Optionale Beantwortung: Die Beantwortung der Freitext- und Aufzählungsfragen ist optional. Möchte der Nutzer eine Frage nicht beantworten, kann er einen leeren Text bzw. eine Liste leerer Stichpunkte eingeben.

FA4 Zeichenvorrat: Die vom Nutzer eingegebenen Texte können folgende Zeichen enthalten:

- Die 26 Buchstaben des Alphabets und die deutschen Umlaute, jeweils als Groß- und Kleinbuchstaben, sowie das „ß“
- Leerzeichen
- Die Ziffern von 0 bis 9
- Die Sonderzeichen Punkt, Komma, Doppelpunkt, Semikolon, Bindestrich, Ausrufe- und Fragezeichen, Anführungszeichen, Apostroph, Und-Zeichen, runde Klammern, Gradzeichen, Prozentzeichen, Eurozeichen, Gleichheitszeichen und Schrägstrich

FA5 Korrektur: Der Nutzer kann einen Text während der Eingabe beliebig korrigieren, indem jedes bisher eingegebene Wort angepasst oder entfernt und an jeder Stelle ein neues Wort eingefügt werden kann.

FA6 Mobilität: Der Nutzer wird durch die Texteingabe nicht in seiner Mobilität eingeschränkt, d.h. sie kann im Stehen erfolgen und der Nutzer kann sich währenddessen frei bewegen.

FA7 Immersion: Die Texteingabe erfordert kein visuelles, auditives oder anderweitiges Einblenden der realen Umgebung in die vom Nutzer wahrgenommene virtuelle Welt.

FA8 Hardware: Die Texteingabe erfolgt allein über die bereits im Projekt eingesetzte Hardware.

FA9 Systemanbindung: Die Texteingabe soll in das bestehende System eingebunden und Bestandteil des Fragebogens werden.

FA10 Benutzertests: Es sollen Benutzertests durchgeführt werden, welche die Texteingabe mittels Sprache, Drumkeyboard und der Kombination aus beiden Verfahren miteinander vergleichen. Dabei sollen sowohl objektive Messungen der Eingabe- und Fehlerrate als auch eine subjektive Einschätzung der Probanden ermittelt werden.

3.3.2 Nicht-funktionale Anforderungen

Die nicht-funktionalen Anforderungen (NFAs) beschreiben die geforderten Qualitätsstandards sowie Rahmenbedingungen in Bezug auf die eingesetzten Drittanbieter-Services.

NFA1 Einsteigertauglichkeit: Die Texteingabe soll intuitiv und einsteigertauglich gestaltet sein, sodass erstmalige Nutzer nur eine kurze Eingewöhnungsphase benötigen.

NFA2 Ergonomie: Die Texteingabe soll vom Nutzer als wenig bis gar nicht körperlich oder mental anstrengend oder unangenehm empfunden werden.

NFA3 Effizienz: Die Texteingabe soll mit einer Geschwindigkeit und Fehleranfälligkeit durchführbar sein, die vom Nutzer als akzeptabel oder besser eingestuft wird.

NFA4 Vermeidung kostenpflichtiger Services: Das System soll keine kostenpflichtigen Services verwenden, auch nicht solche mit kostenlosen Probezeiträumen, Kostenfreiheit bis zu einem gewissen Nutzungslimit oder einem Startguthaben für Neukunden.

NFA5 Datenschutz: Die Nutzereingaben – sei es in Form von Text, Sprache oder auf andere Weise – sollen zur Verarbeitung nicht an fremde Server gesendet werden, deren anderweitige Nutzung, Speicherung oder Weiterverbreitung der Daten nicht kontrolliert werden kann.

3.4 Zusammenfassung

Das bestehende System realisiert die Durchführung von Soundwalks in VR. Teil eines virtuellen Soundwalks ist das Ausfüllen mehrerer Fragebögen. Letztere sollen zukünftig auch Fragen beinhalten, die die Eingabe eines Freitexts oder einer sortierten Aufzählung von Stichpunkten erfordern. Die zu entwickelnde Lösung soll die Beantwortung dieser Fragen ermöglichen und in das System integriert werden. Sie soll zudem mobil und immersiv sein und keiner zusätzlichen Hardware bedürfen. Weiter soll in Benutzertests untersucht werden, welches der ausgewählten Texteingabeverfahren – Spracheingabe, Drumkeyboard sowie deren Kombination – die besten Ergebnisse liefert. Neben diesen funktionalen gibt es auch nicht-funktionale Anforderungen, die eine nutzerfreundliche und effiziente Bedienung sowie die Vermeidung kostenpflichtiger Services und einen sicheren Umgang mit den eingegebenen Daten sicherstellen sollen.

4 Entwurf

Im vorherigen Kapitel wurden mehrere Texteingabeverfahren ausgewählt und detaillierte Anforderungen an die zu entwickelnde Lösung gestellt. In diesem Kapitel wird die Realisierung dieser Anforderungen beschrieben. Kapitel 4.1 betrachtet den technischen Kontext und Kapitel 4.2 geht auf die Spracherkennung ein. Anschließend beschreibt Kapitel 4.3 die Lösung aus fachlicher Sicht und Kapitel 4.4 den zugrundeliegenden Softwareentwurf. Kapitel 4.5 fasst die wesentlichen Aspekte zusammen.

4.1 Technischer Kontext

Die im Projekt verwendete Hardware und Software liefern die Rahmenbedingungen für eine mögliche Lösung (vgl. FA8 und FA9). Dieses Unterkapitel gibt deshalb einen Überblick über den technischen Kontext des zu erstellenden Entwurfs. Kapitel 4.1.1 beschreibt die eingesetzte VR-Hardware und Kapitel 4.1.2 die eingesetzte Software.

4.1.1 Hardwareumgebung

Im Projekt kommt das *Valve Index*-HMD mit den zugehörigen Controllern und Tracking-Sensoren zum Einsatz. Durch letztere wird die Position von HMD und Controllern innerhalb des Trackingsbereichs in Echtzeit ermittelt. Das HMD besitzt zwei integrierte Kopfhörer und an der Unterseite ein integriertes Mikrofon. Damit lässt sich die Sprache des Nutzers auch bei verschiedenen Neigungen des Kopfes und in Bewegung konsistent aufzeichnen, da der Abstand zum Mund immer gleichbleibt. Die Controller verfügen über mehrere Buttons, einen Joystick und ein Touchpad auf der Oberseite. An der Rückseite befindet sich ein Trigger. Sie werden über Gurte an den Handflächen befestigt und müssen somit nicht mit den Fingern gehalten werden. Dies hat zur Folge, dass der Griff ebenfalls als Touchpad dienen kann, mit dem sich erkennen lässt, ob die einzelnen Finger ausgestreckt oder angewinkelt sind.

4.1.2 Softwareumgebung

Die Software besteht aus einem Projekt in der Unreal Engine 4 (UE4). Über Plugins sind verschiedene Subsysteme wie die VR-spezifischen Inhalte und die Client-Komponente zur Kommunikation mit dem Server in das Hauptprojekt eingebunden. Die Inputs der VR-Hardware werden über das *SteamVR*-Interface abstrahiert, welches ebenfalls als Plugin vorliegt. UE4 bietet die Möglichkeit, zur Laufzeit Audio über das HMD-Mikrofon aufzunehmen und als WAV-Datei zu speichern. Auch sind Schnittstellen für die Kommunikation über das Netzwerk verfügbar.

4.2 Spracherkennung

Zur Umsetzung der in der Analyse ausgewählten Texteingabeverfahren bedarf es einer Lösung für die Spracherkennung. Kapitel 4.2.1 befasst sich mit der Auswahl einer passenden Software und Kapitel 4.2.2 mit ihrer Anbindung an UE4.

4.2.1 Auswahl der Spracherkennungssoftware

Zur Auswahl einer geeigneten Lösung für die Spracherkennung wurden einige Kandidaten untersucht. Kostenpflichtige APIs wie *IBM Watson*[1] oder *Google Cloud Speech-to-Text*[2] müssen trotz guter Performance aufgrund von NFA4 und NFA5 ausgeschlossen werden. Die übrigen untersuchten Optionen erfüllen diese Anforderungen, da es sich um frei verfügbare Offline-Modelle handelt. *Julius*[3] entfällt, da es Deutsch nicht unterstützt. *Sphinx*[4] und *Mozilla Deep Speech*[5] eignen sich für die deutsche Sprache, lieferten in eigenen Tests aber keine zufriedenstellenden Ergebnisse bei der Erkennung. Letztendlich kommen somit nur *Vosk*[6] und *Whisper*[7] in Betracht, die sich beide durch eine hohe Geschwindigkeit und geringe Fehlerrate auszeichnen. Im Gegensatz zu *Whisper* unterstützt *Vosk* das kontinuierliche Streaming der erkannten Wörter während des Sprechens. Der Text wird nach und nach ausgegeben, wodurch sich Fehler frühzeitig erkennen und berichtigen lassen.

[1] https://www.ibm.com/de-de/cloud/watson-speech-to-text (abgerufen am 17.06.23)
[2] https://cloud.google.com/speech-to-text/ (abgerufen am 17.06.23)
[3] https://github.com/julius-speech/julius (abgerufen am 17.06.23)
[4] https://cmusphinx.github.io/ (abgerufen am 17.06.23)
[5] https://github.com/mozilla/DeepSpeech/ (abgerufen am 17.06.23)
[6] https://alphacephei.com/vosk/ (abgerufen am 17.06.23)
[7] https://github.com/openai/whisper (abgerufen am 17.06.23)

Bei *Whisper* hingegen muss ein Text vollständig eingesprochen, die Aufnahme beendet und anschließend eine kurze Zeit gewartet werden, bevor das vollständige Ergebnis erscheint. Damit ist die Spracheingabe etwas langsamer und der Korrekturaufwand ggf. höher, wenn sich erst im Nachhinein herausstellt, dass das Ergebnis viele Fehler enthält. Auf der anderen Seite besteht ein großer Nachteil von *Vosk* darin, dass die Ausgabe kaum Satzzeichen enthält und die Groß- und Kleinschreibung größtenteils fehlerhaft ist. *Whisper* wiederum macht sowohl von der korrekten Groß- und Kleinschreibung als auch der korrekten Verwendung von Satzzeichen Gebrauch. Aus diesem Grund wurde *Whisper* trotz mangelnder Streamingfunktionalität für die Spracherkennung ausgewählt. Um die Zeit, in der auf das Ergebnis gewartet wird, möglichst kurz zu halten, wurde von den verfügbaren Modellen das *tiny*-Modell gewählt, welches das kleinste und deshalb schnellste Modell ist. In eigenen Tests beläuft sich die Verarbeitungszeit auf ca. eine Sekunde, was akzeptabel ist, da ein oder mehrere Sätze am Stück eingesprochen werden können. Die Fehlerrate ist aufgrund der geringen Größe höher als bei den übrigen *Whisper*-Modellen, aber immer noch ausreichend klein.

4.2.2 Anbindung der Spracherkennung

Whisper liegt ausschließlich als Pythonmodul vor und lässt sich deshalb nicht direkt in UE4 einbinden. Es wird deshalb Teil eines in Python implementierten Websocket-Servers (von hier an als **Transkriptionsserver** bezeichnet), der als eigener Prozess läuft und über das Netzwerk mit UE4 kommuniziert. Diese Umsetzung hat den Vorteil einer sehr losen Kopplung, wodurch der Server bei Bedarf gegen eine andere Implementierung, möglicherweise basierend auf einem anderen Spracherkennungssystem, ausgetauscht werden kann. Das Websocket-Protokoll wurde gewählt, da es TCP als Transportprotokoll verwendet und somit eine zuverlässige Kommunikation bietet, gleichzeitig aber auch eine weit verbreitete und einfach zu verwendende Abstraktion zu TCP darstellt, die eine Aufteilung des Datenstroms in voneinander getrennte Nachrichten vornimmt.

Wie im technischen Kontext beschrieben, speichert UE4 das aufgezeichnete Audio als WAV-Datei. Zur Transkription sendet die entsprechende UE4-Komponente den Dateipfad an den Transkriptionsserver. Dieser übergibt den Pfad an das *Whisper*-Modul und sendet den erhaltenen Ergebnistext als Antwort zurück. Anschließend löscht der Transkriptionsserver die Datei, um den Speicherplatz freizugeben.

Damit die aktuell verarbeitete Datei nicht durch eine neue überschrieben wird, wird jede Datei mithilfe eines Zählers eindeutig benannt.

4.3 Fachliches Design

Nachdem die Möglichkeiten und Einschränkungen des technischen Kontexts und der Spracheingabe dargelegt wurden, diskutiert dieses Unterkapitel darauf aufbauend das fachliche Design der Lösung. Dies geschieht vor dem Hintergrund der in Kapitel 3 definierten Anforderungen. Kapitel 4.3.1 widmet sich dem Drumkeyboard, Kapitel 4.3.2 der Spracheingabe und Kapitel 4.3.3 dem Korrekturinterface. Kapitel 4.3.4 erläutert die Integration der Elemente in den Fragebogen.

