Design, Implementierung und Evaluation neuartiger 3D Selektionsmetaphern in Virtual Reality

— Bachelorarbeit —
Studiengang: Informatik
Fachbereich 3: Mathematik und Informatik

vorgelegt von
Florian Alexander Rohde
Landstr. 4, 27308 Kirchlinteln
Matrikel-Nummer: 2776865

05. März 2018

Referent der Arbeit:	Prof. Dr. Gabriel Zachmann
Korreferent der Arbeit:	Prof. Dr. Johannes Schöning

Eidesstattliche Erklärung

Hiermit versichere ich, die vorliegende Arbeit selbstständig und unter ausschließlicher Verwendung der angegebenen Literatur und Hilfsmittel erstellt zu haben.

Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen wurden, habe ich unter Angabe der Quellen als solche kenntlich gemacht.

Grafiken, die keine Quellenangabe aufweisen, wurden im Rahmen dieser Bachelorarbeit von mir selbst erstellt.

Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt.

Bremen, 05. März 2018

Zusammenfassung

In dieser Arbeit geht es um den Vergleich der Effizienz und Effektivität verschiedener Selektionsmethoden in virtuellen Umgebungen. Dazu werden innerhalb der Arbeit vier neuartige Selektionsmetaphern auf Basis einer Implementierung in der Unreal Engine 4 evaluiert.

Abstract

This work is about comparing the efficiency and effectiveness of different selection methods in virtual environments. For this purpose, four novel selection metaphors are evaluated based on an implementation using Unreal Engine 4.

Genderhinweis

Aus Gründen der leichteren Lesbarkeit wird in der vorliegenden Bachelorarbeit die gewohnte männliche Sprachform bei personenbezogenen Substantiven und Pronomen verwendet. Dies impliziert jedoch keine Benachteiligung des weiblichen Geschlechts, sondern soll im Sinne der sprachlichen Vereinfachung als geschlechtsneutral zu verstehen sein.

Inhaltsverzeichnis

1. Einleitung

Die Selektion ist eine grundlegende Aufgabe unseres täglichen Lebens sowie bei der Interaktion in virtuellen Umgebungen (*virtual environments*) [6].

Die Effizienz einer solchen Selektion hat daher direkten Einfluss auf das Ergebnis jeglicher Bearbeitungsaufgaben, insbesondere im Bereich der virtuellen Montage [25]. Dennoch hat die bisherige Forschung gezeigt, dass eine einzelne Technik nicht für jede mögliche Situation bestmögliche Ergebnisse liefern kann. Dies liegt daran, dass es sich um einen sehr komplexen Vorgang handelt, der eine Vielzahl verschiedener Abhängigkeiten aufweist. Dazu zählen beispielsweise das Design der Aufgabe, aber auch die Erfahrung des Nutzers. Weitere Einflussfaktoren sind unter anderem Ermüdung, Feedback und Spaß [7].
Ein bekanntes Beispiel dafür stellt eine Supermarktszene dar, die von den Gewinnern des "Grand Prize" des IEEE 3DUI, einer jährlich stattfindenden Konferenz mit den Themen Virtual Reality und 3D Benutzeroberflächen, vorgestellt wurde. Dabei hatten Probanden die Aufgabe, kleine und teilweise verdeckte Objekte aus der stark überladenen Szene ("*cluttered environment*") auszuwählen (siehe Abbildung 2, Seite A-1) [16]. Ein normaler *Raycast* führt hier zu hohen Fehlerraten und langen Selektionszeiten, da zunächst entweder Objekte verschoben werden müssen, oder eine Annäherung im virtuellen Raum erfolgen muss [3]. Bei *Raycast* handelt es sich um eine grundlegende Selektionstechnik, bei der ein virtueller Strahl vom Controller ausgehend berechnet wird. Sein Kollisionspunkt wird für die Objektauswahl bei der Selektion verwendet.

Die Anzahl existenter Selektionsmetaphern sowie -ansätze ist entsprechend groß und es bedarf einer Evaluation bezüglich ihrer Effizienz.

Aufgrund der zuvor beschriebenen Situation bestand die Motivation für diese Arbeit darin, existiert eine Vielzahl von Ansätzen, die versuchen, die Selektion so effizient und effektiv wie möglich zu gestalten.

In vielen Fällen handelt es sich um Kombinationen bereits existenter Metaphern, die, im Vergleich zu ihren Komponenten, eine bessere Performance bieten.
In der Regel erfolgt eine Evaluation dieser "neuen" Techniken jedoch hauptsächlich mit den ihnen zugrundeliegenden Methoden, wodurch eine direkte Vergleichbarkeit aller bekannten Metaphern untereinander nicht oder nur auf theoretischer Basis gegeben ist.

An dieser Stelle setzt nun die vorliegende Bachelorarbeit an. *Raycast* [23] wird zusammen mit drei neuartigen Selektionsmetaphern - *lenSelect* [30], *PRECIOUS!* [22] sowie *VOTE* [24] - implementiert und in Bezug auf Effizienz und Effektivität verglichen.

Dafür wird die *Unreal Engine 4* von *Epic Games* mit dem Head Mounted Display *HTC Vive* verwendet und die Selektion erfolgt über die dazugehörigen Motion Controller.

2. Stand der Forschung

Grundsätzlich lassen sich Selektionstechniken in zwei Hauptkategorien einteilen.
Auf der einen Seite gibt es Techniken mit unmittelbarer Selektion, die lediglich eine präzise Auswahl ohne anschließende Verfeinerung nutzen. Zum Anderen existieren solche, die mittels mehrerer iterativer Schritte zum Ziel führen. Bei Letzteren sind die Schritte aufeinander aufbauend, wodurch die Bearbeitung einer Selektionsaufgabe mit weniger Fehlern durchgeführt werden kann, gleichzeitig impliziert diese Art aber auch eine längere Selektionszeit. Einige dieser Methoden werden im Folgenden näher vorgestellt.

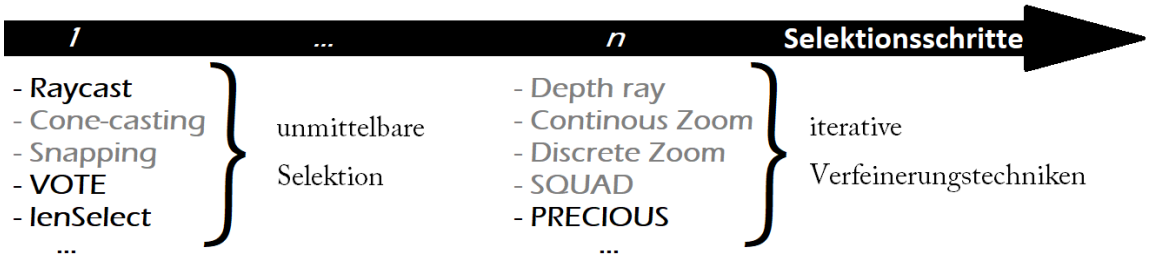


Abbildung 2.1.: Kategorien von Selektionstechniken

2.1. Techniken mit unmittelbarer Selektion

Eine weit verbreitete Selektionstechnik ist *Raycast*, welche auf einem virtuellen Strahl basiert, der vom in der Hand befindlichen Controller des Nutzers ausgehend, in die Richtung, in die er zeigt, berechnet wird. Dadurch kann der Nutzer auf ein Objekt im Raum zeigen, um es auszuwählen. Das System kalkuliert den Verlauf des Strahls und liefert, sofern vorhanden, einen Kollisionspunkt (*Hitpoint*) zurück. In ihrer Grundausführung weißt die Technik jedoch, aufgrund der Einfachheit, einere Reihe von Nachteilen auf. Beispielsweise gibt es eine Fluktuation durch natürliches Zittern der Hand und/oder ungenaues Tracking der Controller (*Jitter*). Dies erschwert es dem Nutzer den Ursprung und die Ausrichtung des virtuellen Strahls genau zu kontrollieren. Weiterhin resultiert aus einer kleinen Handbewegung mit zunehmender Länge des Strahls eine deutlich verstärkte Bewegung des Kollisionspunktes, was bei weit entfernten oder kleinen Objekten zu einer erhöhten Ungenauigkeit führt. In diesem Fall erfordert deren Auswahl einen hohen Präzisionsgrad [28].

Um diesen Problemen zu begegnen, wurde eine Vielzahl von Verbesserungsvorschlägen erarbeitet. Einige Beispiele dafür stellen [3] vor. Das sogenannte *Cone-Casting* [21] verwendet anstelle des Strahls ein kegelförmiges Selektionsvolumen. Dadurch wird es einfacher, Objekte auf Distanz auszuwählen. Allerdings hat dies bei überladenen (*cluttered*) Umgebungen den Nachteil, dass es zwangsläufig zu multiplen Selektionen kommt. Dadurch wird entweder eine noch höhere Präzision oder zumindest ein weiterer iterativer Verfeinerungsschritt notwending. *Snapping* [10] verwendet ebenfalls ein kegelförmiges Selektionsvolumen und "bewertet" die Objekte innerhalb dessen über eine definierte Zeitspanne hinweg: Wenn sich ein Objekt in einem Frame innerhalb des Kegels befindet, erhält es eine *Stimme*. Die Stimmen werden über eine feste Anzahl Frames summiert und verglichen. Ein Frame bezeichnet dabei ein vom Computer generiertes Bild. Die Framerate gibt an, wie viele Bilder pro Sekunde dargestellt werden (können). Idealerweise versucht man mindestens 90 Frames pro Sekunde (FPS) zu erreichen, damit der Nutzer kein hängen oder stocken des Bildes bemerkt. Eine sehr ähnliche Strategie verfolgt *VOTE* welche sich durch die Verwendung eines Strahls statt des Kegels von *Snapping* unterscheidet. Dadurch wird sowohl der *Jitter*-Effekt als auch der Nachteil der Verlängerung des Selektionsvorganges durch zwangsläufig auftretende Disambiguierung angegangen. Eine solche wird aufgrund der Selektion mehrerer Objekte durch Verwendung eines Kegels, beispielsweise beim *Snapping*, notwendig.

Andere Techniken verbessern *Raycast*, indem der *Control-display ratio* manipuliert wird. Dieser gibt das Verhältnis zwischen der natürlichen Bewegung im physischen und der resultierenden im virtuellen Raum an. Ein einfaches Beispiel dafür ist das Verhältnis zwischen der Bewegung der Computermaus und der daraus resultierenden Bewegung des Mauszeigers auf dem Bildschirm.

Einen ähnlichen Ansatz verfolgt *lenSelect*: Hier wird die effektive Breite des Zielobjektes in Abhängigkeit von der Entfernung zum *Hitpoint* manipuliert (Abbildung 2.2), wodurch die Selektion vereinfacht wird und die zuvor beschriebenen Nachteile ebenfalls vermieden werden.

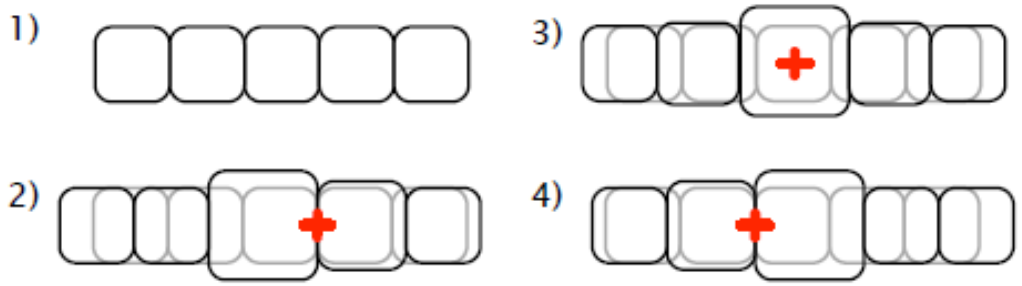


Abbildung 2.2.: Darstellung des Vergrößerungseffektes von *lenSelect* [30]

2.2. Techniken mit iterativer Verfeinerung

Auch wenn die zuvor beschriebenen Techniken die Präzision von *Raycast* verbessern, zwingen sie den Nutzer jederzeit vorsichtig und aufmerksam zu bleiben, um Fehler zu vermeiden.

Daher gibt es auch iterative Verfeinerungstechniken, die durch Verwendung mehrerer Schritte zur Selektion gelangen. Beispielsweise stellten Grossman und Balakrishnan [18] die *depth-ray* Technik vor, die dem klassischen *Raycast* eine Kontrolle über die Tiefe hinzufügt, um okkludierte Objekte auswählen zu können. Dabei werden zwei Aktionen zur Selektion ausgeführt.

Eine weitere Möglichkeit ist die Verwendung von Zoom-Funktionen, welche den Nutzer praktisch näher an das Zielobjekt heranbewegen, oder bestimmte Bereiche beziehungsweise Objekte vergrößern. Hierfür wurden beispielsweise *Continous Zoom* und *Discrete Zoom* vorgestellt [3]. Bei Letzterem wird das Bild in vier Quadranten eingeteilt, zwischen denen ausgewählt werden kann. Dieser Schritt wird so lange (iterativ) wiederholt, bis sich nur noch ein Objekt in jedem befindet (Abbildung 2.3).

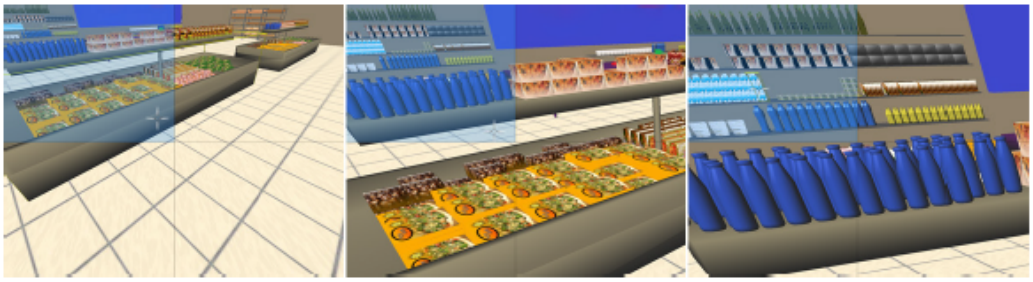


Abbildung 2.3.: Verfeinerungsschritte des diskreten Zooms [3]

In Abbildung 2.4 wird der Vorgang der Verfeinerung mit kontinuierlichem Zoom dargestellt: Der Zoom ist hier stufenlos in jede beliebige Richtung möglich. Der Nutzer wählt eine Region im Raum aus und drückt eine Taste solange wie der Zoom ausgeführt werden soll. Während dieses Vorgangs kann die Region angepasst werden. Ein Rückwärtszoom ist ebenfalls möglich. Im Anschluss wird die eigentliche Selektion mittels *Raycast* ausgeführt.

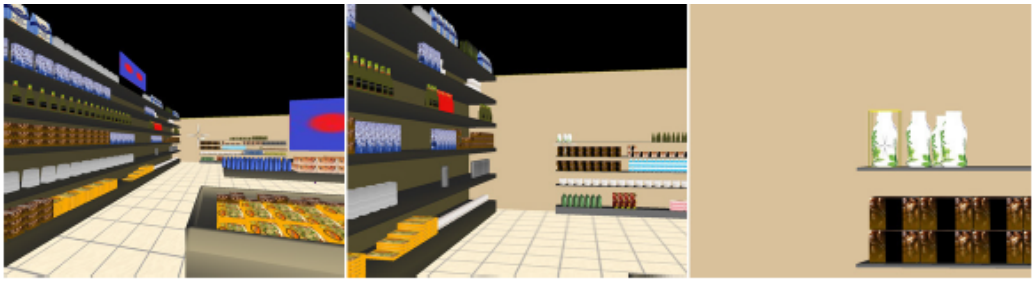


Abbildung 2.4.: Verfeinerungsschritte des kontinuierlichen Zooms [3]

Ein weiterer Ansatz wurde in *SQUAD* [3] umgesetzt. Hierbei wird eine Sphäre zur Grundselektion verwendet. Alle Objekte, die sich innerhalb des Radius dieser befinden, werden auf vier Quadranten (Abbildung 2.5, rechts) aufgeteilt. Danach wird iterativ ein Quadrant gewählt und alle Objekte darin wiederum in vier neue Gruppen aufgeteilt. Sobald sich nur noch ein Objekt im ausgewählten Quadranten befindet, erfolgt die abschließende Selektion.

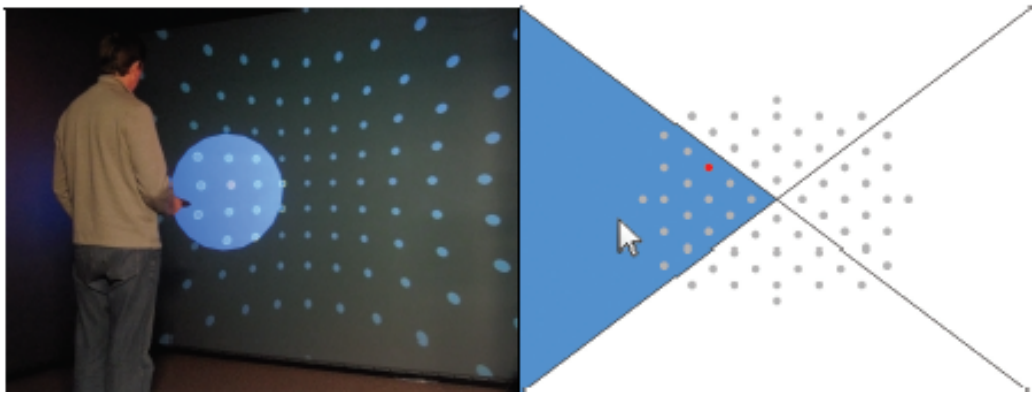


Abbildung 2.5.: Darstellung von *SQUAD* [3]

Dies ermöglicht eine detlich erhöhte Genauigkeit, kann allerdings bei überladenen Umgebungen schnell zu sehr langwierigen Auswahlprozessen führen.

Um dieses Problem anzugehen, wurde unter anderem *PRECIOUS* als Kombination von *Cone-casting*, *Zoom*, *Stretch Go-Go* [5] und schließlich einem Disambiguierungsansatz [11] vorgestellt. Bei *Stretch Go-Go* handelt es sich um eine Verbesserung der *Go-Go* Technik [26], bei welcher der Arm des Nutzers, und somit dessen Reichweite, iterativ verlängert wird, sofern er in einem definierten Abstand ausgestreckt ist. Bei *Stretch Go-Go* geht dies bis ins Unendliche.

Auf Distanz wird dies äußerst unpräzise, weshalb hier ein Kegel, welcher sich vom Controller ausgehend ausbreitet, im Sinne einer Taschenlampenmetapher zur Verfeinerung der Selektion verwendet (siehe Abbildung 2.6, A).

Abbildung 2.6.: Darstellung von *PRECIOUS* [22]

Der Effekt der Verlängerung wird auf den Kegel angewendet, und zwar in Abhängigkeit davon, wie weit der Nutzer seinen Arm ausstreckt.

Befinden sich bei der Selektion mehrere Objekte innerhalb des Kegels, wird gemäß Disambiguierungsansatz verfahren.
Das heißt, wenn es genau zwei Objekte sind, werden diese nebeneinander positioniert, während alle anderen Objekte ausgeblendet werden. Dies ermöglicht eine genaue Auswahl (manuelle Verfeinerung) eines einzelnen Objektes.
Handelt es sich hingegen um mehrere Ziele, so erfolgt eine Teleportation in die Richtung, in die der Nutzer zeigt (Abbildung 2.6, B).
Daraufhin ist vom neuen Standpunkt aus eine weitere Selektion möglich, die auch auf den zuvor benannten Abläufen beruht. Abschließend wird, nach Selektion eines einzelnen Objektes, der Nutzer wieder an seinen Ausgangspunkt zurückbewegt (Abbildung 2.6, C).

3. Architektur

3.1. Aufbau

In diesem Kapitel folgt eine nähere Beschreibung der Software, die im Rahmen dieser Bachelorarbeit entwickelt wurde.

Es handelt sich dabei um den Pawn `ViveDefaultPawn` sowie die Klassenstrukturen `SelectableObject`, deren Kindklassen als Repräsentanten für Objekte, sowie `Arrow_Marker` und `Start Activator`. Der Pawn ist die physische Repräsentation des Spielers in der virtuellen Welt. Er enthält die programmierte Logik, also die Definition des Aussehens, der Interaktion und der Kollision mit der Umgebung [15].