4.3.1 Drumkeyboard

Das grundlegende Design des Drumkeyboards orientiert sich an der in Kapitel 2.2.2 vorgestellten Literatur. Das Layout entspricht größtenteils der deutschen PC-Tastatur; ein paar Anpassungen sind nötig, um den in FA4 spezifizierten Zeichenvorrat abzubilden (Abb. 4.1). So wurde das Paragraphenzeichen durch das Gradzeichen ersetzt, das Dollarzeichen durch das Eurozeichen und der Unterstrich durch den Apostroph. Zusätzlich wurde eine **Aufnahme-Taste** für das Starten und Stoppen der Spracheingabe hinzugefügt, die mit einem Mikrofonsymbol gekennzeichnet ist. Die Backspace-Taste ist nicht Teil des Drumkeyboards, sie wurde nach vorläufigen Tests auf den Trigger des linken Controllers gelegt, was ein schnelles mehrmaliges Betätigen erleichtert, wenn mehrere Zeichen gelöscht werden sollen. Ein Bedienhinweis am Controller zeigt dies an. Von beiden Controllern gehen virtuelle Drumsticks aus, mit denen die Tasten des Drumkeyboards angeschlagen werden können (Abb. 4.2).

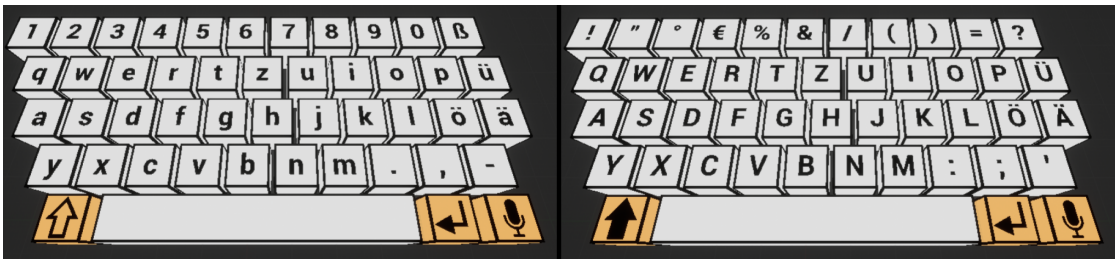


Abbildung 4.1: Das Drumkeyboard ohne (links) und mit aktivierter Shift-Taste (rechts) [Eigene Grafik].

Abbildung 4.2: Virtuelle Abbilder der Controller mit Drumsticks und Bedienhinweis für die Backspace-Taste [Eigene Grafik].

Mit der Shift-Taste (unten links) kann zwischen Kleinbuchstaben/Ziffern und Großbuchstaben/Sonderzeichen umgeschaltet werden. Eine Taste zeigt dabei immer das Zeichen an, das aktuell über sie eingegeben werden kann (analog zu Smartphone-Tastaturen). Um Tastendrücke zu sparen, wird die Shift-Taste nach Eingabe eines Zeichens automatisch deaktiviert, da meist nur ein einzelner Buchstabe großgeschrieben wird (der Anfang eines Wortes) bzw. ein einzelnes Sonderzeichen eingegeben wird. Zu Beginn eines neuen Satzes wird sie hingegen automatisch aktiviert. Die Shift-, Enter- und Aufnahme-Taste sind zum schnellen Auffinden farblich hervorgehoben.

Eine Taste wird bei Berührung mit dem Drumstick nicht sofort angeschlagen, sondern muss ein kleines Stück nach unten gedrückt werden, um ein versehentliches Anschlagen unwahrscheinlicher zu machen. Das Überschreiten des Schwellwerts wird dem Nutzer durch visuelles und auditives Feedback mitgeteilt, indem die Taste ihre Farbe zu Blau ändert und ein Klickgeräusch ertönt. Beim Loslassen nimmt die Taste wieder ihre ursprüngliche Farbe an (mit Ausnahme der Aufnahme-Taste, siehe Kapitel 4.3.2). Als zusätzliche Unterstützung befindet sich direkt hinter der Tastatur ein schmales Textfeld, welches das aktuell eingegebene Wort anzeigt (Abb. 4.3). Dadurch muss zur Verifikation nicht so häufig der Kopf gehoben werden, um einen Blick auf das große Textfeld über der Tastatur zu werfen.

Um eine bequeme Bedienung zu gewährleisten, kann die Höhe des Drumkeyboards angepasst werden. Dafür befinden sich rechts und links neben der Tastatur spezielle Bedienelement, genannt *Handles*, die entsprechend mit Pfeilen nach oben und unten gekennzeichnet sind. Bei Berührung mit dem Drumstick kann der Nutzer anschließend durch Heben oder Senken der Hand das Drumkeyboard vertikal bewegen. Abb. 4.3 zeigt ein gesamtheitliches Bild.

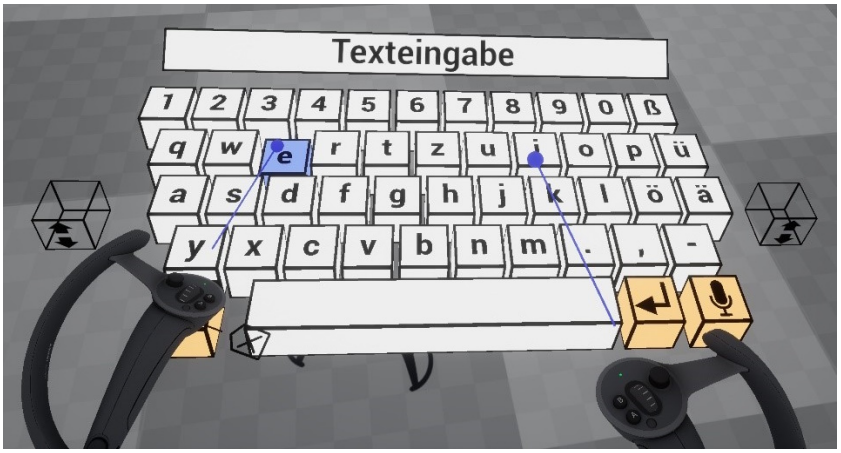


Abbildung 4.3: Interaktion mit dem Drumkeyboard. Eingabe des Wortes „Texteingabe“, die Taste „e“ wird gedrückt [Eigene Grafik].

4.3.2 Spracheingabe

Das Anschlagen der Aufnahme-Taste aktiviert die Spracheingabe. Währenddessen sind die übrigen Tasten deaktiviert und können nicht gedrückt werden. Visuell wird dies dadurch signalisiert, dass die Aufnahme-Taste ihre Farbe zu Rot und alle anderen zu Grau ändern. Zusätzlich wird das aktive Textfeld rot umrandet (siehe Kapitel 4.3.3). Nun kann der Nutzer etwas einsprechen.

Durch erneutes Anschlagen der Aufnahme-Taste wird die Spracheingabe beendet und die Aufnahme-Taste ebenfalls deaktiviert. Sobald das Ergebnis vorhanden ist, wird es dem aktiven Textfeld hinzugefügt und alle Tasten kehren in den ursprünglichen, aktivierten Zustand zurück. Falls der Ergebnistext mit einem vollständigen Satz endet (d.h. das letzte Zeichen ist ein Punkt, Ausrufezeichen oder Fragezeichen), wird zusätzlich ein Leerzeichen angehängt.

4.3.3 Korrekturinterface

Über dem Drumkeyboard befindet sich das Korrekturinterface. Bei Freitextfragen besteht dieses aus einem großen Textfeld, in dem ein mehrere Zeilen langer Ausschnitt des eingegebenen Textes sichtbar ist (Abb. 4.4). Ein blinkender Cursor zeigt die aktuelle Position an, an der weitere Zeichen hinzugefügt werden.

Abbildung 4.4: Das Korrekturinterface für Freitextfragen über dem Drumkeyboard [Eigene Grafik].

Für ein einheitliches Bedienkonzept findet die Interaktion mit dem Korrekturinterface ebenfalls über die Drumsticks statt. Durch Berühren des Textfelds kann der Cursor an die entsprechende Position gesetzt werden. Ein Entlangziehen des Drumsticks über das Textfeld markiert die darunterliegenden Zeichen, die anschließend entfernt oder ersetzt werden können. Auf beiden Seiten des Textfelds befinden sich zudem jeweils zwei Buttons,

über die durch Berühren nach oben oder unten gescrollt werden kann. Somit lässt sich der Text beliebig anpassen (vgl. FA5).

Bei Aufzählungsfragen besteht das Korrekturinterface aus mehreren vertikal angeordneten Textfeldern, die jeweils nur eine Zeile hoch sind und zur Eingabe eines Stichpunktes dienen (Abb. 4.5). Es ist jederzeit genau eines dieser Textfelder aktiv, d.h. dort wird der Cursor angezeigt und neuer Text hinzugefügt. Die übrigen Textfelder sind ausgegraut. Die Korrektur erfolgt genau wie bei den Freitextfragen über die Drumsticks; durch Berühren eines Textfelds wird dieses zugleich aktiviert. Zu Beginn werden standardmäßig drei Textfelder erzeugt, wovon das oberste aktiv ist.

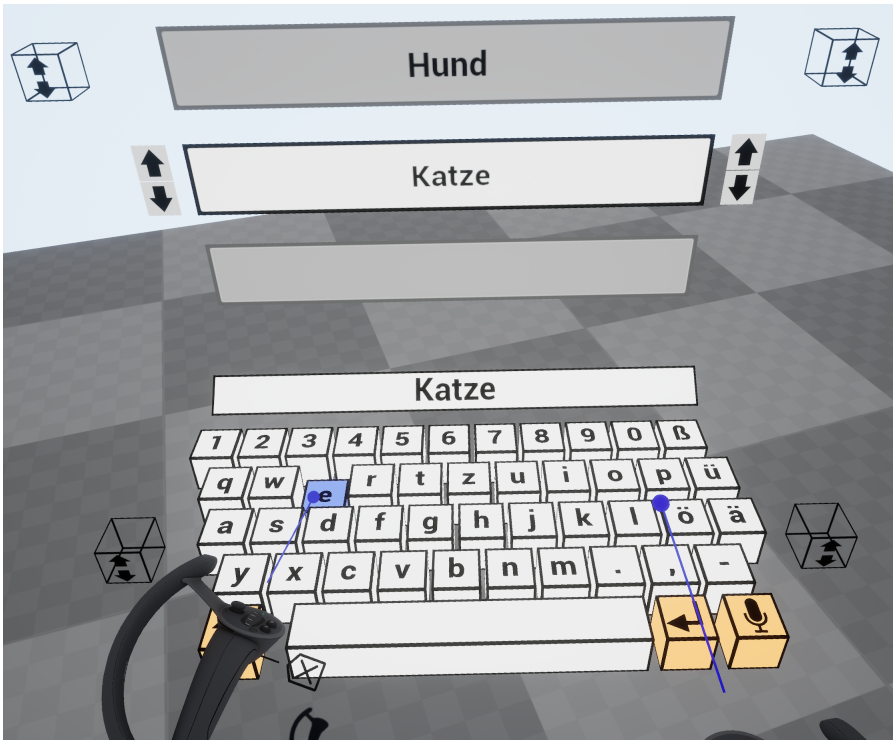


Abbildung 4.5: Das Korrekturinterface für Aufzählungsfragen über dem Drumkeyboard. Das mittlere Textfeld ist aktiv [Eigene Grafik].

Während bei Freitextfragen über die Enter-Taste ein neuer Absatz begonnen werden kann, dient diese bei Aufzählungsfragen dazu, in das darunterliegende Textfeld zu wechseln. Ist bereits das unterste Textfeld aktiviert, wird unter diesem ein neues, leeres Textfeld erzeugt. Zum Entfernen eines Textfelds muss zunächst der gesamte Inhalt gelöscht

und dann erneut Backspace gedrückt werden. Auf diese Weise kann eine variable Anzahl an Stichpunkten eingegeben werden.

Über die Buttons, die bei Freitextfragen zum Scrollen dienen, kann bei Aufzählungsfragen stattdessen die Reihenfolge der Textfelder angepasst werden, um eine Priorisierung vorzunehmen. So lässt sich ein Textfeld durch Drücken des entsprechenden Buttons um eine Position nach oben oder unten verschieben. Da das oberste und unterste Textfeld nur in jeweils eine Richtung verschoben werden können, werden hier nur die dafür zuständigen Buttons angezeigt.

Zur Unterstützung der Ergonomie verfügt das Korrekturinterface genau wie das Drumkeyboard über zwei Handles, mit denen bei Freitextfragen die Höhe des Textfeldes eingestellt werden kann und bei Aufzählungsfragen sämtliche Textfelder gemeinsam nach oben oder unten bewegt werden können.