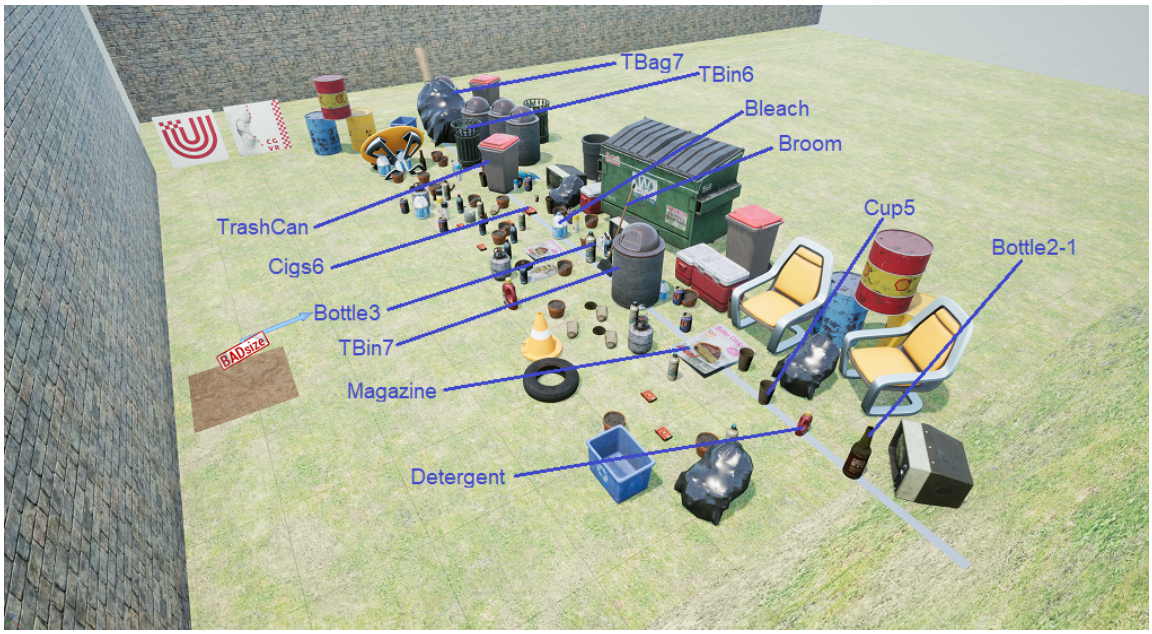


Abbildung 3.1.: Darstellung der Ziele in virtueller Umgebung

Bei dem verwendeten Bildmaterial handelt es sich um Objekte aus dem Unreal Starter Content. Zusätzlich enthalten sind *City Trash and Waste Set* [9], *oil barrel* [2], *street cones* [1] sowie *watering can* [12]. Diese Objekte sind entweder frei verfügbar oder wurden im Rahmen der Bachelorarbeit käuflich erworben.

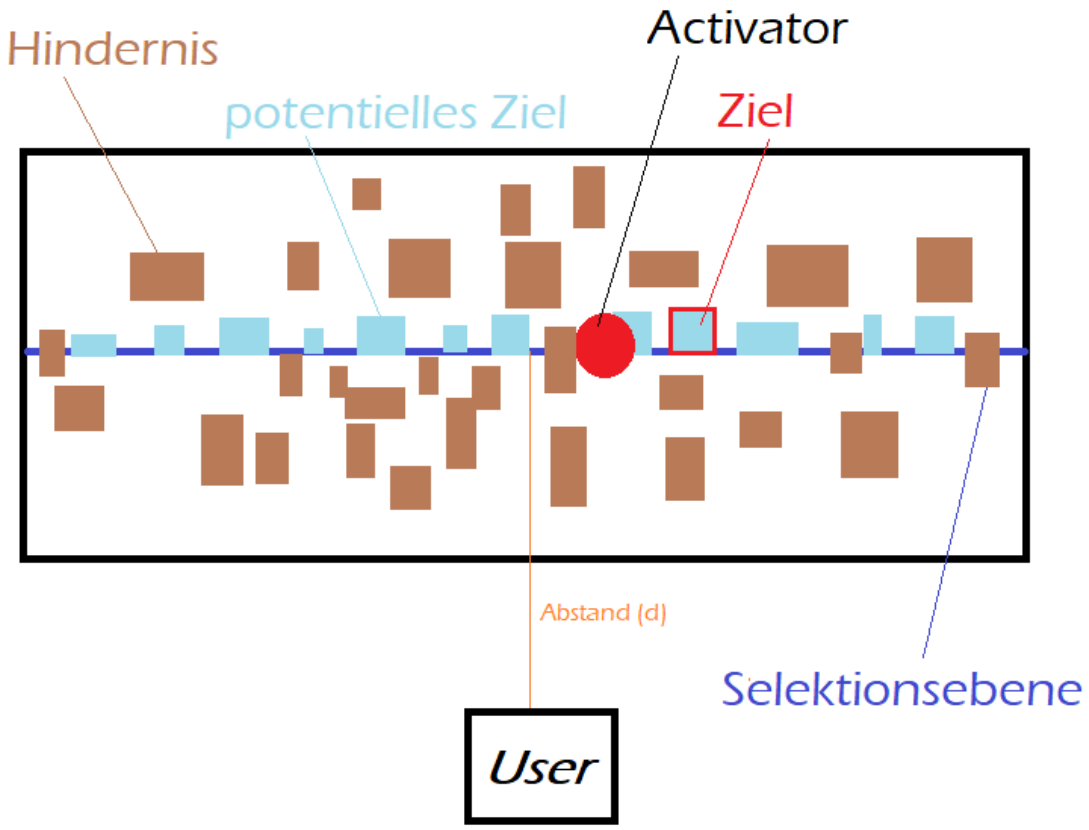


Abbildung 3.2.: Beispielhafter, schematischer Aufbau (Vogelperspektive)

Der Nutzer steht im vorgegebenen Bereich (im Abstand d zur Selektionsebene) und darf diesen nicht verlassen. Auf letzterer Ebene befinden sich verschiedene Objekte, von denen 12 während der Evaluation als Ziel vorgegeben werden. Die Objekte stellen jeweils verschiedene Schwierigkeitsgrade dar.

Zusätzlich wurden sowohl auf der Ebene, als auch verteilt im Raum, weitere Hindernisse positioniert. Dadurch entsteht eine *cluttered environment*, welche die Komplexität der Aufgabe erhöht.

Ein roter Kreis (`Activator`) wird pro Objekt in fest definiertem Abstand und Winkel zum aktuellen Ziel positioniert. Durch Aktivierung wird der Kreis grün und der Selektionsvorgang beginnt.

Der Nutzer hat einen Controller in der Hand, über dessen “Trigger-Taste” er eine Aktion auslösen kann (Abbildung 1 im Anhang A-1, Markierung 7).

3.2. Ruheposition

Grundsätzlich muss der Nutzer vor jeder Selektion den Arm hängen lassen (Controller zeigt nach unten) und den Trigger betätigen. Dies wird als Ruheposition bezeichnet.
Daraufhin wird die Szene dynamisch angepasst, da jedes Objekt einen festgelegten *Index of difficulty* (ID) darstellt, dieser aber abhängig von der Position des Controllers ist.
Der ID beschreibt die Komplexität einer Selektion nach Fitt's Law [13]. Die Berechnung wird in Abbildung 3.3 verdeutlicht und im Folgenden beschrieben.

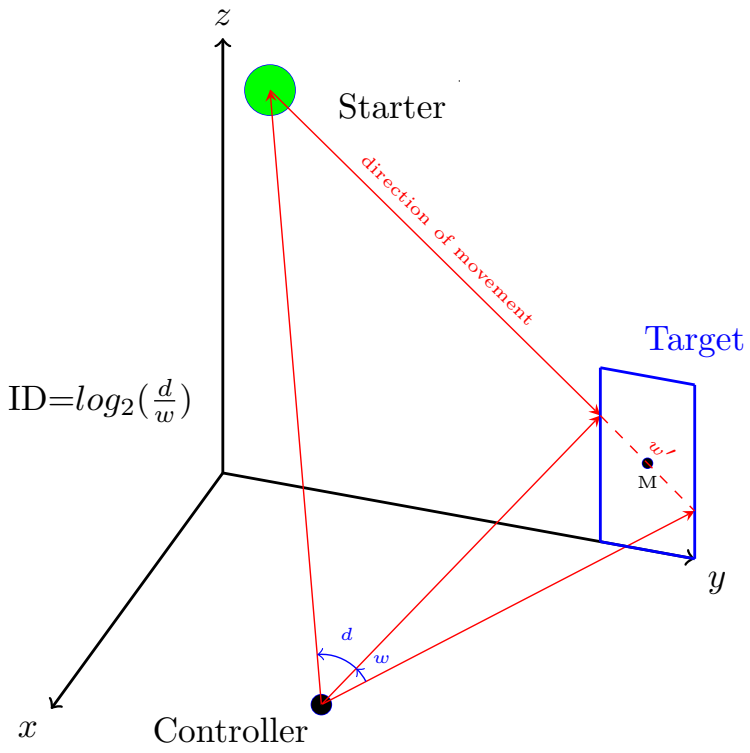


Abbildung 3.3.: Berechnung des *Index of difficulty*

Ausgehend vom Controller wird der Logarithmus des Quotienten der Winkel d und w berechnet, welche durch die Vektoren zwischen Controller, *Starter* (bzw. *Activator*) und dem Zielobjekt aufgespannt werden. Der Vektor zwischen *Starter* und dem Zielobjekt wird dabei in Richtung der Bewegung berechnet, was zur Folge hat, dass die effektive Objektbreite w' (*effective target width*) nicht zwangsläufig gleich der *realen* Breite des Objekts ist.
Dieser Vektor läuft vom Mittelpunkt des Kreises durch den des Ziels, wobei sein dortiger Ein- und Austrittspunkt zur Berechnung von w' (und somit auch w) verwendet werden. Demzufolge hat ein großes (leicht selektierbares) Objekt einen niedrigen, ein kleines Objekt einen hohen ID.

3.3. Selektionsablauf

Nachdem der Trigger in der Ruheposition betätigt wurde, startet eine Selektionsaufgabe. Das Zielobjekt wird dabei mit einem roten Rahmen dargestellt und zusätzlich durch horizontale und vertikale Pfeile markiert.
Weiterhin wird oberhalb des Objekts der `Activator` sichtbar, welcher durch einen roten Kreis dargestellt wird. Um den Selektionsvorgang auszuführen, muss zunächst der Kreis ausgewählt und angeklickt werden. Er wird daraufhin grün. Anschließend kann das eigentliche Ziel selektiert werden (Abbildung 3.4).

Dies hat den Sinn, dass die Zeit der Zielsuche nicht mit in die Messung eingeht: Die Bearbeitungszeit beginnt in dem Moment, in dem der Kreis aktiviert wurde.
Zusätzlich werden ab diesem Zeitpunkt auch die Fehler gezählt, die bei der Auswahl auftreten. Letztere kann erfolgen, sobald ein Objekt markiert ("gehovered") ist. Dies ist entweder der Fall, wenn darauf gezeigt wird oder es durch eine methodenspezifische Auswahl in blau hervorgehoben wurde.

Der Nutzer kann daraufhin durch Betätigung der Trigger-Taste die Selektion auslösen.
Falls es sich bei der Auswahl nicht um das Ziel handelte, wird dies mit einer einfachen Vibration an den Nutzer weitergegeben, während der Task weiter läuft. Bei korrekter Selektion sind es hingegen zwei Vibrationen und kurz darauf die Aufforderung, sich wieder in die Ruheposition zu begeben.

Abbildung 3.4.: Activator vor und nach Aktivierung

3.3.1. *lenSelect*

lenSelect basiert auf der Idee, eine dynamische Vergrößerung der Ziele in Abhängigkeit von der Entfernung zu dem Punkt, auf den gezeigt wird, auszuführen.
Praktisch wird in einem Radius um den *Hitpoint* eine prozentuale Vergrößerung vorgenommen, wodurch sich die effektive Breite des Objektes erhöht (Abbildung 3.5).