4.3.4 Integration in den Fragebogen

Während der Beantwortung des Fragebogens wird bei einer Freitext- oder Aufzählungsfrage zunächst nur der Fragetext angezeigt, damit dieser nicht durch das Korrekturinterface verdeckt wird. Erst nach erneutem Drücken des Bestätigungsbuttons werden das Drumkeyboard, die Drumsticks und das zum Fragetyp passende Korrekturinterface erzeugt und die Bedienhinweise der Controller angepasst. Abb. 4.6 zeigt dies am Beispiel einer Aufzählungsfrage. Der Benutzer kann daraufhin den Text oder die Stichwortliste eingeben und anschließend wie gewohnt den Bestätigungsbutton betätigen, um die Antwort abzusenden, woraufhin Drumkeyboard usw. wieder entfernt werden.

Abbildung 4.6: Beantwortung einer Aufzählungsfrage im Rahmen des Fragebogens [Eigene Grafik].

4.4 Architektur

Nachdem die fachliche Sicht erläutert wurde, dokumentiert der folgende Abschnitt die Softwarearchitektur. Kapitel 4.4.1 gibt einen Gesamtüberblick über das System und die Integration der Lösung zur Texteingabe. Die weiteren Unterkapitel widmen sich den einzelnen Komponenten und ihrem Zusammenspiel.

4.4.1 Systemüberblick

Aus dem technischen Kontext geht hervor, dass sich die Integration in das vorhandene Projekt am einfachsten gestaltet, wenn die Lösung zur Texteingabe als UE4-Plugin umgesetzt wird. Das UE4-Projekt, in das dieses Plugin eingebettet ist, wird im Folgenden

als *Soundwalk Application* und das Plugin selbst als *Text Entry VR Plugin* bezeichnet. Abb. 4.7 zeigt die Komponenten des Systems sowie des Transkriptionsservers und ihre Abhängigkeiten zueinander. Es werden nur die relevanten Komponenten des Ist-Systems und der UE4 betrachtet.

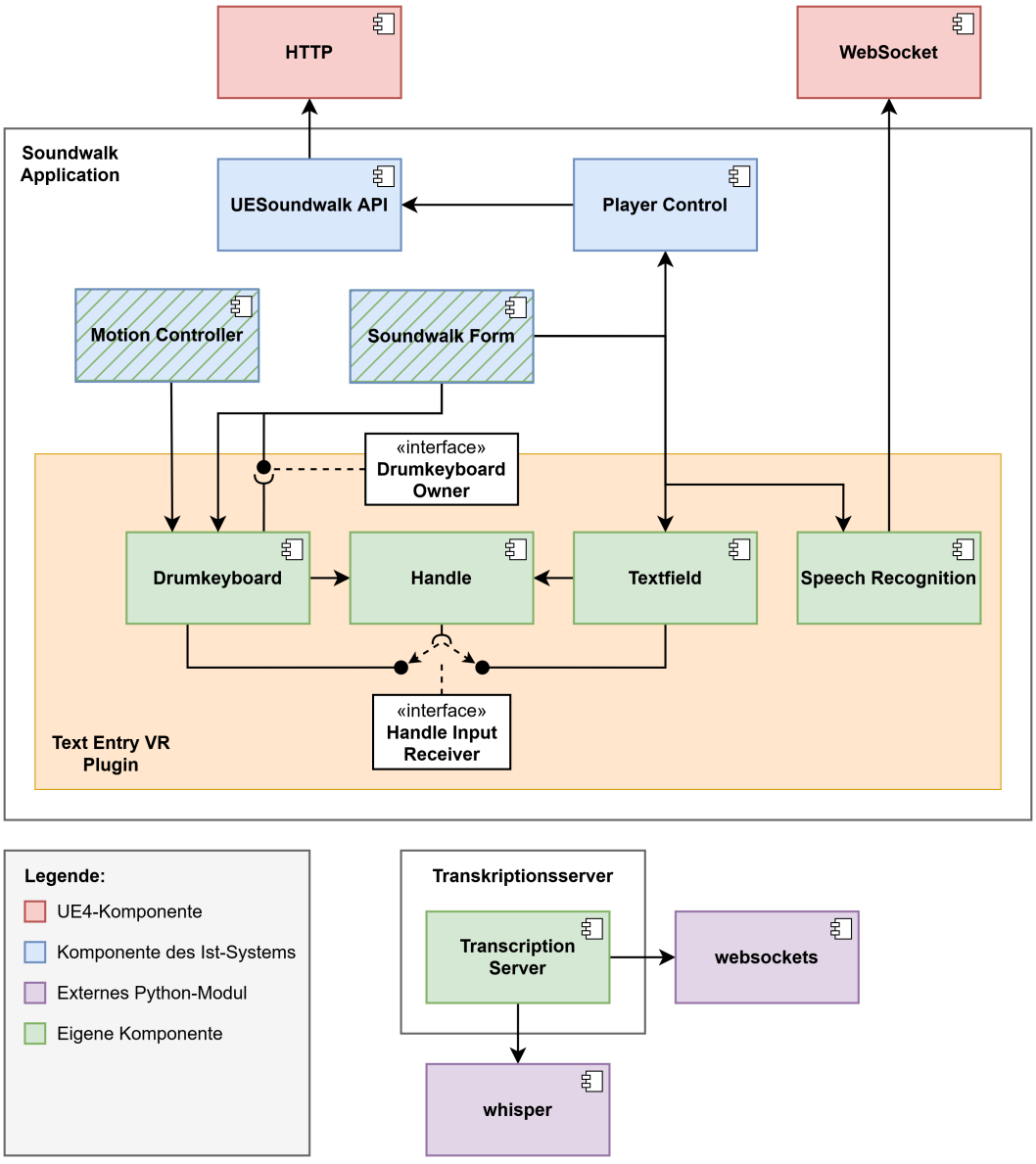


Abbildung 4.7: Komponentensicht des Gesamtsystems [Eigene Grafik].

4.4.2 Drumkeyboard

Die *Drumkeyboard*-Komponente beinhaltet das Drumkeyboard und die zugehörigen Drumsticks. Abb. 4.8 zeigt den internen Aufbau. Das Drumkeyboard wird durch die gleichnamige Klasse repräsentiert und dessen Tasten durch die Klasse *Drumkeyboard Key*, die durch fünf Subklassen für verschiedene Arten von Tasten konkretisiert wird. *Drumkeyboard Key* vereint die gemeinsame Funktionalität aller Tasten des Drumkeyboards: Die Interaktion mit den Drumsticks, das Informieren des Drumkeyboards, wenn die Taste gedrückt wurde, und das Aktivieren/Deaktivieren. Über Hook-Methoden können die Subklassen ihr spezielles Verhalten einbinden.

Character Key repräsentiert eine Taste, über die ein Zeichen eingegeben werden kann (Buchstabe, Ziffer oder Sonderzeichen). Die Leertaste wird durch die Klasse *Drumkeyboard Space Key* besonders behandelt, weil ein Leerzeichen das Ende eines Wortes markiert und bei dessen Eingabe beispielsweise der Inhalt des schmalen Textfelds über der Tastatur (Klasse *Wordfield*) gelöscht werden muss. Die Shift-, Enter- und Aufnahme-Taste haben ebenfalls besondere Aufgaben und werden deshalb als eigene Klasse umgesetzt.

Um externe Komponenten über das Anschlagen einer Taste informieren zu können und Tasten aktivieren/deaktivieren zu lassen, definiert die Komponente das Interface *Drumkeyboard Owner*, mit dem *Drumkeyboard* arbeitet. Die Klasse implementiert wiederum das von der Komponente *Handle* zur Verfügung gestellte Interface *Handle Input Receiver*, sodass mitgeteilt werden kann, dass sich das Drumkeyboard nach oben oder unten bewegen soll (siehe Kapitel 4.4.3).

Die Klasse *Drumstick* benötigt keine Abhängigkeiten zu den übrigen Klassen, da die Interaktion mit den Tasten (d.h. das Überprüfen auf eine Kollision) allein über die Physiksimulation stattfindet.

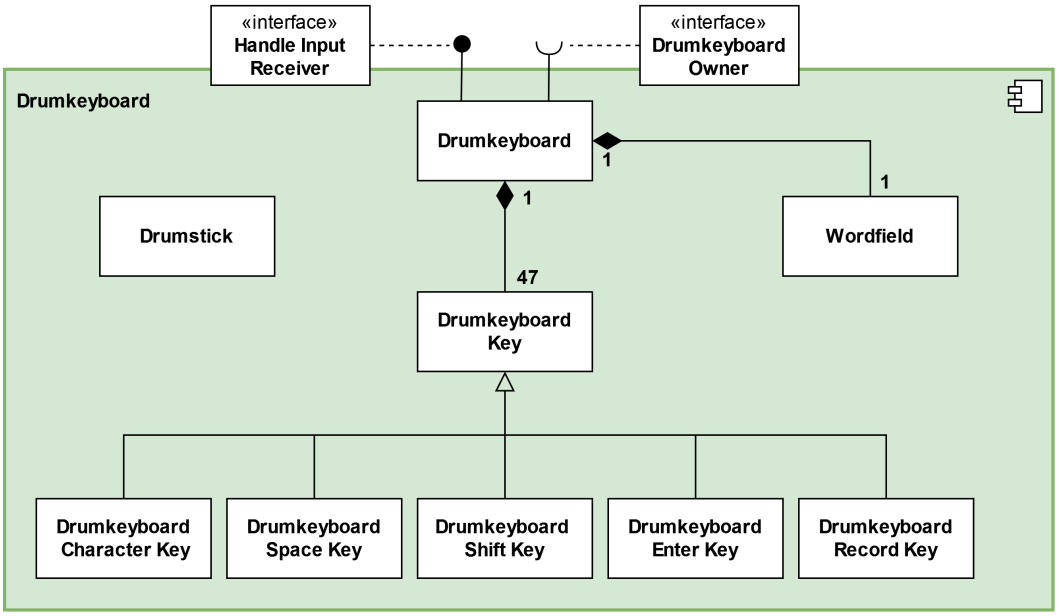


Abbildung 4.8: Die Komponente *Drumkeyboard* [Eigene Grafik].

4.4.3 Handle

Wie aus dem vorherigen Abschnitt bereits hervorgeht, stellt die Komponente *Handle* die Implementierung der Handles dar, über die sich die Höhe von Drumkeyboard und Korrekturinterface einstellen lässt. Sie enthält nur eine Klasse und wird deshalb nicht gesondert grafisch dargestellt. Über das definierte Interface *Handle Input Receiver* werden die Besitzer einer Handle informiert, wenn eine Bewegung stattfinden soll. Somit besteht keine externe Abhängigkeit und die Handles können universell eingesetzt und an beliebige andere Komponenten angebunden werden.

4.4.4 Textfield

Die Komponente *Textfield* bildet die Umsetzung der zwei verschiedenen Korrekturinterfaces. Beide setzen sich aus einem *Textfield Wrapper* und einem oder mehreren *Textfield*-Instanzen zusammen (Abb. 4.9). Dieser zweistufige Aufbau dient dazu, den Zugriff von außen einheitlich zu gestalten, obwohl intern ein großer Unterschied besteht, da

das Korrekturinterface für Freitextfragen ein einziges, großes Textfeld besitzt, das Korrekturinterface für Aufzählungsfragen hingegen aus mehreren Textfeldern besteht. Eine verwendende Komponente muss somit nur die Klasse *Textfield Wrapper* kennen, nicht aber die übrigen Klassen. Tatsächlich instanziiert wird entweder die Subklasse *Scrollable Textfield Wrapper* für Freitextfragen oder *Reorder List Textfield Wrapper* für Aufzählungsfragen. Beide arbeiten mit der Klasse *Textfield*, die ein einzelnes Textfeld repräsentiert und den Großteil der Funktionalität eines solchen beinhaltet, z.B. die Berührung des Drumsticks zum Verändern der Cursorposition und dem Markieren von Text, das Hinzufügen oder Löschen von Zeichen und das Auslösen eines Events, wenn einer der Buttons gedrückt wurde. Da sich das Verhalten je nach Korrekturinterface zum Teil aber unterscheidet, findet hier erneut eine Aufteilung in zwei Subklassen statt: *Scrollable Textfield* implementiert ein alleinstehendes Textfeld, in dem gescrollt werden kann. *Reorder List Textfield* stellt einen Knoten in einer doppelt-verketteten Liste von Textfeldern dar und kennt seinen Vorgänger (das Textfeld darüber) und Nachfolger (das Textfeld darunter), mit denen es die Plätze tauschen kann, wenn die Liste umsortiert werden soll.

Genau wie das Drumkeyboard implementieren beide Wrapper das Interface *Handle Input Receiver*, um auf eine Interaktion mit den zugehörigen Handles reagieren zu können und die eigene Position zu verändern.