Per definitionem muss der *Index of difficulty* im Vergleich zum unmodifizierten Objekt sinken.

Abbildung 3.5.: lenSelect Effekt

3.3.2. *PRECIOUS*

Als einzige Selektionsmethode mit iterativer Verfeinerung (in dieser Evaluation) ist *PRECIOUS* ein Spezialfall.

Zum Einen wird anstatt des Strahls ein Kegel als Selektionsvolumen verwendet, zum Anderen hat dieses weitere spezielle Eigenschaften:
Es werden drei Regionen definiert, in denen sich der Controller befinden kann, um den Kegel zu strecken (ausfahren) oder zu stauchen (einfahren). Initial ist der Kegel immer eingefahren. Wenn der Nutzer den Arm ausstreckt, wird er ausgefahren, beim Zurücknehmen des Armes fährt er wieder ein. Dies wird durch eine Anzeige neben dem Controller visualisiert (Abbildung 3.6).

Abbildung 3.6.: Selektionsvolumen und Verfeinerungsschritt bei PRECIOUS

Weiterhin kann durch die Rotation des Handgelenkes der Öffnungswinkel des Kegels eingestellt werden, um viele oder wenige Objekte zu selektieren.

Die nächste Besonderheit folgt direkt daraus: durch die Taschenlampenmetapher wird eine multiple Selektion möglich, welche, je nach Anzahl der Objekte, in unterschiedlichen Aktionen resultiert.

Wenn mehr als zwei Objekte innerhalb des Volumens liegen, sobald die Selektion ausgelöst wurde, teleportiert das System den Benutzer näher an die Objekte heran (siehe Abbildung 2.6 A+B, Seite 6). Bei genau zwei Objekten werden diese nebeneinander angereiht und ermöglichen so eine Einzelselektion (Abbildung 3.6, rechts), während die anderen Objekte ausgeblendet werden.

Der Teleportationsschritt kann wiederholt werden, bis sich nur noch einzelne Objekte anwählen lassen.

3.3.3. *Raycast*

Bei *Raycast* handelt es sich um die einfachste Implementierung dieser Art. Ausgehend vom Controller wird ein virtueller Strahl berechnet, dessen erste Kollision das Objekt darstellt, auf welches gezeigt wird.

3.3.4. *VOTE*

Der Ablauf ist aus Sicht des Nutzers bei *VOTE* äquivalent zum Vorherigen, jedoch gibt es aus technischer Sicht einen Unterschied.

Bei *VOTE* wird ein Bewertungssystem genutzt, welches in jedem Frame das aktuell markierte Objekt in einer Map speichert, beziehungsweise dessen Bewertung um den Wert eins erhöht (siehe Abbildung 3.7). Sobald die Selektion erfolgt, wird das Objekt, welches die höchste Bewertung (bzw. die meisten *Stimmen*) hat, ausgewählt.

Objekt	Wertung
1	1
2	5
3	0
9	7
11	1
91	0
26	19
13	4

Abbildung 3.7.: Schema der Bewertungsliste (Queue) bei *VOTE*

Ausgehend von einer Map der Länge 45 und 90 FPS hat dies zur Folge, dass die Markierungen der letzten 0,5 Sekunden betrachtet werden. Durch den Effekt kann beispielsweise eine Fluktuation durch Zittern oder fehlerhaftes Tracking ausgeglichen werden.

3.4. Distanz

Um eine erweiterte Vergleichbarkeit zu schaffen, wurde zudem die Möglichkeit entwickelt, die Distanz zwischen Nutzer und den Objekten (sowohl Ziele als auch Hindernisse) zu erhöhen. Hierbei handelt es sich um eine simple Verschiebung der Objekte um einen bestimmten Wert. Während der Evaluation werden zwei Distanzen verwendet (siehe Abbildung 3.8).

Abbildung 3.8.: Startscreen mit normaler und erhöhter Distanz

3.5. Spielerposition

Abbildung 3.9.: Fehlermeldung bei Verlassen des Bereiches

Der Nutzer sollte immer an der gleichen Stelle stehen, damit die Ergebnisse vergleichbar bleiben. Daher ist diese durch einen braunen Kasten auf dem virtuellen Boden markiert. Weiterhin wird softwareseitig ein Verlassen des Kastens zwar nicht verhindert, allerdings bricht in dem Fall die aktuelle Aufgabe (sofern aktiv) ab, und das jeweilige Ziel muss zu einem späteren Zeitpunkt erneut ausgewählt werden. Der Nutzer erfährt dies durch eine eindeutige Aufforderung, wieder an seinen Platz zurückzukehren (siehe Abbildung 3.9).

4. Implementierung

Die Implementierung der entstandenen Software *SSA* (*Simple Selection Analysis*) erfolgte, bis auf die Speicherung der Daten, vollständig in *Blueprints*. Dabei handelt es sich um ein Scripting-System, welches das Konzept einer knotenbasierten Oberfläche verfolgt. Es wird verwendet, um objektorientierte Klassen oder Objekte zu definieren. Das System ist besonders flexibel und leistungsstark, weil es Designern den Zugriff auf jegliche Funktionen ermöglicht, die normalerweise Programmierern vorbehalten sind. Weiterhin ist es auch problemlos möglich *Blueprints* mit `C++`-Code zu kombinieren [14].

4.1. ViveDefaultPawn

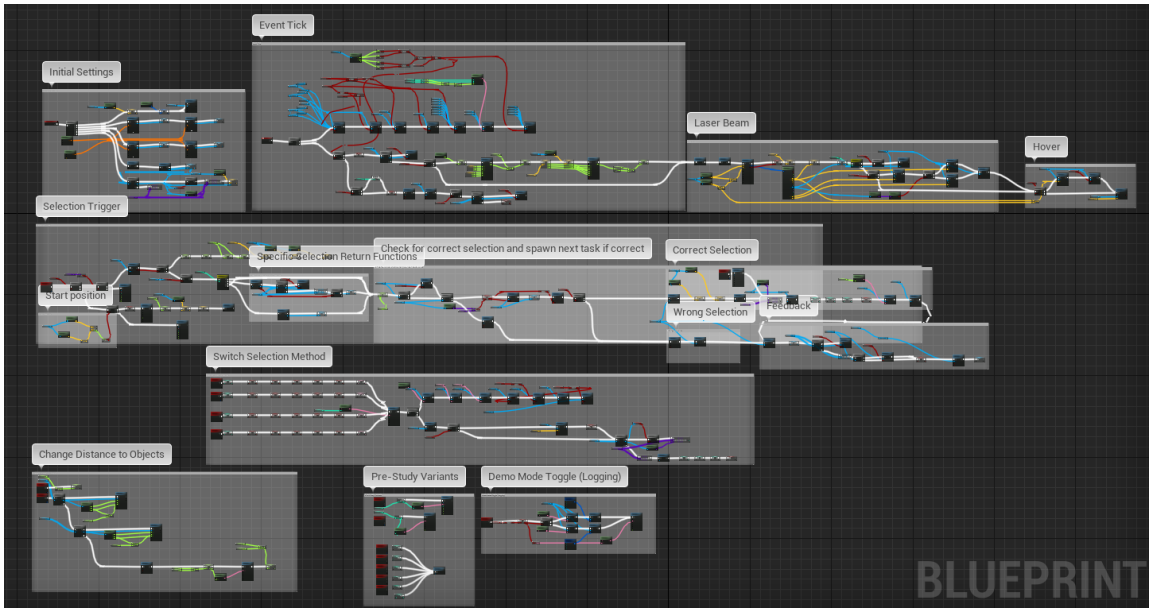


Abbildung 4.1.: `ViveDefaultPawn`

Die hauptsächliche Logik der entstandenen Software steckt im `ViveDefaultPawn`. Dieser ist in verschiedene Bereiche unterteilt (siehe Abbildung 4.1), welche im Folgenden erläutert werden.

4.1.1. Initial Settings

Der `Initial Settings` - Blueprint (Abbildung 4.2) ist dafür zuständig, die benötigten Objekte und Variablen initial zu erzeugen bzw. zu setzen und beispielsweise Instanzen der Markierungspfeile (`Arrow Marker`) zu spawnen. Dies bezeichnet die Erzeugung und Platzierung einer Instanz des Objekts in der virtuellen Welt.

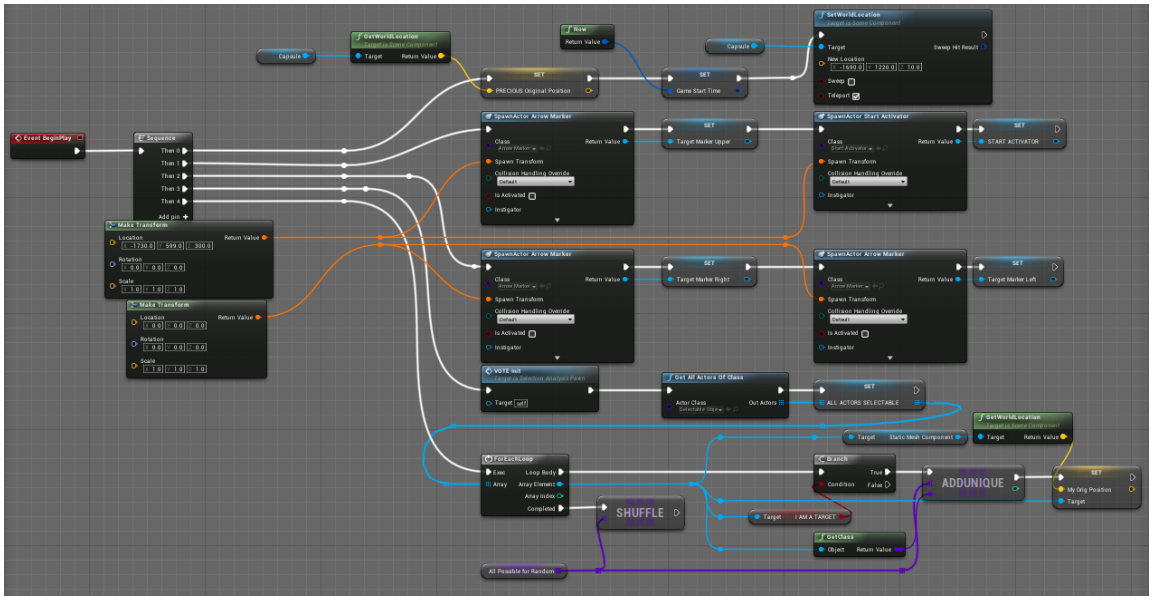


Abbildung 4.2.: Blueprint `Initial Settings`

Die Funktion `VOTE Init`, welche auch aufgerufen wird, erzeugt einen Platzhalter für die *VOTE*-Queue und das entsprechende Array, durch welches sie repräsentiert wird.