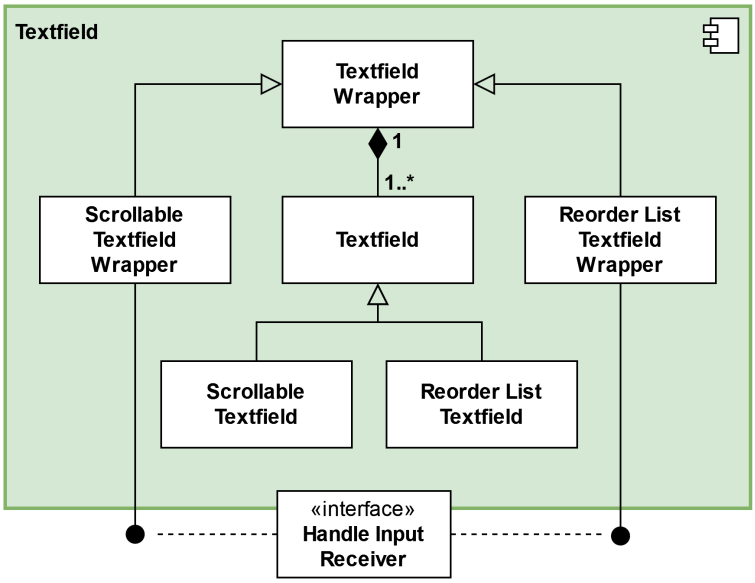


Abbildung 4.9: Die Komponente *Textfield* [Eigene Grafik].

4.4.5 Speech Recognition

Speech Recognition stellt die letzte Komponente des *Text Entry VR Plugin* dar. Sie ist unabhängig von den übrigen Komponenten des Plugins und bietet die Möglichkeit der Spracherkennung. Dabei handelt es sich im Grunde lediglich um eine Kapselung der Kommunikation mit dem Transkriptionsserver, für welche die einzige Klasse dieser Komponente das von UE4 bereitgestellte *WebSocket*-Modul verwendet.

4.4.6 Soundwalk Form

Die Komponente *Soundwalk Form* des Ist-Systems beinhaltet die Logik des Fragebogens. Abb. 4.10 zeigt ihre interne Struktur. Ihre Hauptklasse *Soundwalk Form* bildet den Rahmen des Fragebogens und nimmt den Input der Controller entgegen. Der Inhalt, d.h. die konkreten Fragen, werden über Implementierungen des Interfaces *Soundwalk Form Question Widget* abgebildet, welche die bereits vorhandenen Fragen, die über einen Slider oder eine diskrete Auswahl beantwortet werden, sowie die Überschriften zu Beginn eines neuen Frageblocks darstellen. Zur Umsetzung der Freitext- und Aufzählungsfragen wurde eine weitere Implementierung *Soundwalk Form Text Question Widget* hinzugefügt, die in Abb. 4.10 grün gefärbt ist. Sie ist der Hauptanknüpfungspunkt für das *Text Entry VR Plugin* und übernimmt das Erzeugen und Entfernen des Korrekturinterfaces und des Drumkeyboards und implementiert das entsprechende Interface *Drumkeyboard Owner*. Darüber hinaus bietet sie eine Funktion zum Auslesen des eingegebenen Textes an, damit dieser als Antwort verschickt werden kann.

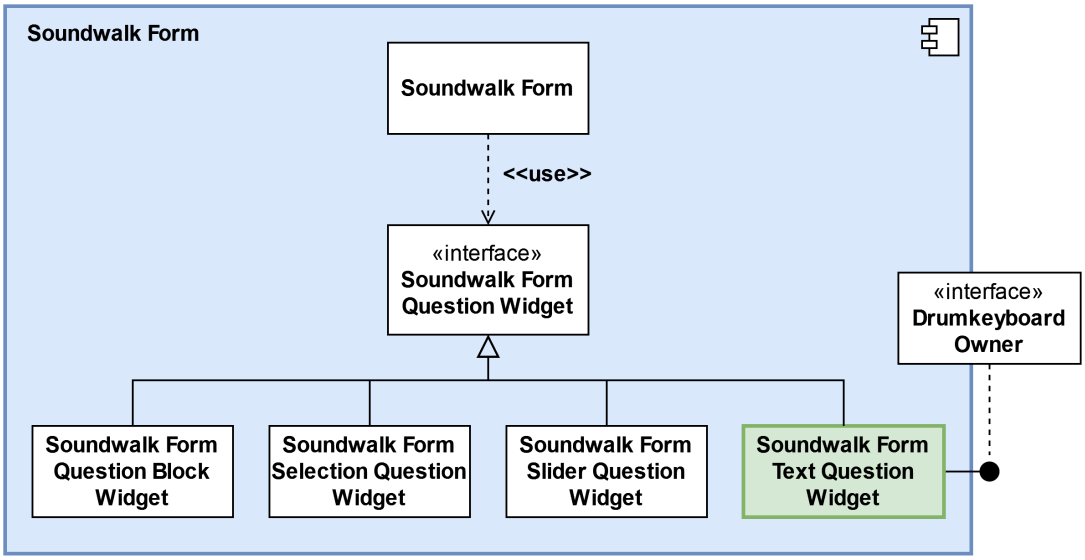


Abbildung 4.10: Die Komponente *Soundwalk Form* [Eigene Grafik].

4.4.7 Motion Controller

Ebenfalls angepasst wurde die Komponente *Motion Controller* des Ist-Systems. Darin befinden sich Klassen für die virtuellen Abbilder der VR-Controller, die sich je nach Modell unterscheiden. In dieser Arbeit wurde nur die Klasse für die Controller der eingesetzten *Valve Index* betrachtet. Sie wurde dahingehend erweitert, dass die Drumsticks erzeugt und entfernt und die Bedienhinweise für die Texteingabe angezeigt oder ausgeblendet werden können.

4.4.8 Player Control

Die Komponente *Player Control* erfüllt eine Vielzahl von Aufgaben im Ist-System, wie die initiale Verarbeitung der Nutzereingaben und die allgemeine Steuerung der Simulation. Zu letzterer gehört auch die Erzeugung des Fragebogens und das Auslesen der Antworten, weshalb eine gegenseitige Abhängigkeit mit der Komponente *Soundwalk Form* besteht. Das Setup zu Beginn der Simulation findet ebenfalls zum Großteil hier statt und wurde für diese Arbeit dahingehend erweitert, dass das Objekt zur Spracherkennung erzeugt wird, auf das alle Instanzen von *Soundwalk Form Text Question Widget* global zugreifen.

4.4.9 UE Soundwalk API

Zum Einlesen der Daten des aktuellen Soundwalks und den Inhalten des Fragebogens sowie zum Abschicken der Antworten verwendet *Player Control* die Komponente *UE Soundwalk API*, welche die Kommunikation mit dem Backend-Server übernimmt und dafür vom *HTTP*-Modul der UE4 Gebrauch macht. Um die neu eingeführten Fragetypen zu unterstützen, wurden hier entsprechende Anpassungen vorgenommen.

4.4.10 Transcription Server

Wie in Kapitel 4.2.2 beschrieben, läuft der Transkriptionsserver als eigener Prozess und ist deshalb unabhängig von der Hauptapplikation in UE4. Er ist vom Umfang her sehr klein und besteht nur aus einer einzigen Komponente *Transcription Server*. Das Modell für die Spracherkennung wird über das Python-Modul *whisper* eingebunden und zur Umsetzung des WebSocket-Servers dient das *websockets*-Modul. Um die Eingabe mittels Spracherkennung und Drumkeyboard konsistent zu halten, werden alle Zeichen aus dem Ergebnis der Transkription entfernt, die nicht in FA4 spezifiziert sind und sich daher auch nicht über das Drumkeyboard eingeben lassen. Dafür wird das Modul *re* für reguläre Ausdrücke verwendet.

4.5 Zusammenfassung

In diesem Kapitel wurde der Entwurf der Lösung dargelegt und dafür zunächst der Hardware- und Software-Kontext betrachtet, bestehend aus dem *Valve Index*-HMD und einem UE4-Projekt. Im Anschluss wurden verschiedene Optionen für die Spracherkennung diskutiert und aufgrund der guten Rechtschreibung schließlich *Whisper* von OpenAI ausgewählt. Es wurde zudem die Anbindung der Spracherkennung über einen WebSocket-Server erläutert. Das fachliche Design der Lösung wurde ausführlich beschrieben und dabei auf die drei Hauptbestandteile des Drumkeyboards, der Spracheingabe und der zwei unterschiedlichen Korrekturinterfaces eingegangen, sowie ihrer Integration in den Fragebogen. Abschließend wurde ein Überblick über die Softwarearchitektur des Systems gegeben und die verschiedenen Komponenten und ihre Beziehungen wurden genauer erläutert. Ergebnis des Entwurfs sind ein Transkriptionsserver und ein UE4-Plugin, welches die Funktionalität der Texteingabe bündelt und auf einfache Weise in das bestehende System eingebunden werden kann.

5 Benutzertest

In Kapitel 3 wurden drei verschiedene Arten der Texteingabe in VR ausgewählt und es wurde die Anforderung formuliert, diese miteinander zu vergleichen. Dafür wurden auf Basis der in Kapitel 4 vorgestellten Implementierung Benutzertests durchgeführt, welche in diesem Kapitel beschrieben werden. Insbesondere sollte damit die Hypothese überprüft werden, dass eine Kombination aus Spracheingabe und Drumkeyboard bessere Ergebnisse erzielt als nur eine der beiden Eingabemethoden allein. Kapitel 5.1 erläutert die Durchführung der Benutzertests und Kapitel 5.2 wertet die Ergebnisse aus. Kapitel 5.3 widmet sich der Kritik und Kapitel 5.4 fasst die wesentlichen Ergebnisse zusammen.

5.1 Durchführung

In diesem Abschnitt wird zunächst in Kapitel 5.1.1 auf die Probanden eingegangen, die an den Tests teilgenommen haben. Anschließend werden die Rahmenbedingungen (Kapitel 5.1.2), der Ablauf (Kapitel 5.1.3) und der auszufüllende Fragebogen (Kapitel 5.1.4) beschrieben.

5.1.1 Probanden

Insgesamt nahmen 16 Probanden im Alter zwischen 22 und 38 Jahren an den Benutzertests teil. Bis auf eine Ausnahme handelt es sich bei allen von ihnen um Informatikstudierende. Der Altersdurchschnitt beträgt 27 Jahre und alle Teilnehmer gaben Deutsch als Muttersprache an. Bezüglich der Erfahrung im Umgang mit VR gab ein Proband an, es mindestens einmal im Monat zu verwenden. Alle anderen Probanden verwendeten VR noch seltener oder nie.

Die Probanden wurden im Vorhinein darüber aufgeklärt, dass ihre Ergebnisse anonymisiert in dieser Arbeit veröffentlicht werden.

5.1.2 Rahmenbedingungen

Die Benutzertests vergleichen die Eingabe eines aus mehreren Sätzen bestehenden Textes mithilfe drei unterschiedlicher Eingabeverfahren, die im Folgenden als *Modus* bezeichnet werden: *Sprache*, *Drumkeyboard* und *Multimodal*. Dafür wurde die in Kapitel 4 beschriebene Implementierung und im Speziellen das Korrekturinterface für Freitextfragen verwendet, das aus einem großen Textfeld besteht und für längere Texte geeignet ist. Die Ergebnisse lassen sich aber auch auf die Eingabe einer sortierten Aufzählung übertragen, da die reine Texteingabe bis auf minimale Unterschiede identisch ist (vgl. Kapitel 4).

Der einzugebende Text enthält Umlaute, Ziffern und Sonderzeichen, um den gesamten unterstützten Zeichenvorrat zu repräsentieren und einem typischen deutschen Text zu entsprechen. Er lautet: „Das Universum fasziniert uns Menschen seit jeher mit seiner unendlichen Weite und seinen mysteriösen Geheimnissen. Die Milchstraße, unsere Heimatgalaxie, hat ein geschätztes Alter von etwa 13,5 Milliarden Jahren. Unsere Sonne, ein gewöhnlicher Stern, befindet sich in den Außenbereichen dieser prächtigen Galaxie. Sie enthält 99,86 % der Masse unseres Sonnensystems.“

Zur Durchführung der Tests wurden Korrekturinterface und Drumkeyboard leicht angepasst bzw. erweitert: Das Textfeld wurde in der Höhe um die Hälfte verkleinert und darunter ein zweites Textfeld hinzugefügt, welches den aktuell einzugebenden Satz anzeigt. Passend dazu erhielt das Drumkeyboard eine weitere Taste (im Folgenden als OK-Taste bezeichnet), über die die Eingabe des Satzes bestätigt werden kann, woraufhin der nächste einzugebende Satz angezeigt wird. Sie ist durch ihre grüne Farbe hervorgehoben und mit einem Haken gekennzeichnet. Die nicht benötigte Enter-Taste wurde entfernt, da der einzugebende Text keine Absätze enthält. Abb. 5.1 zeigt den veränderten Aufbau.

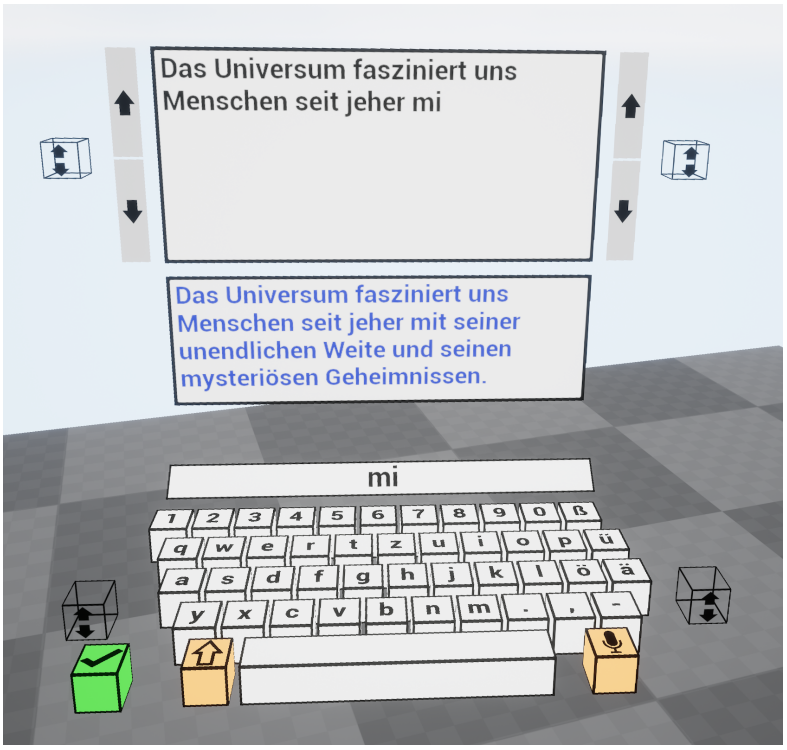


Abbildung 5.1: Angepasstes Drumkeyboard und Korrekturinterface für die Benutzertests [Eigene Grafik].

Je nach Modus sind unterschiedliche Tasten des Drumkeyboards deaktiviert. Im Modus *Sprache* erfolgt die Eingabe allein durch Einsprechen, dementsprechend sind nur die Aufnahme- und OK-Taste verwendbar. Im Modus *Drumkeyboard* hingegen sind alle Tasten bis auf die Aufnahme-Taste aktiviert, sodass keine Spracheingabe erfolgen kann. Im Modus *Multimodal* sind alle Tasten aktiviert: Text kann sowohl eingesprochen als auch eingetippt werden. Das Korrekturinterface ist in allen Modi uneingeschränkt verwendbar, d.h. die Cursor-Position kann verändert werden, es kann Text markiert und ggf. gescrollt werden. Die Backspace-Taste des Controllers kann ebenfalls jederzeit verwendet werden.

5.1.3 Ablauf

Der Ablauf eines Benutzertests war für alle Probanden gleich: Zu Beginn wurde die Nutzung des Systems vorgeführt, die Bedienung erklärt und die zu erfüllende Aufgabe beschrieben (Eingabe desselben Textes auf drei verschiedene Arten, möglichst schnell und

fehlerfrei). Anschließend konnte der Proband das HMD richtig einstellen, die Höhe des Drumkeyboards und der Textfelder anpassen und das System selbst kurz ausprobieren. Dann erfolgte die Eingabe des Textes, zunächst im Modus *Sprache*, dann im Modus *Drumkeyboard* und zuletzt im Modus *Multimodal*. Dabei wurde jeweils die Zeit vom ersten Tastenanschlag bis zum Bestätigen des letzten Satzes über die OK-Taste gemessen. Im Anschluss füllte der Proband einen Fragebogen aus (siehe Kapitel 5.1.4). Insgesamt dauerte ein Benutzertest ca. 30 Minuten.

Den Probanden wurde im Vorhinein mitgeteilt, in der Abwägung zwischen Geschwindigkeit und Korrektheit den Fokus auf eine möglichst fehlerfreie Eingabe zu legen. Weil im Modus *Sprache* die Korrektur aber nur bedingt möglich ist, da auch mehrmaliges Einsprechen eines Wortes oft nicht zum gewünschten Ergebnis führt, wurden die Probanden gebeten, in diesem Modus nicht mehr als zwei Korrekturversuche für eine bestimmte Stelle im Text zu verwenden und einen Fehler ggf. stehen zu lassen.

5.1.4 Fragebogen

Der Fragebogen dient der Ermittlung der subjektiven Einschätzung der Probanden (vgl. FA10). Es sollen die drei Modi hinsichtlich der in Kapitel 2 vorgestellten Bewertungskriterien miteinander verglichen werden und überprüft werden, inwieweit die in Kapitel 3 aufgeführten Anforderungen bezüglich der Nutzerfreundlichkeit erfüllt wurden. Für jeden Modus wurden dieselben Fragen beantwortet, wobei der Proband jeweils aus diskreten Antwortmöglichkeiten auswählen konnte. Die Fragen und möglichen Antworten sind in Tabelle 5.1 aufgeführt.

5.2 Auswertung

Um die verschiedenen Modi miteinander zu vergleichen, wurden sowohl Messungen der Eingabe- und Fehlerrate vorgenommen als auch die subjektive Einschätzung der Probanden ermittelt. Erstere werden in Kapitel 5.2.1 ausgewertet und letztere in Kapitel 5.2.2.

Frage	**Antwortmöglichkeiten**
Die Geschwindigkeit, mit der ich Texte eingeben konnte, war:	Sehr gut Gut Akzeptabel Schlecht Sehr schlecht
Die Fehleranfälligkeit der Texteingabe war:	Sehr gering Gering Akzeptabel Hoch Sehr hoch
Die Texteingabe war intuitiv und einsteigertauglich:	Stimme voll zu Stimme eher zu Stimme eher nicht zu Stimme gar nicht zu
Die Texteingabe war körperlich anstrengend oder unangenehm:	Stimme voll zu Stimme eher zu Stimme eher nicht zu Stimme gar nicht zu
Die Texteingabe war mental anstrengend oder unangenehm:	Stimme voll zu Stimme eher zu Stimme eher nicht zu Stimme gar nicht zu
Alles in allem empfand ich diese Art der Texteingabe als:	Sehr gut Gut Akzeptabel Schlecht Sehr schlecht

Tabelle 5.1: Fragebogen der Benutzertests.

5.2.1 Auswertung der Eingabe- und Fehlerrate

Während der Eingabe wurde die benötigte Zeit in Sekunden gemessen und daraus, wie in Kapitel 2.1.1 beschrieben, die Eingabegeschwindigkeit in wpm berechnet (Abb. 5.2). Die Fehlerrate wurde in Form der MSD ER ermittelt (siehe Kapitel 2.1.2) und wird in (Abb. 5.3) gezeigt. In Kapitel 2.1.2 wurde die Total ER als die mächtigste Fehlermetrik beschrieben, jedoch lässt sie sich mit der bestehenden Implementierung nicht messen, da es nicht möglich ist, die Anzahl korrigierter Zeichen zu ermitteln. Dies wurde erst nach Fertigstellung der Implementierung festgestellt und deshalb stattdessen die MSD ER gewählt, auch wenn diese nur etwas über die Fehlerhaftigkeit des fertig eingegebenen Textes aussagt und nicht über den eventuellen Korrekturaufwand. Der Korrekturaufwand ist aber indirekt in der Eingabegeschwindigkeit enthalten und war zudem Teil des Fragebogens, sodass dennoch eine Aussage dazu getroffen werden kann.

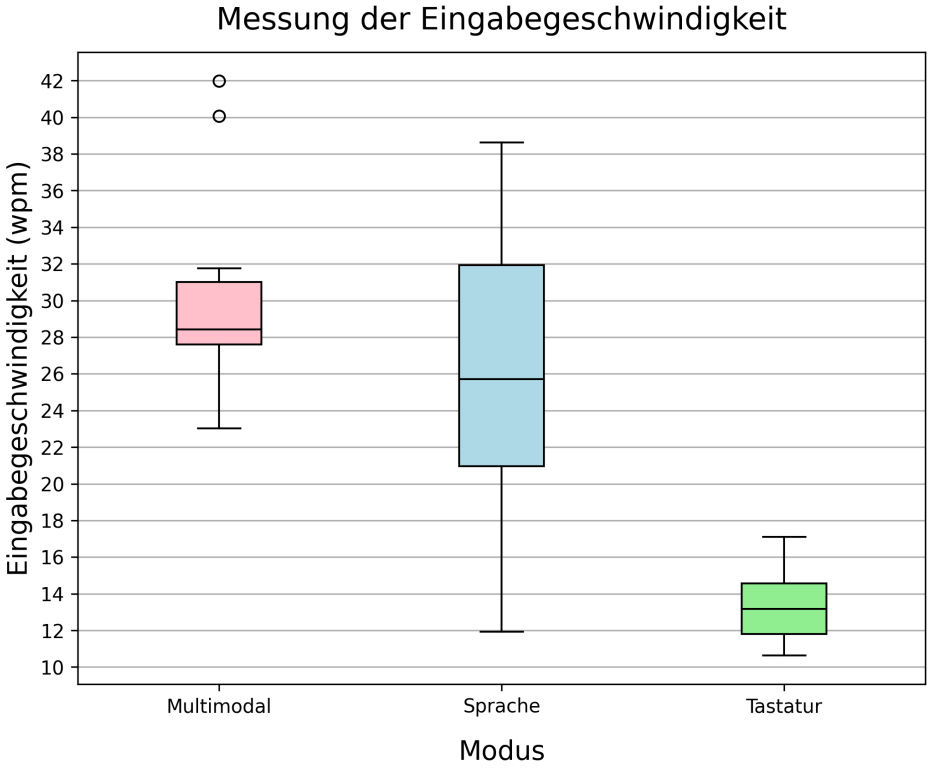


Abbildung 5.2: Messung der Eingabegeschwindigkeit [Eigene Grafik].

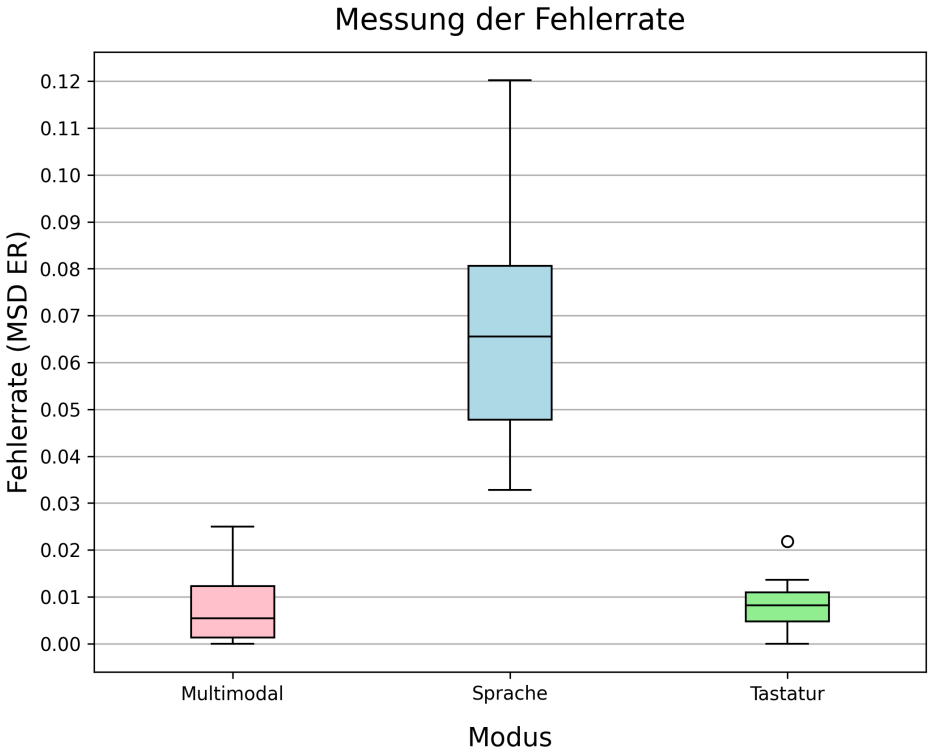


Abbildung 5.3: Messung der Fehlerrate [Eigene Grafik].