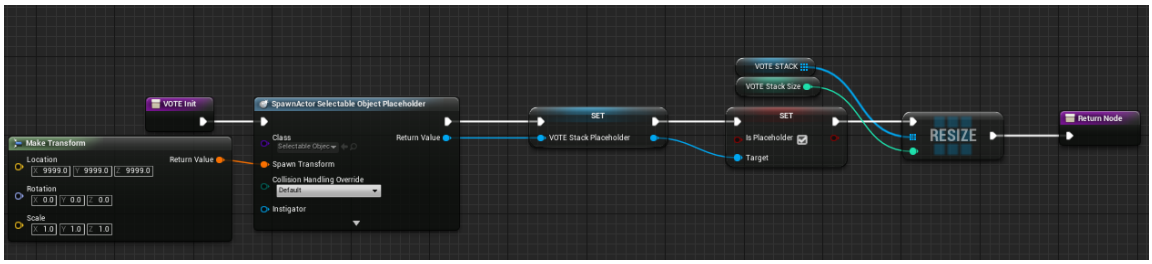


Abbildung 4.3.: Blueprint `VOTE Init`

Weiterhin werden noch alle Objekte in der Map, die den *Boolean* `I_AM_A_TARGET` gesetzt haben, in ein Array aufgenommen, was schließlich durchgemischt wird. Dies hat den Zweck, dass zu jeder Zeit die Anzahl bzw. Auswahl der Ziele geändert werden kann, indem im Level neue Objekte vom Typ `SelectableObject` erzeugt bzw. bestehende angepasst werden.

4.1.2. Event Tick

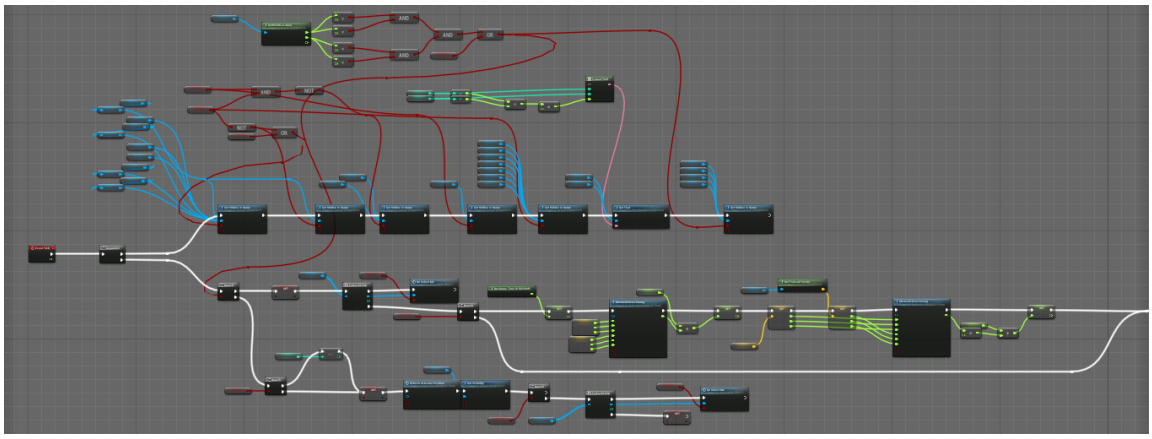


Abbildung 4.4.: Blueprint `Event Tick`

Hierbei handelt es sich um den wohl komplexesten Teil der Software, da dies die Hauptfunktion ist, welche in jedem Frame aufgerufen wird.

Im oberen Teil der Funktion wird die Sichtbarkeit aller relevanten Objekte definiert. Dabei handelt es sich um die Zielmarkierungspfeile, den Kegel für *PRECIOUS*, den Laserstrahl, die Anzeigetafeln usw.

Der mittlere Teil wird danach ausgeführt, wenn sich der Spieler in einem definierten Bereich befindet, in dem er stehen darf und das Spiel nicht unterbrochen wurde. Dies soll verhindern, dass die Probanden während der Evaluation instinktiv näher an Ziele herantreten und so die Schwierigkeit der Aufgabe beeinflussen könnten.

Wenn diese Bedingungen erfüllt sind, beginnt der eigentliche Programmablauf und es werden Attribute wie die Distanz, die der Controller seit dem letzten Frame zurückgelegt hat, ausgelesen und fortlaufend summiert.

Andernfalls, wenn der Nutzer den Bereich verlassen hat, wird der unterste Programmpfad aufgerufen, welcher den *Boolean* `STARTER` auf `falsch` setzt, den `Activator` und Laserstrahl ausblendet und schließlich alle Objekte “unhovered”. Als Gegenteil von “hover” bezeichnet dies das Entfernen der Markierung, welche durch ein halbtransparentes blaues Material dargestellt wird.

4.1.3. Laser Beam

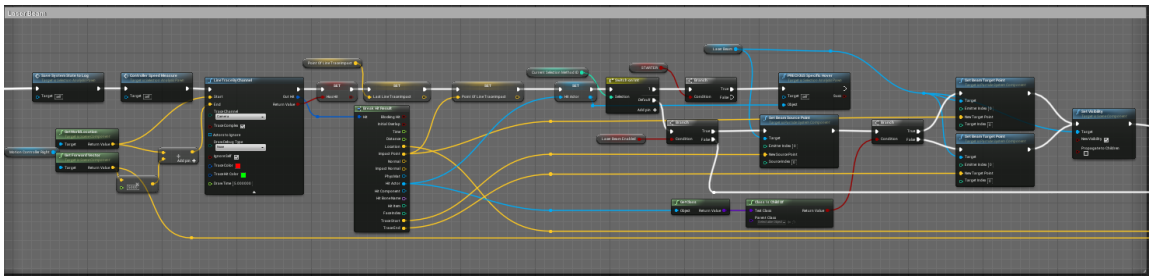


Abbildung 4.5.: Blueprint `LaserBeam`

Als nächstes wird der Systemstatus in einer Logdatei gespeichert.
Dieser enthält jegliche relevanten Informationen, wodurch die Funktion sehr unübersichtlich wird. Prinzipiell wird aber lediglich ein String aus den Variablen erzeugt und an den `SaveFileAgent` weitergegeben.

Weiterhin wird die Controller-Bewegungsgeschwindigkeit gemessen. Dazu wird in der Funktion `Controller Speed Measure` die Distanz der Punkte, auf die im letzten und aktuellen Frame gezeigt wurde, berechnet. Dies wird als Geschwindigkeit (x/Frame) bezeichnet und findet bei der Implementierung von *VOTE* Verwendung.

Schließlich wird noch ein *LineTraceByChannel* berechnet, welches die Implementierung der *Raycast*-Grundfunktion in Unreal darstellt: Mit je einer Koordinate als Start und Ziel berechnet diese die erste Kollision mit einem Objekt auf dem resultierenden Pfad. Diese Kollision (in Form des Objekts) wird gespeichert und zukünftig `Hit Actor` genannt, unter anderem in den Hover-Funktionen.

Weiterhin folgt dann die erste Unterscheidung nach aktiver Selektionsmethode: *PRECIOUS* verwendet ab diesem Punkt eine spezifische Funktion, da hier der Kegel anstatt des Strahls als Selektionsvolumen zum Einsatz kommt.

Bei allen anderen Methoden wird der `LaserBeam` visualisiert und dessen Attribute gesetzt. Sofern das Objekt, welches die Kollision ausgelöst hat, ein `SelectableObject` (also ein Zielobjekt) ist, wird der Strahl bis zu diesem Objekt dargestellt, andernfalls geht er unendlich weiter, beispielsweise durch Wände hindurch.

4.1.4. Hover

Die `Hover Object` - Funktion prüft zunächst, ob der Boolean `STARTER` *wahr* und `Hit Actor` gültig ist. Danach werden Booleans gesetzt, welche angeben, ob es sich bei letzterem um ein Zielobjekt handelt und ob es bereits markiert wurde.
Dann wird es zu einem `SelectableObject` gecastet, welches die folgenden spezifischen Hover-Methoden einfach verwenden können.

In Abhängigkeit von der aktiven Selektionsmethode wird die spezifische Hover-Funktion aufgerufen. Diese Funktionen werden auf den folgenden Seiten einzeln erklärt.

Abschließend wird nur noch der `QHovers`-Zähler erhöht. Dieser enthält somit die Anzahl der ausgeführten Hover-Effekte bis zur Selektion.

Algorithmus 1 Grober Ablauf des Hover-Vorgangs

```
1:  procedure HOVER OBJECT(HitActor)
2:      if STARTER == false then return false
3:      if !isValid(HitActor) then return false
4:      IsChild ← ClassIsChildOf(SelectableObject, HitActor.GetClass)
5:      IsSelected ← (CurrentlyHoveredObject == HitActor)
6:      switch CurrentSelectionMethod do
7:          case 0
8:              Entfernen des Hover-Effektes von CurrentlyHoveredObject und hovern des HitActor
9:              CurrentlyHoveredObject ← HitActor
10:         case 1
11:             Führe PRECIOUS-spezifisches Hover aus
12:         case 2
13:             Führe VOTE-spezifisches Hover aus
14:         case 3
15:             Führe lenSelect-spezifisches Hover aus
16:     QHovers ← QHovers+1
        return true
```

4.1.4.1. Hover für *Raycast*

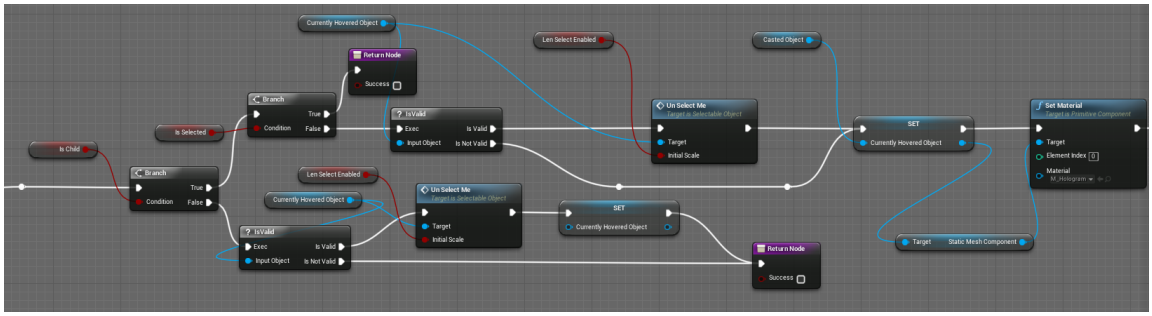


Abbildung 4.6.: Blueprint `Hover` und `Unhover`

Die *Raycast*-Version prüft trivial, ob es sich beim `Hit Actor` um eine Instanz einer Kindklasse von `SelectableObject` handelt (anhand der im ersten Teil erwähnten Booleans).