Im Modus *Sprache* liegt eine große Streuung der Ergebnisse vor. Die Geschwindigkeit reicht von 12 bis 39 wpm und die Fehlerrate von 0,03 bis 0,12. Diese große Varianz ist darauf zurückzuführen, dass die Probanden sehr unterschiedlich schnell und deutlich sprachen. Manche sprachen demonstrativ langsam und betont, andere ganz normal und natürlich. Entgegen den Erwartungen lieferte das natürliche Sprechen meist bessere Ergebnisse. Dadurch nahm bei Probanden, die besonders langsam sprachen, nicht nur das Einsprechen an sich mehr Zeit in Anspruch, sondern auch die Korrektur, da mehr Fehler vorhanden waren. Diese ließen sich durch erneutes Einsprechen nur schlecht korrigieren und oft wurden vergeblich mehrere Versuche aufgewendet.

Bei der Eingabe allein über die Tastatur liegen sowohl Geschwindigkeiten als auch Fehlerraten der Probanden viel näher zusammen, mit Ergebnissen von 11 bis 17 wpm und 0 bis 0,02 MSD ER. Auch hier gab es Unterschiede in der Bedienung: Während manche Probanden sehr vorsichtig und bedacht tippten und dadurch einerseits langsam waren, andererseits kaum Fehler machten, schlugen andere die Tasten deutlich schneller an, trafen dabei aber häufig versehentlich zwei Tasten auf einmal, sodass viele Korrekturen

nötig waren. Im Endeffekt benötigten beide Gruppen dadurch ähnlich viel Zeit für die Eingabe. Die Fehlerrate ist durchweg sehr gering, da die Korrektur hier sehr viel verlässlicher funktionierte als bei der Spracheingabe. Die wenigen vorhandenen Fehler wurden vermutlich übersehen, da die Probanden gebeten wurden, den Text exakt so einzugeben wie das Original.

Mit der multimodalen Methode konnten alle Probanden den Text am schnellsten Eingeben. Sie machten intuitiv von den Stärken der zwei Eingabeverfahren Gebrauch, indem sie einen Satz zunächst einsprachen und anschließend die Fehler (falls vorhanden) über das Drumkeyboard korrigierten. Damit ließ sich eine hohe Geschwindigkeit von 23 bis 42 wpm und eine niedrige Fehlerrate zwischen 0 und 0,03 erzielen. Die Geschwindigkeiten sind auch hier relativ breit gefächert, was auf den Einfluss der Spracheingabe zurückzuführen ist. Die Fehlerraten wiederum sind sehr ähnlich zum Modus *Tastatur*, im Mittel sogar etwas geringer.

Wie erwartet, war das Einsprechen um einiges schneller aber auch fehleranfälliger als das Tippen, sodass die Kombination aus beiden Verfahren insgesamt zu den besten Ergebnissen führte. Die Performance des Drumkeyboards allein war etwas schlechter als in der Literatur (Kapitel 2.2.2). Allerdings hatte das Drumkeyboard in den untersuchten Studien auch keine Ziffern, Sonderzeichen oder Umlaute und der einzugebende Text war dementsprechend ebenfalls weniger komplex. Die Ergebnisse für die multimodale Eingabe sind hingegen vergleichbar mit denen aus der Literatur (Kapitel 2.2.4).

5.2.2 Auswertung des Fragebogens

Im Fragebogen gaben die Probanden zunächst eine subjektive Einschätzung der Eingabegeschwindigkeit und Fehleranfälligkeit der drei Modi an. Abb. 5.4 und 5.5 zeigen die Ergebnisse.

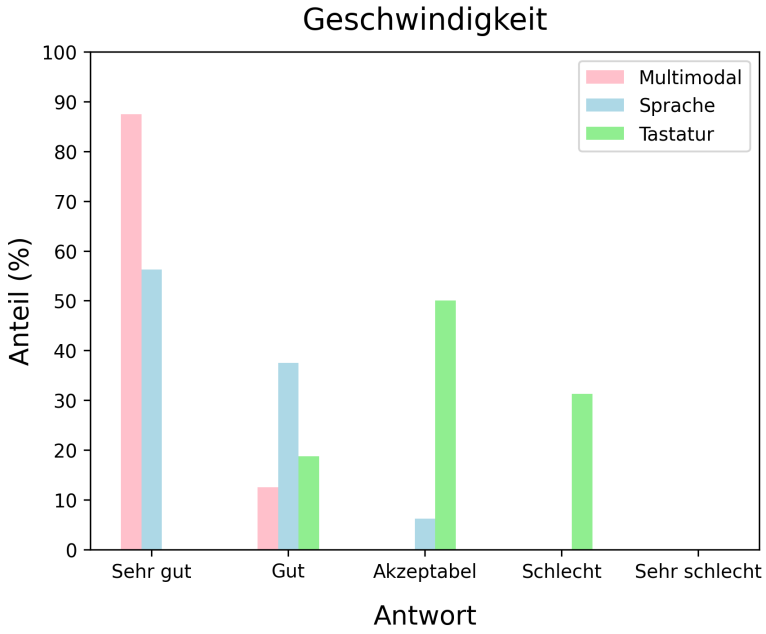


Abbildung 5.4: Einschätzung der Eingabegeschwindigkeit [Eigene Grafik].

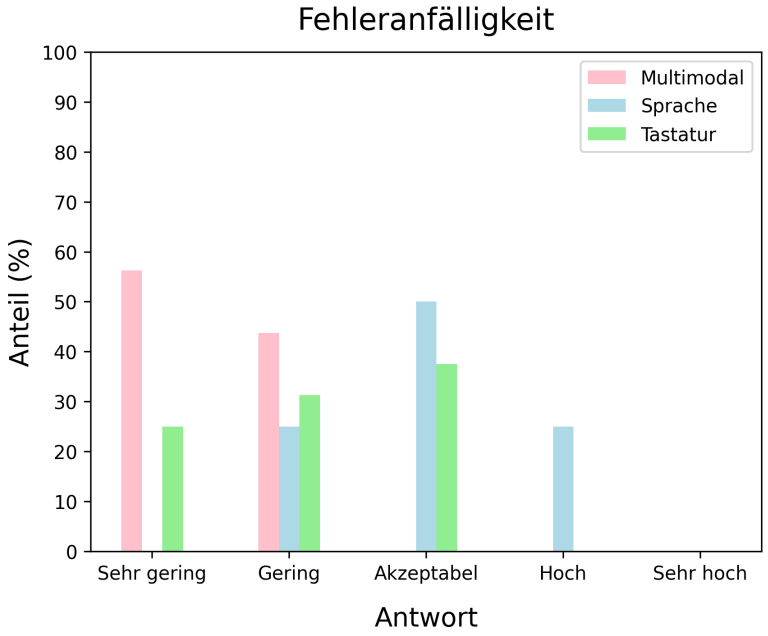


Abbildung 5.5: Einschätzung der Fehleranfälligkeit [Eigene Grafik].

Die Einschätzungen der Geschwindigkeit decken sich mit den Messergebnissen: Die multimodale Eingabe wurde am besten bewertet, die reine Spracheingabe etwas schlechter und die ausschließliche Nutzung des Drumkeyboards deutlich schlechter. Die Einschätzungen der Fehleranfälligkeit der Modi *Multimodal* und *Sprache* spiegeln ebenfalls die gemessenen Werte wider. Interessant ist, dass der Modus *Tastatur* als deutlich fehleranfälliger als der Modus *Multimodal* eingestuft wurde, obwohl die tatsächlichen Fehlerraten sehr ähnlich sind.

Als nächstes sollten die Probanden angeben, ob sie die unterschiedlichen Verfahren intuitiv und einsteigertauglich fanden (Abb. 5.6). Für die Modi *Multimodal* und *Sprache* gab es von allen Probanden Zustimmung, größtenteils sogar volle Zustimmung. Der Modus *Tastatur* schnitt etwas schlechter ab und ein kleiner Teil der Probanden stimmte eher nicht zu. Das entspricht den Erwartungen, da es sich bei der Spracheingabe um eine sehr natürliche Art der Texteingabe handelt, die auch aus dem Alltag bekannt ist. Ein Drumkeyboard hingegen verwendeten alle Probanden zum ersten Mal, und auch wenn einige Parallelen zur vertrauten PC- oder Smartphone-Tastatur bestehen, ist die Art der Interaktion doch grundlegend anders und zunächst ungewohnt.

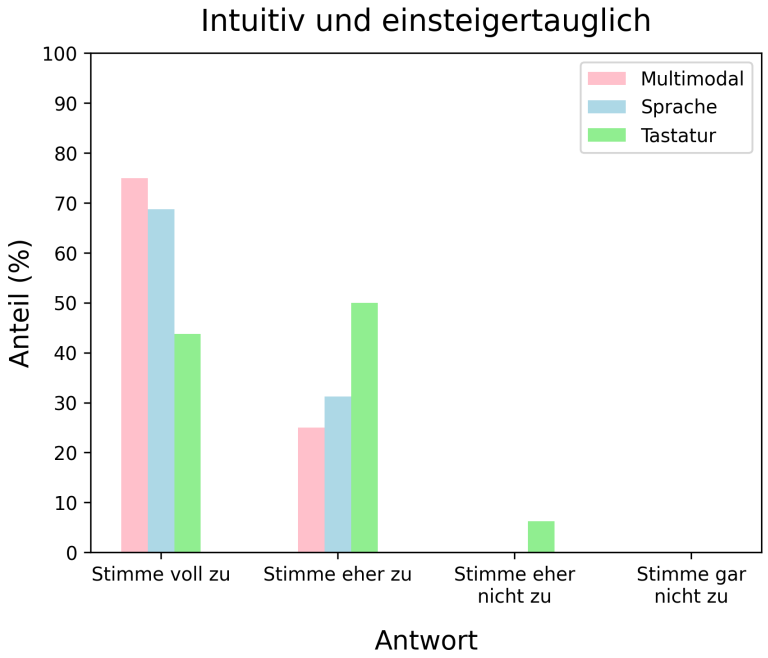


Abbildung 5.6: Einschätzung der Einsteigertauglichkeit [Eigene Grafik].

Es folgte die Einschätzung der Ergonomie, welche für den körperlichen und den mentalen Aspekt sehr ähnlich ausfiel (Abb. 5.7 und 5.8). Die Modi *Sprache* und *Multimodal* erhielten diesbezüglich ausschließlich und der Modus *Tastatur* größtenteils positive Bewertungen. Da die Nutzung der Spracheingabe nur zwei Tastendrücke zum Starten und Beenden benötigt, war damit zu rechnen, dass sie als kaum oder gar nicht körperlich belastend eingestuft wird. Dennoch war das Auswählen und Bearbeiten des Textes über das Korrekturinterface vermutlich mit einer gewissen körperlichen und mentalen Anstrengung verbunden. Das Tippen auf dem Drumkeyboard erfordert sowohl einige Bewegungen in den Armen und Handgelenken als auch ein gewisses Maß an Konzentration, um genau zu zielen und die richtigen Tasten zu treffen, was erklärt, warum die Ergonomie von manchen Probanden negativ bewertet wurde.

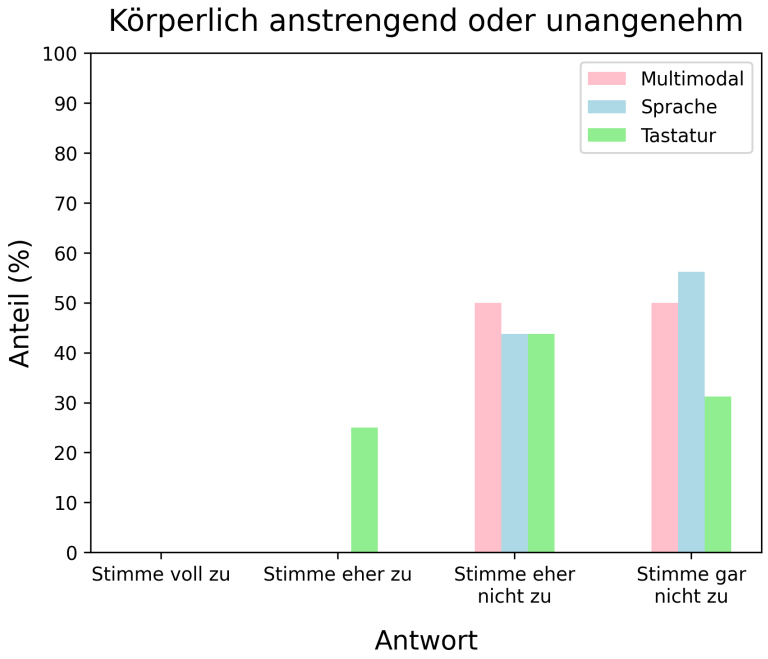


Abbildung 5.7: Einschätzung der körperlichen Ergonomie [Eigene Grafik].

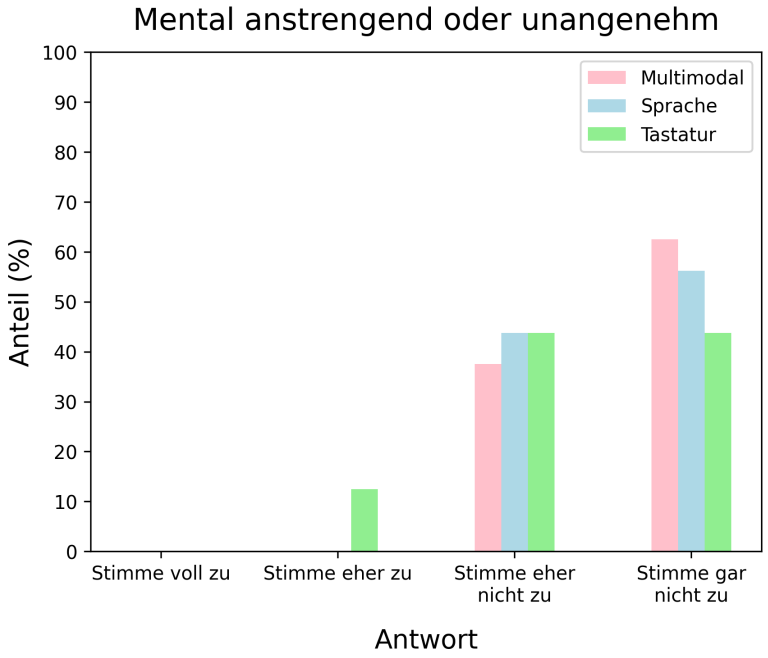


Abbildung 5.8: Einschätzung der mentalen Ergonomie [Eigene Grafik].

Zuletzt sollte eine Gesamtbewertung für die drei Modi abgegebenen werden, deren Resultate in Abb. 5.9 dargestellt werden. Für die reine Spracheingabe besteht Uneinigkeit, die Angaben reichen von „Sehr gut“ bis „Schlecht“. Erneut ist dies wohl darauf zurückzuführen, dass die Spracherkennung aufgrund unterschiedlicher Sprechweisen für manche Probanden deutlich besser funktionierte als für andere. Die ausschließliche Nutzung des Drumkeyboards zur Eingabe wurde von allen Probanden als akzeptabel oder besser eingestuft, wobei es nur eine sehr gute Bewertung gab. Die beste Bewertung erhielt die multimodale Eingabe, sie wurde von der Mehrheit der Probanden als „Sehr gut“ und von den übrigen als „Gut“ empfunden.

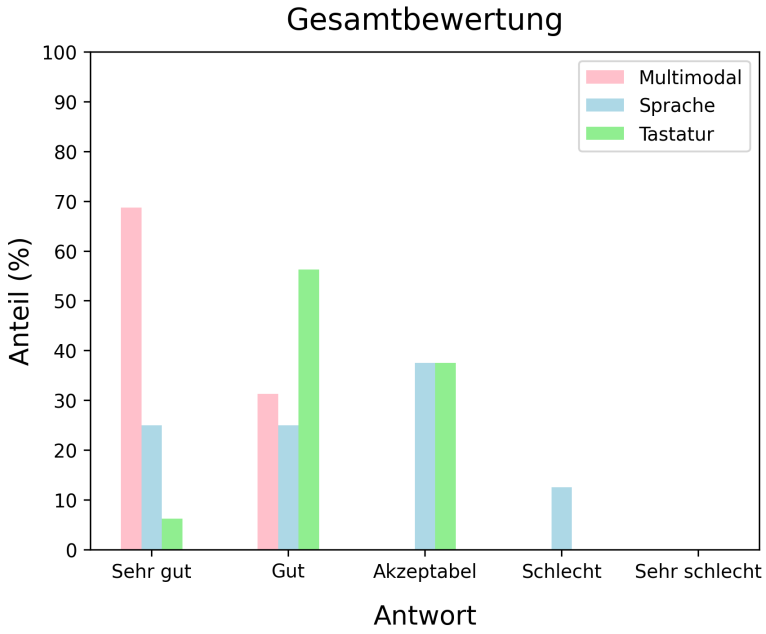


Abbildung 5.9: Gesamtbewertung der drei Modi [Eigene Grafik].

5.3 Kritik

Verschiedene Aspekte der Benutzertests können kritisiert werden, diese werden in Kapitel 5.3.1 aufgeführt. Kapitel 5.3.2 betrachtet mögliche Kritiken an der Implementierung.

5.3.1 Kritik der Benutzertests

Die Rahmenbedingungen und Durchführung der Benutzertests beinhalten mögliche Fehlerquellen. Zunächst ist anzumerken, dass die Ergebnisse der Tests aufgrund der geringen Teilnehmerzahl nicht statistisch aussagekräftig sind. Einzelne Probanden, deren Messwerte oder Antworten stark von der Mehrheit abwichen, hatten einen nicht geringen Einfluss auf das Gesamtbild und konnten es ggf. verzerren. Außerdem waren die Probanden relativ jung, überwiegend technikaffin und mit dem Layout der Tastatur vertraut. Deshalb fielen die Ergebnisse wahrscheinlich besser aus, als wenn ein repräsentativeres Abbild der Bevölkerung gewählt worden wäre.

Des Weiteren kann der Ablauf der Tests kritisiert werden. Dadurch, dass die Reihenfolge der Modi immer gleich war, hatten die Probanden bei der multimodalen Eingabe bereits etwas Übung im Umgang mit der Spracheingabe und dem Drumkeyboard aus den ersten beiden Durchläufen. Zusätzlich war der einzugebende Text immer derselbe und konnte mit der Zeit erlernt werden. Es ist davon auszugehen, dass die Ergebnisse dadurch ein wenig zugunsten der multimodalen Eingabe verändert wurden.

Zuletzt gab es Probleme mit der verwendeten Hardware. Bei einem Probanden kam es mehrmals zu Trackingfehlern, wodurch das virtuelle Abbild des linken Controllers zu weit nach vorne versetzt war und nicht mit der tatsächlichen Position übereinstimmte, was das Zielen mit dem Drumstick erschwerte. Auch wiesen einige Probanden darauf hin, dass die Sehschärfe über das HMD nur unzureichend eingestellt werden konnte und das Bild etwas unscharf war. Möglicherweise beeinträchtigte dies ihre Performance.

5.3.2 Kritik der Implementierung

Die Benutzertests zeigten einige Verbesserungsmöglichkeiten der Implementierung auf. Ein Großteil der Probanden hatte Probleme, längere Textabschnitte zu markieren, da die Kugel am Ende des Drumsticks dafür sehr präzise innerhalb des Textfelds entlanggeführt werden muss. Oft wurde versehentlich durch das Textfeld hindurchgestochen und beim Zurückziehen der Hand die Markierung gelöscht. Das ließe sich beheben, indem der gesamte Drumstick zum Markieren genutzt werden kann, nicht nur das Ende.

Der geringe Abstand zwischen den Tasten des Drumkeyboards führte ebenfalls zu Problemen. Wurde eine Taste nicht mittig, sondern nur am Rand getroffen, kam es oft vor, dass die Taste daneben ebenfalls ungewollt angeschlagen wurde. Eventuell sollten die Tasten deshalb weiter auseinander platziert werden. Das würde gleichzeitig aber auch die Ausdehnung des Drumkeyboards vergrößern und ein längeres Ausstrecken der Arme erfordern, um weiter entfernte Tasten zu erreichen. Eine Verlängerung der Drumsticks könnte dies wiederum ausgleichen.

Bei zwei Probanden kam es vor, dass sie das Korrekturinterface bedienten und dabei aus Versehen mit der anderen Hand das Drumkeyboard berührten und Zeichen eingaben. Hier gäbe es mind. zwei Optimierungsmöglichkeiten: Zum einen könnte das Drumkeyboard deaktiviert werden, während eine Interaktion mit dem Korrekturinterface stattfindet. Keiner der Probanden nutzte beides gleichzeitig, sodass davon auszugehen ist, dass hier kein Konflikt entstehen würde. Zum anderen könnte beim Drücken einer Taste überprüft

werden, ob diese sich innerhalb des Sichtfelds befindet, und ein Anschlagen nur dann stattfinden, wenn die Taste auch angesehen wird.

Weiteres Feedback richtete sich an die Nutzung der Controllerbuttons. Drei der Probanden gaben an, dass sie die Shift-Taste auf dem linken Trigger intuitiver empfunden hätten als die Backspace-Taste und wünschten sich, beide Tasten über einen Trigger bedienen zu können. Auch das Starten und Stoppen der Spracheingabe mittels eines Buttons wurde vorgeschlagen. Hier besteht Potenzial zur Optimierung. Es ist jedoch zu beachten, dass eine ausgiebigere Verwendung der Controller eventuell auch die Bedienung komplizierter und weniger einsteigertauglich machen könnte.

Die Spracherkennung wurde von manchen Probanden aufgrund vieler Fehler bei der Transkription kritisiert. Möglicherweise könnte sich der Einsatz eines besseren Sprachmodells lohnen, auch wenn sich die Verarbeitungszeit dadurch erhöhen würde. Zusätzlich könnte das Ergebnis der Transkription durch einen Spellchecker verbessert werden.

Ein paar Aspekte der Implementierung wurden auch besonders positiv hervorgehoben. So habe sich das schmale Textfeld über dem Drumkeyboard, welches das aktuelle Wort anzeigt, als sehr hilfreich erwiesen. Auch die Einstellmöglichkeit der Höhe von Drumkeyboard und Korrekturinterface wurden gelobt.

5.4 Zusammenfassung

In Benutzertests wurden die Eingabe über Sprache, über das Drumkeyboard und über die Kombination aus beiden Verfahren untersucht und miteinander verglichen. Trotz der angemerkten Kritik an der geringen Teilnehmerzahl, zeigen sowohl die Messungen von Eingabe- und Fehlerrate als auch die subjektiven Einschätzungen der Probanden eine sehr hohe Evidenz für die Hypothese, dass die multimodale Eingabe ein deutlich besseres Verfahren zur Texteingabe in VR darstellt, als wenn nur eine der beiden Methoden allein verwendet wird. Mit diesem Ansatz könnten auch die Anforderungen NFA1, NFA2 und NFA3 an die Nutzerfreundlichkeit und Effizienz der Lösung erfüllt sein. Dies wäre in einer groß angelegten Benutzerstudie unter Berücksichtigung der methodischen und technischen Verbesserungen zu verifizieren.

6 Fazit

In dieser Arbeit wurde eine Lösung für die Texteingabe in VR entwickelt. Dafür wurden die in der Literatur vorgeschlagenen Eingabeverfahren untersucht und anhand verschiedener Kriterien wie der Effizienz, Ergonomie und Nutzerfreundlichkeit bewertet. Die Spracheingabe und das Drumkeyboard wurden für am vielversprechendsten befunden und in einer Implementierung umgesetzt, sodass beide sowohl unabhängig voneinander als auch in Kombination verwendet werden können. In Benutzertests wurden die drei Methoden miteinander verglichen und der Schluss gezogen, dass die Spracheingabe unterstützt durch das Drumkeyboard am besten für die Texteingabe in VR geeignet scheint.

In Kapitel 2 wurden zunächst die wichtigsten Bewertungskriterien vorgestellt und daraufhin eine Vielzahl unterschiedlicher Eingabemethoden wie physische und virtuelle Tastaturen, Handschrift und Sprache untersucht. Es wurde gezeigt, dass die PC-Tastatur am effizientesten zu bewerten ist, für den mobilen Einsatz in VR jedoch die Spracheingabe – evtl. unterstützt durch eine andere Inputmodalität – die vielversprechendste Lösung darstellt.

In Kapitel 3 wurde das Ist-System beschrieben, in das die zu entwickelnde Lösung integriert werden sollte. Es wurden eine Auswahl der vielversprechendsten, zum Anwendungsfall passenden Eingabeverfahren getroffen und die genauen funktionalen und nichtfunktionalen Anforderungen an die zu entwickelnde Lösung formuliert.

In Kapitel 4 wurden der technische Kontext der Implementierung erläutert, verschiedene Optionen für die Spracherkennung diskutiert und die Auswahl der Software *Whisper* von OpenAI begründet. Weiter wurde das fachliche Design der Lösung vorgestellt, welches aus einem Korrekturinterface und einem Drumkeyboard, über das auch die Spracheingabe bedient werden kann, besteht. Zuletzt wurde der zugrundeliegende technische Entwurf präsentiert.

Die Rahmenbedingungen und der Ablauf der Benutzertests wurden in Kapitel 5 beschrieben. Es wurde eine Auswertung der Messungen von Eingabe- und Fehlerrate sowie der subjektiven Einschätzung der Probanden vorgenommen, welche stark darauf hindeuten, dass die multimodale Eingabe die besten Ergebnisse liefert. Zugleich wurde die Durchführung der Tests in einigen Punkten kritisiert und darauf hingewiesen, dass insbesondere aufgrund der geringen Teilnehmerzahl weitere Studien nötig sind, um eine definitive Aussage treffen zu können.