Wenn nicht, wird das aktuell gehoverte Objekt zurückgesetzt und der Methodenaufruf beendet. Andernfalls wird geprüft, ob das Objekt noch nicht hervorgehoben wurde.
Wenn dem nicht so ist, wird erneut geprüft, ob es bereits ein gehovertes Objekt gibt, welches dann zurückgesetzt wird.

Im nächsten Schritt wird das neue Objekt gespeichert und dessen Material auf das Hover-Material geändert (also der Hover-Effekt somit aktiviert).

4.1.4.2. Hover für *PRECIOUS*

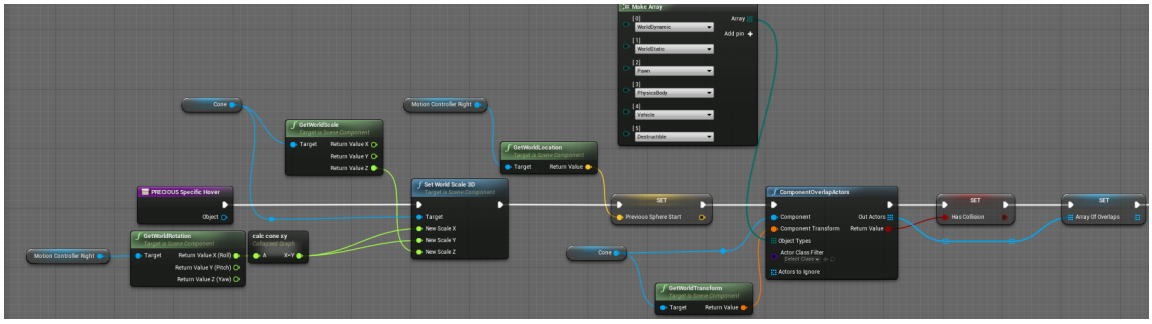


Abbildung 4.7.: Blueprint `Dynamic cone cast`

Im ersten Teil der Funktion wird der Kegel angepasst.
Anhand der Rotation des Handgelenks lässt sich dessen Öffnungswinkel variieren.

Dadurch, dass die Anpassung der Skalierung des *Static Mesh Component*s zwar keinen Einfluss auf die Position des Ursprungs des Objekts hat, aber dennoch die Bounding Box vergrößert wird, muss das Objekt entsprechend verschoben werden, damit der Kegel weiterhin direkt vor dem Controller “beginnt”.

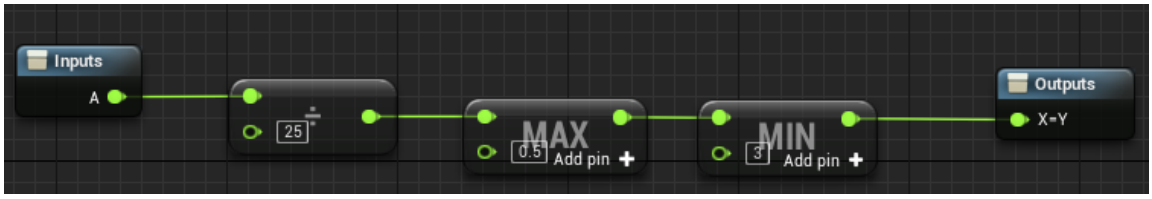


Abbildung 4.8.: Blueprint `calc cone xy`

Dafür werden die $X-$ und Y-Koordinaten des Cones mittels folgender Formel berechnet:

$$X, Y = min\left(3, max\left(0.5, \frac{A}{25}\right)\right)$$

Weiterhin werden die Objekte, die innerhalb des Selektionsvolumens liegen, mittels *ComponentOverlapActors* berechnet und zurückgegeben.

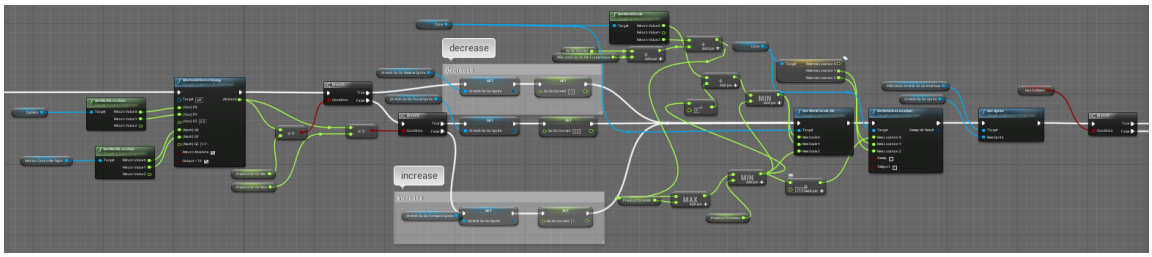


Abbildung 4.9.: Blueprint `stretch gogo interface`

Der Abstand zwischen Kamera und Motion Controller wird berechnet, um die korrekte Anzeige für das *Stretch Go-Go* Interface von *PRECIOUS* zu wählen.

Hierbei wird der Float `Go Go Current` verwendet, welcher mit `-1, 0` oder `1` gesetzt wurde. Dieser Faktor wird später mit dem `PRECIOUS Go Go Per Frame Value` multipliziert, wodurch sich dann durch Addition mit der Z-Koordinate und weiteren Rechenoperationen die neue Skalierung für den Kegel ergibt:

$$Z = max\left(CMin, \left(X_{prev} + (\texttt{C} \cdot \texttt{F})\right), CMax\right),$$
$$X, Y = min\left(\left(X_{prev} + (\tfrac{Z}{9})\right), Z\right).$$

Mit $C =$ `Go Go Current`, $F =$ `PRECIOUS Go Go Per Frame Value`, $CMin =$ `PRECIOUS Cone Min`, $CMax =$ `PRECIOUS Cone Max`. Hierbei handelt es sich um Variablen, die für die entsprechenden Werte in Unreal genutzt wurden.

Die relative Weltposition des Kegels wird mit $(D \cdot 50, 62)$ berechnet, was sich als gute Approximation herausgestellt hat.

Die Korrektur der Position erfolgt, wie zuvor bei der Handrotation, zum Ausgleich der Skalierung.

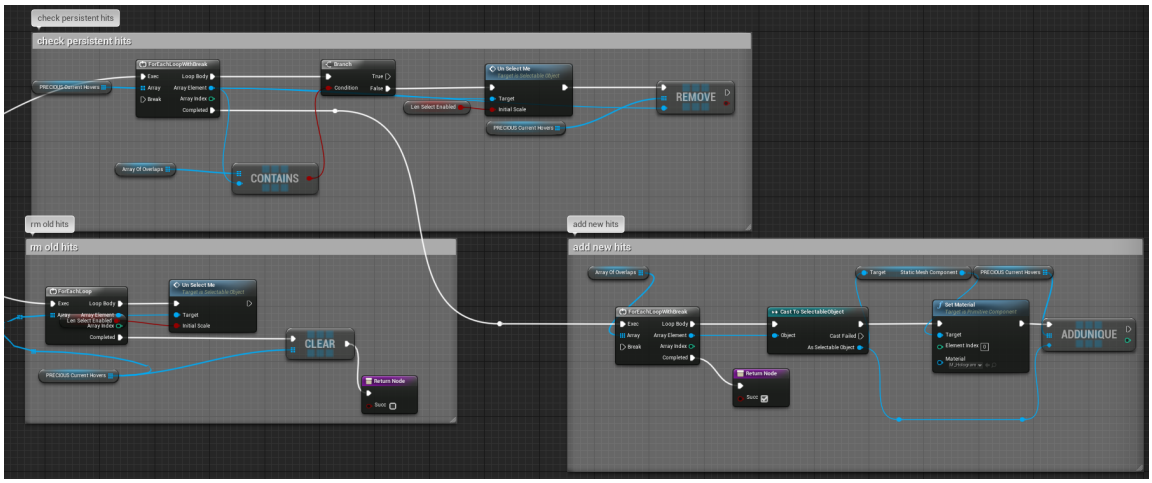


Abbildung 4.10.: Blueprint `check hits`

Letztendlich wird noch geprüft, ob es aktuell Überlappungen zwischen dem Kegel und (beliebigen, nicht weiter definierten) Objekten gibt (siehe Abbildung 4.10).
Wenn dem so ist, wird geprüft, ob noch immer die vorherigen Objekte im Kegel liegen, andernfalls werden diese entfernt.

In einem zweiten Durchlauf werden alle Objekte, die sich innerhalb des Bereichs befinden, gehovered. Falls es keine Kollision gab, werden alle zuvor hervorgehobenen Elemente aus dem entsprechenden Array gelöscht und die Funktion endet, ohne den `Succ`-Boolean zu setzen, wodurch in der übergeordneten Funktion der Counter `QHovers` nicht erhöht werden würde.

4.1.4.3. Hover für *VOTE*

Bei *VOTE* basiert die Hover-Funktion auf zwei spezifischen Funktionen:
Zunächst entfernt `VOTE Hover` das älteste Objekt aus der Queue (Abbildung 4.11).
Dieser Vorgang wird in Abhängigkeit von der relativen Controllergeschwindigkeit wiederholt, woraufhin entweder der `Hit Actor` oder ein Platzhalter in die Queue eingefügt wird.
Die Entscheidung dabei hängt davon ab, ob der erstere Parameter gültig ist.

`VOTE Return Selection` ist dafür zuständig, in der Queue zu zählen, welches Objekt am meisten “Stimmen” hat. Dabei wird der erste, zweite und dritte Platz berechnet und zurückgegeben.