6.1 Ausblick

Mit dem zunehmenden Einsatz von VR im Berufs- und Privatleben steigt die Notwendigkeit einer zuverlässigen, performanten und leicht zu bedienenden Möglichkeit der Texteingabe. Aufgrund der rasanten Fortschritte in den Bereichen Spracherkennung und Natural Language Processing ist davon auszugehen, dass die Eingabe über Sprache eine zentrale Rolle in der Lösung dieses Problems spielen wird. Sie verspricht nicht nur eine sehr hohe Effizienz, sondern sticht auch in den Punkten Nutzerfreundlichkeit und Ergonomie besonders hervor. Zudem öffnet sie die Tür für eine inklusivere Nutzung von VR, um Menschen mit unterschiedlichen Hintergründen und Fähigkeiten eine natürliche Art der Bedienung zu ermöglichen.

Jedoch werden bis auf Weiteres auch andere Eingabemethoden von Bedeutung sein, um die Spracheingabe zu unterstützen und ihre noch vorhandenen Schwächen auszugleichen. Das zeigt sich am Beispiel der in dieser Arbeit entwickelten Lösung, welche eine Basis für die mobile Texteingabe in VR bildet, zugleich aber noch viel Potenzial zur Verbesserung aufweist.

Die Erforschung und Entwicklung von Texteingabe in Virtual Reality stehen noch am Anfang und weitere Studien sind erforderlich, um die Herausforderungen in Bezug auf Genauigkeit, Kontextsensitivität und Privatsphäre zu bewältigen. Die Kombination von technischem Fortschritt mit einem tieferen Verständnis für die Bedürfnisse der Nutzer wird es ermöglichen, die Interaktion zwischen Mensch und Maschine in der virtuellen Welt stetig zu verbessern.

Literaturverzeichnis

[1] Adhikary, Jiban ; Vertanen, Keith: Text entry in virtual environments using speech and a midair keyboard. In: *IEEE Transactions on Visualization and Computer Graphics* 27 (2021), Nr. 5, S. 2648–2658

[2] Adhikary, Jiban K. ; Vertanen, Keith: *Typing on Midair Virtual Keyboards: Exploring Visual Designs and Interaction Styles*. S. 132–151, 08 2021. – ISBN 978-3-030-85609-0

[3] Arif, Ahmed ; Stuerzlinger, Wolfgang: Analysis of text entry performance metrics. In: *TIC-STH'09: 2009 IEEE Toronto International Conference - Science and Technology for Humanity* (2009), 10, S. 100 – 105

[4] Arif, Ahmed S.: Predicting and reducing the impact of errors in character-based text entry. (2013)

[5] Boletsis, Costas ; Kongsvik, Stian: Controller-based Text-input Techniques for Virtual Reality: An Empirical Comparison. In: *Int. J. Virtual Real.* 19 (2019), S. 2–15

[6] Boletsis, Costas ; Kongsvik, Stian: Text Input in Virtual Reality: A Preliminary Evaluation of the Drum-Like VR Keyboard. In: *Technologies* 7 (2019), Nr. 2. – URL https://www.mdpi.com/2227-7080/7/2/31. – ISSN 2227-7080

[7] Bovet, Sidney ; Kehoe, Aidan ; Crowley, Katie ; Curran, Noirin ; Gutierrez, Mario ; Meisser, Mathieu ; Sullivan, Damien O. ; Rouvinez, Thomas: Using Traditional Keyboards in VR: SteamVR Developer Kit and Pilot Game User Study. (2018), S. 1–9

[8] Chen, Sibo ; Wang, Junce ; Guerra, Santiago ; Mittal, Neha ; Prakkamakul, Soravis: Exploring Word-Gesture Text Entry Techniques in Virtual Reality. (2019), S. 1–6. – URL https://doi.org/10.1145/3290607.3312762. ISBN 9781450359719

[9] Dash, Samir: *Bluetap: The Ultimate Virtual Reality (VR) Keyboard.* https://medium.com/eunoia-i-o/bluetap-the-ultimate-virtual-reality-vr-keyboard-77f1e3d57d6f. 2017. – abgerufen am 04.08.2023

[10] Didehkhorshid, Seyed Amir A. ; Philip, Siju ; Samimi, Elaheh ; Teather, Robert J.: Text Input in Virtual Reality Using a Tracked Drawing Tablet. (2020), S. 314–329. – URL https://doi.org/10.1007/978-3-030-59990-4_24. ISBN 978-3-030-59989-8

[11] Dörner, Ralf ; Steinicke, Frank: *Wahrnehmungsaspekte von VR.* In: Dörner, Ralf (Hrsg.) ; Broll, Wolfgang (Hrsg.) ; Grimm, Paul (Hrsg.) ; Jung, Bernhard (Hrsg.): *Virtual und Augmented Reality (VR/AR): Grundlagen und Methoden der Virtuellen und Augmentierten Realität.* Berlin, Heidelberg : Springer Berlin Heidelberg, 2019. – URL https://doi.org/10.1007/978-3-662-58861-1_2. – ISBN 978-3-662-58861-1

[12] Dube, Tafadzwa ; Arif, Ahmed: *Text Entry in Virtual Reality: A Comprehensive Review of the Literature.* S. 419–437, 06 2019. – ISBN 978-3-030-22642-8

[13] Elmgren, Rasmus: Handwriting in VR as a Text Input Method. (2017)

[14] Fashimpaur, Jacqui ; Kin, Kenrick ; Longest, Matt: PinchType: Text Entry for Virtual and Augmented Reality Using Comfortable Thumb to Fingertip Pinches. (2020), S. 1–7. – URL https://doi.org/10.1145/3334480.3382888. ISBN 9781450368193

[15] Grubert, Jens ; Witzani, Lukas ; Ofek, Eyal ; Pahud, Michel ; Kranz, Matthias ; Kristensson, Per: Effects of Hand Representations for Typing in Virtual Reality. (2018), 03, S. 151–158

[16] Hoppe, Adrian H. ; Otto, Leonard ; Camp, Florian van de ; Stiefelhagen, Rainer ; Unmüssig, Gabriel: qVRty: Virtual Keyboard with a Haptic, Real-World Representation. (2018), S. 266–272

[17] Jiang, Haiyan ; Weng, Dongdong: HiPad: Text entry for Head-Mounted Displays Using Circular Touchpad. (2020), 03, S. 692–703

[18] Kern, Florian ; Niebling, Florian ; Latoschik, Marc E.: Text Input for Non-Stationary XR Workspaces: Investigating Tap and Word-Gesture Keyboards in Virtual and Augmented Reality. In: *IEEE Transactions on Visualization and Computer Graphics* 29 (2023), Nr. 5, S. 2658–2669

[19] Kim, Youngwon R. ; Kim, Gerard J.: HoVR-Type: Smartphone as a Typing Interface in VR Using Hovering. (2016), S. 333–334. – URL https://doi.org/10.1145/2993369.2996330. ISBN 9781450344913

[20] Leng, Jiaye ; Wang, Lili ; Liu, Xiaolong ; Shi, Xuehuai ; Wang, Miao: Efficient Flower Text Entry in Virtual Reality. In: *IEEE Transactions on Visualization and Computer Graphics* 28 (2022), Nr. 11, S. 3662–3672

[21] Lu, Xueshi ; Yu, Difeng ; Liang, Hai-Ning ; Xu, Wenge ; Chen, Yuzheng ; Li, Xiang ; Hasan, Khalad: Exploration of Hands-free Text Entry Techniques For Virtual Reality. (2020), S. 344–349

[22] Lyons, Kent ; Starner, Thad ; Plaisted, Daniel ; Fusia, James ; Lyons, Amanda ; Drew, Aaron ; Looney, EW: Twiddler typing: One-handed chording text entry for mobile phones. (2004), S. 671–678

[23] Ma, Xinyao ; Yao, Zhaolin ; Wang, Yijun ; Pei, Weihua ; Chen, Hongda: Combining Brain-Computer Interface and Eye Tracking for High-Speed Text Entry in Virtual Reality. (2018). – URL https://doi.org/10.1145/3172944.3172988. ISBN 9781450349451

[24] Malik, Mishaim ; Malik, Muhammad K. ; Mehmood, Khawar ; Makhdoom, Imran: Automatic speech recognition: a survey. In: *Multimedia Tools and Applications* 80 (2021), S. 9411–9457

[25] Markussen, Anders ; Jakobsen, Mikkel R. ; Hornbæk, Kasper: Vulture: A Mid-Air Word-Gesture Keyboard. (2014). – URL https://doi.org/10.1145/2556288.2556964. ISBN 9781450324731

[26] Pham, Duc-Minh ; Stuerzlinger, Wolfgang: HawKEY: Efficient and Versatile Text Entry for Virtual Reality. (2019). – URL https://doi.org/10.1145/3359996.3364265. ISBN 9781450370011

[27] Pick, Sebastian ; Puika, Andrew S. ; Kuhlen, Torsten W.: SWIFTER: Design and evaluation of a speech-based text input metaphor for immersive virtual environments. (2016), S. 109–112

[28] Prätorius, Manuel ; Valkov, Dimitar ; Burgbacher, Ulrich ; Hinrichs, Klaus: DigiTap: An Eyes-Free VR/AR Symbolic Input Device. (2014), S. 9–18. – URL https://doi.org/10.1145/2671015.2671029. ISBN 9781450332538

[29] Rajanna, Vijay ; Hansen, John P.: Gaze Typing in Virtual Reality: Impact of Keyboard Design, Selection Method, and Motion. (2018). – URL https://doi.org/10.1145/3204493.3204541. ISBN 9781450357067

[30] Ruan, Sherry ; Wobbrock, Jacob O. ; Liou, Kenny ; Ng, Andrew ; Landay, James: Speech is 3x faster than typing for english and mandarin text entry on mobile devices. In: *arXiv preprint arXiv:1608.07323* (2016)

[31] Salthouse, Timothy A.: Effects of age and skill in typing. In: *Journal of Experimental Psychology: General* 113 (1984), Nr. 3, S. 345

[32] Soto, Eduardo ; Teather, Robert J.: Text Entry in Virtual Reality: Implementation of FLIK Method and Text Entry Testbed. (2020), S. 225–244. – URL https://doi.org/10.1007/978-3-030-59990-4_18. ISBN 978-3-030-59989-8

[33] Soukoreff, R. W. ; MacKenzie, I. S.: Metrics for Text Entry Research: An Evaluation of MSD and KSPC, and a New Unified Error Metric. (2003), S. 113–120. – URL https://doi.org/10.1145/642611.642632. ISBN 1581136307

[34] Speicher, Marco ; Feit, Anna M. ; Ziegler, Pascal ; Krüger, Antonio: Selection-Based Text Entry in Virtual Reality. (2018), S. 1–13. – URL https://doi.org/10.1145/3173574.3174221. ISBN 9781450356206

[35] Venkatakrishnan, Roshan ; Venkatakrishnan, Rohith ; Chung, Chih-Han ; Wang, Yu-Shuen ; Babu, Sabarish: Investigating a Combination of Input Modalities, Canvas Geometries, and Inking Triggers on On-Air Handwriting in Virtual Reality. 19 (2022), nov, Nr. 4. – URL https://doi.org/10.1145/3560817. – ISSN 1544-3558

[36] Yanagihara, Naoki ; Shizuki, Buntarou: Cubic Keyboard for Virtual Reality. (2018), S. 170. – URL https://doi.org/10.1145/3267782.3274687. ISBN 9781450357081

[37] Yildirim, Caglar ; Osborne, Ethan: Text Entry in Virtual Reality: A Comparison of 2D and 3D Keyboard Layouts. (2020), S. 450–460. – URL https://doi.org/10.1007/978-3-030-59990-4_33. ISBN 978-3-030-59989-8

[38] Yu, Chun ; Gu, Yizheng ; Yang, Zhican ; Yi, Xin ; Luo, Hengliang ; Shi, Yuanchun: Tap, Dwell or Gesture? Exploring Head-Based Text Entry Techniques for HMDs.

(2017), S. 4479–4488. – URL https://doi.org/10.1145/3025453.3025964. ISBN 9781450346559

[39] Yu, Difeng ; Fan, Kaixuan ; Zhang, Heng ; Monteiro, Diego ; Xu, Wenge ; Liang, Hai-Ning: PizzaText: Text Entry for Virtual Reality Systems Using Dual Thumbsticks. In: *IEEE Transactions on Visualization and Computer Graphics* 24 (2018), Nr. 11, S. 2927–2935

Erklärung zur selbstständigen Bearbeitung

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne fremde Hilfe selbständig verfasst und nur die angegebenen Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich gemacht.

Ort	Datum	Unterschrift im Original