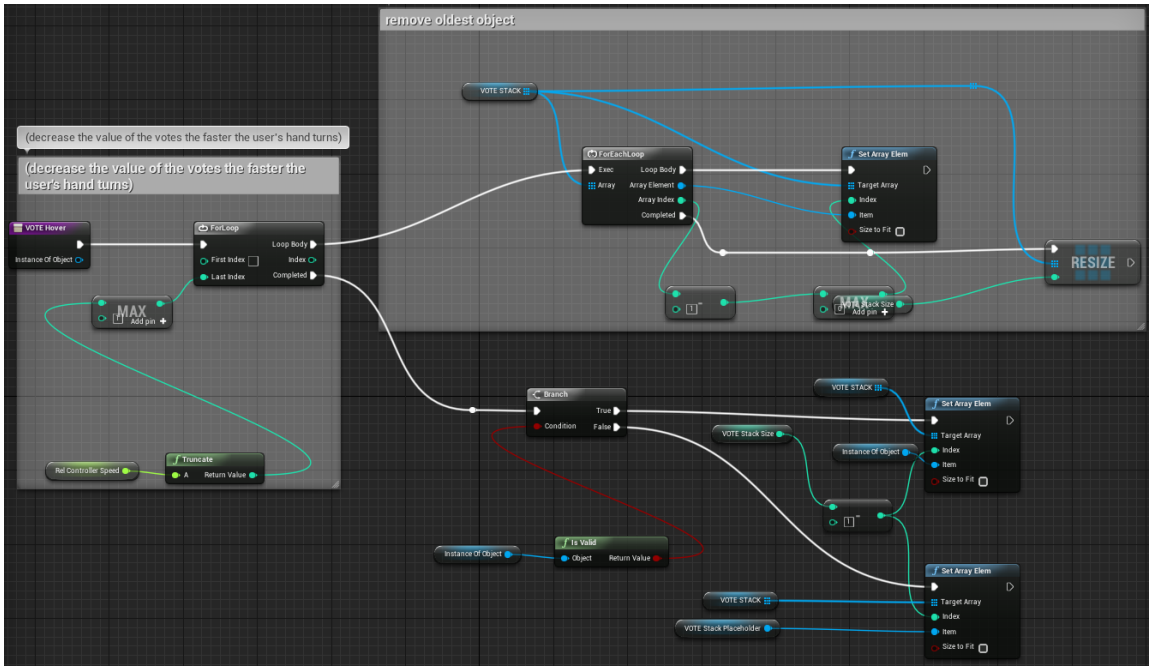


Abbildung 4.11.: Blueprint `VOTE Hover`

Es handelt sich dabei um zwei aufeinanderfolgende *ForEach*-Schleifen, die zunächst entweder jedes Objekt im Stack in die zur Speicherung der Bewertungen vorhandene Map[`SelectableObject`, `Integer`] aufnehmen oder deren Wert um `1` erhöhen, wenn bereits ein Eintrag vorhanden ist. Daraufhin werden die drei Objekte mit den höchsten Punktzahlen herausgefiltert (siehe Algorithmus 2).

Algorithmus 2 Grober Ablauf der Bewertung bei *VOTE*

```
1:  procedure VOTE Return Selection
2:      for each SelectableObject a ∈ VOTEStack do
3:          if a.IsPlaceholder then return false
4:          if VOTECounter contains a then
5:              VOTECounter.Set(a, VOTECounter.Get(a)+1)
6:          if !VOTECounter contains a then
7:              VOTECounter.Add(a, 1)
8:      for each SelectableObject b ∈ VOTEStack do
9:          if VOTECounter.Get(a) > VOTE_Temp_Max then
10:             VOTE Third ← VOTE Second
11:             VOTE Second ← VOTE_Temp_Max
12:             VOTE_Temp_Max ← b
        return VOTE_Temp_Max, VOTE Second, VOTE Third
```

4.1.4.4. Hover für *lenSelect*

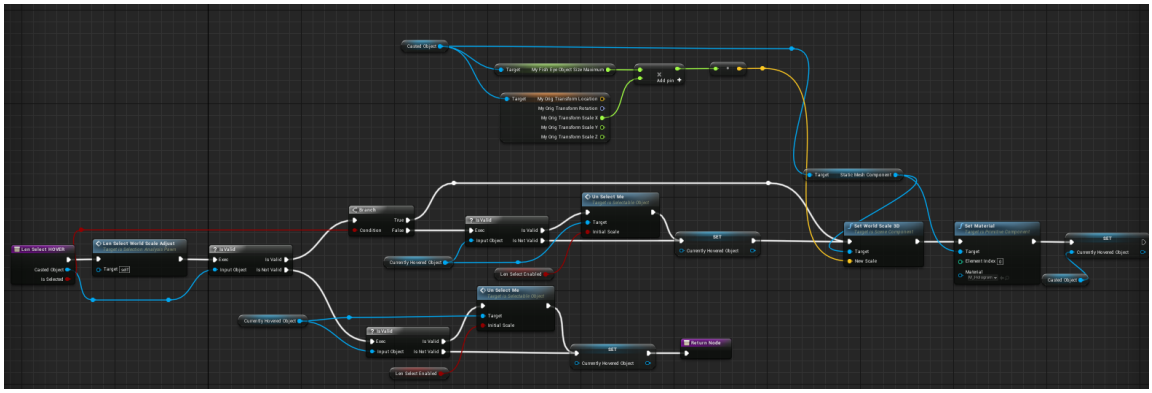


Abbildung 4.12.: Blueprint `lenSelect Hover`

`lenSelect Hover` prüft, ob es sich bei dem gecasteten Objekt, welches als Parameter übergeben wird, um das bereits zuvor selektierte handelt. In diesem Fall wird direkt die Größe auf das Maximum (normale Größe des Objekts multipliziert mit dem Maximalfaktor) gesetzt und das Hover-Material aktiviert. Selbiges passiert, wenn es sich nicht um das gleiche Objekt handelt, jedoch wird zuvor der Hovereffekt des vorherigen Objekts rückgängig gemacht.

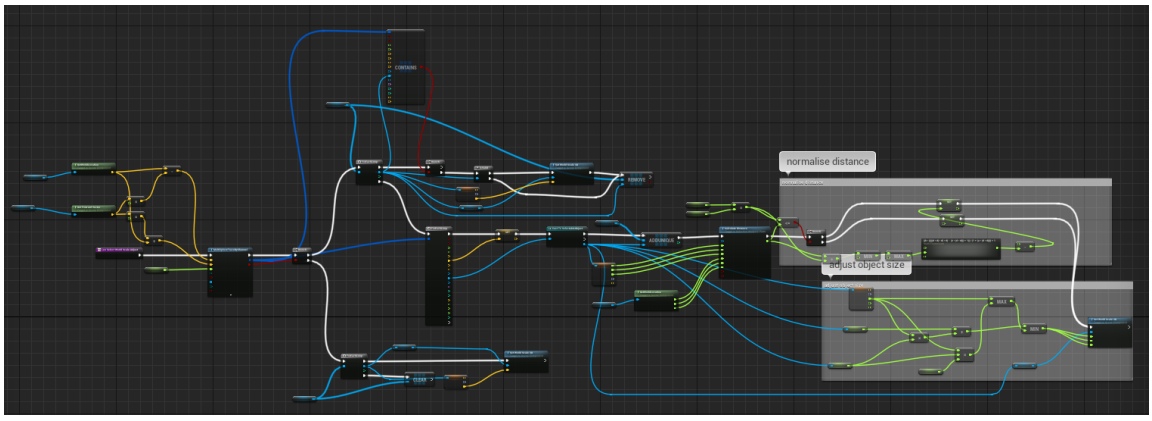


Abbildung 4.13.: Blueprint `lenSelect World Scale Adjust`

Zuvor wird noch die Funktion `lenSelect World Scale Adjust` aufgerufen, welche mittels eines *MultiSphereTraceByChannel* alle Objekte innerhalb einer Sphäre ermittelt. Diese Funktion arbeitet analog zum *LineTrace*, jedoch mit dem Unterschied, dass eine Sphäre anstatt einer Linie verwendet wird, um auf Kollision zu prüfen. Der Durchmesser dieser Sphäre ist der sogenannte `lenSelect Radius`, also der Radius, innerhalb dessen die Vergrößerung stattfinden soll (siehe Abbildung 4.14). Gibt es kein Ergebnis (keinen *Hit*), endet der Aufruf nachdem alle Objekte im Array `lenSelect Hovered Objects` wieder auf ihre Ursprungsgröße gesetzt und aus Letzterem entfernt wurden. Andernfalls wird derselbe

Vorgang für alle Objekte ausgeführt, die im aktuellen "Hit" nicht mehr enthalten sind, im Vorherigen aber vergrößert wurden.
Ein Hit bezeichnet dabei eine Kollision, also die Rückgabe des *LineTrace*, oder, in diesem Fall, des *MultiSphereTraceByChannel*.

Nach Abschluss dieser Operation wird schließlich der Abstand zwischen der Sphäre und dem Objekt berechnet, normalisiert und als Multiplikator bei der Berechnung der neuen Skalierung des Objekts verwendet. Die Skalierung des Objekts erfolgt prozentual in Abhängigkeit von einem Maximalwert (d_{max}). Der Multiplikator berechnet sich im Falle einer Kollision wie folgt:

$$s(Y) = 1 - \left(-1 \cdot \frac{(Y^4 - 2Y^2 + 1)}{(1 + (-1 \cdot Y^2))} + 1 \right) \quad \text{, falls } Y \leq d_{max},$$
$$s(Y) = 0 \quad \text{, sonst.}$$

mit

$$Y = max\left(\left(min\left(\left(\frac{D}{lenSelectRadius \cdot 20} \right), 1 \right) \right), 1 \right)$$

Wobei D den Abstand zwischen *Hitpoint* und dem Selektionsstrahl bezeichnet (siehe Abbildung 4.14). Sofern *lenSelect Radius* $> D$ ist, erfolgt keine Anpassung, da keine Kollision zwischen dem "Kreis" und dem Zielobjekt existiert.

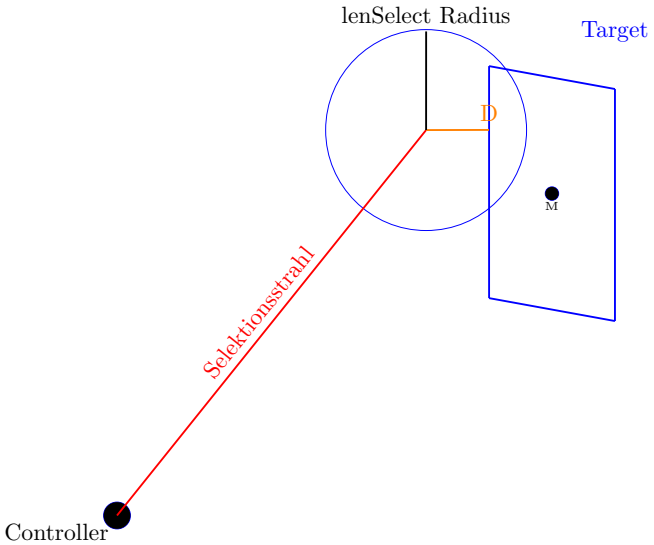


Abbildung 4.14.: Berechnung der Vergrößerung bei `lenSelect`

Der zuvor erwähnte Maximalwert d_{max} wiederum wird berechnet, indem der Durchmesser D_B der Bounding Box des Objekts berechnet wird. Dieser wird dann mit der Formel $D_{max} = \frac{GlobalMaxD}{D_B}$ normiert. Der Parameter `GlobalMaxD` ist dabei der Wert für den maximalen Durchmesser von Objekten, die vom Effekt betroffen sind (dieser wird in der Vorstudie festgelegt).
Sollte der errechnete Durchmesser größer als `GlobalMaxD` sein, so wird der Maximalwert auf die normale Größe des Objekts gesetzt, wodurch kein Effekt auftritt.

Dieser normierte Durchmesser D_n wird wiederum in die Formel zur Berechnung der maximalen Vergrößerung eingesetzt:

$$d_{max} = 2 \cdot \frac{(D_n^4 - 2 \cdot D_n^2 + 1)}{(1 + 2 \cdot D_n^2)} + 1.$$

Der Wert, der dadurch für d_{max} berechnet wurde, wird abschließend mit dem zuvor berechneten Faktor $s(Y)$ und dem "normalen" (initialen) Wert für die Objektgröße ("WorldScale") multipliziert und dann dem Objekt zugewiesen.

Somit wird der Vergrößerungseffekt in Abhängigkeit von der Entfernung zum Selektionsstrahl angewendet, sofern sich das Objekt innerhalb eines Radius um den *Hitpoint* befindet, welcher durch den *MultiSphereTraceByChannel* repräsentiert wird.

4.1.5. Selection Trigger

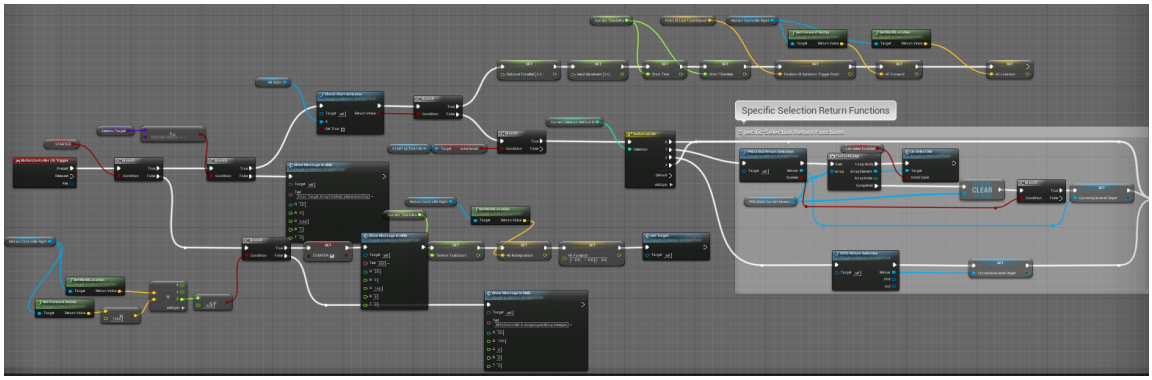


Abbildung 4.15.: Blueprint `Selektion Trigger`

Zunächst prüft die Funktion `Selection Trigger`, ob das Spiel "pausiert", also die letzte Selektion abgeschlossen wurde bzw. der Nutzer seinen Standpunkt verlassen hat. Dies führt zu einer Unterbrechung des Spiels bzw. der aktuellen Selektion, welche als Pause bezeichnet wird.

Wenn dem so ist (`STARTER==false`), wird geprüft, ob sich der Controller in der Ruheposition befindet. Diese ist wie folgt definiert: Wenn der Punkt, der sich aus *Position* + (1000 · *Forward Vector*) des Controllers ergibt, eine Z-Koordinate ≤ 500 aufweist, zeigt der Controller nach unten und der Proband befindet sich in jener Position.

Ist die Bedingung erfüllt, wird das Spiel fortgesetzt und das nächste Ziel vorgegeben, welches von `Set Target` zufällig aus den 12 vorgegebenen Objekten ausgewählt und markiert wird.

Auch führt diese Funktion die Korrektur des ID aus (Methode `Perform ID Correction`), indem der aktuelle Wert berechnet wird und das Objekt sukzessive vergrößert oder verkleinert wird, bis Soll- und Ist-ID bis auf eine Abweichung minimale übereinstimmen oder aber ein Iterationszähler überschritten wird.
Da die Objekte bereits initial vorangepasst sind, handelt es sich hier nur um eine Feinanpassung, die in kürzester Zeit und ohne spürbare Unterbrechung abläuft:

Algorithmus 3 Grober Ablauf der Korrektur des *index of difficulty*

```
1:  procedure PerformIDCorrection(Target)
2:      QIterations ← 0
3:      ID ← aktueller ID des Targets
4:      targetID ← gewünschter ID des Targets
5:      if targetID == 0.00 then return false
6:      if targetID == ID then return true
7:      QIterations ← QIterations+1
8:      while ID > targetID do
9:          resizeFactor ← 0.0001
10:         Setze WorldScale auf WorldScale + resizeFactor
11:         ID ← Berechne neuen ID des Targets
12:     while ID < targetID do
13:         resizeFactor ← -0.0001
14:         Setze WorldScale auf WorldScale + resizeFactor
15:         ID ← Berechne neuen ID des Targets
16:     if (ID <> targetID && QIteration >= 5) then go to 7
        return true
```

Wenn das Spiel nicht pausiert, so wird zunächst geprüft, ob es sich bei dem selektierten Objekt um den `Activator` handelt, welcher dann ggf. aktiviert wird. Auch werden in diesem Fall die Variablen, die für die Auswertung des Selektionstasks notwendig sind, entsprechend (zurück-) gesetzt.

Handelte es sich nicht um den `Activator` und ist dieser aktiviert, werden die methodenspezifischen "Return Selection"-Funktionen aufgerufen. Dies ist bei *PRECIOUS* und *VOTE* der Fall, da die anderen Methoden die Rückgabe des *LineTraceByChannel* verwenden.
Danach ist das Attribut `Currently Hovered Object` mit dem Objekt belegt, welches - unabhängig von der Methode - das aktuell selektierte repräsentiert.

Daher wird anschließend geprüft, ob es sich um das Zielobjekt handelt und dementsprechend die Anzahl der Fehler aktualisiert.

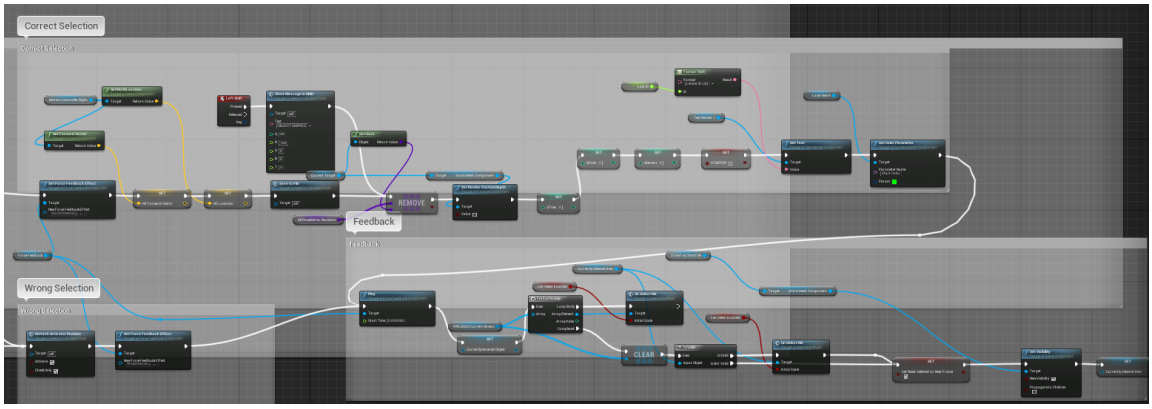


Abbildung 4.16.: Blueprint `Selektion Trigger`

Im Falle einer korrekten Selektion werden die Werte gespeichert, die Variablen zurückgesetzt und das Spiel pausiert. Anschließend (und auch im Falle einer Falschselektion) wird ein *Force Feedback Effect* in Form einer Vibration des Controllers ausgelöst und alle potentiell gehoverten Objekte zurückgesetzt.

War die Auswahl nicht korrekt, wird das Spiel nicht pausiert, sondern der Proband kann es weiter versuchen.

4.1.6. Change Distance

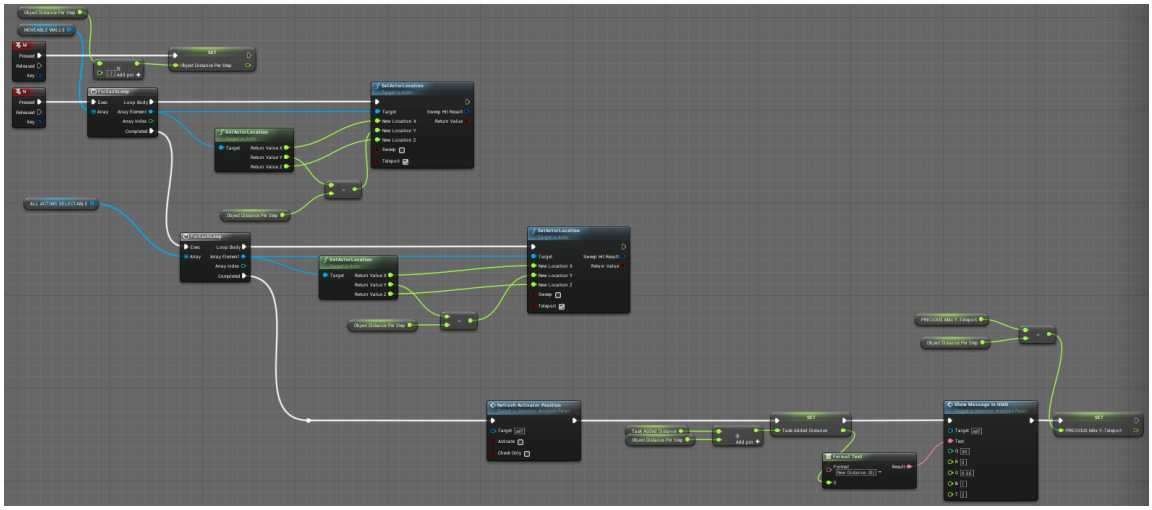


Abbildung 4.17.: Blueprint `Change Distance`

Die Änderung der Distanz zu den Objekten wird mithilfe von *ForEach*-Schleifen realisiert, die für alle Objekte von der Y-Koordinate einen per Variable `Object Distance Per Step` definierten Faktor subtrahieren. Der Faktor kann durch die Taste M negiert werden, wodurch eine Änderung der Distanz in beide Richtungen möglich wird.

4.1.7. Demo Mode

Ein Tastendruck auf D wechselt zwischen Demo- und Produktivmodus. In Ersterem werden Selektionen nicht gespeichert und kein Protokoll angelegt. Verwendung findet der Modus bei den ersten Testselektionen der Probanden.

4.1.8. Pre-Study Variants

Bei `Pre-Study Variants` (Abbildung 14, Seite A-8) handelt es sich um eine veraltete Funktion, die eine einfachere Durchführung der Vorstudie ermöglichte. Aus diesem Grund sind die Knoten der Funktion nicht mehr aktiv.

Es konnten mit der Methode `Pre Study Perform Var Change` mit den jeweiligen Tasten die verschiedenen Werte für die Variablen während des Spiels unterbrechungsfrei gesetzt werden.

4.2. SelectableObject

Das `SelectableObject` stellt die Mutterklasse aller Objekte in der virtuellen Umgebung dar und verfügt über die Methode `UnSelect Me`, welche das Objekt in den Ausgangszustand versetzt.
Dieser wird beim Programmstart gesetzt (siehe Abbildung 4.19).

Weiterhin verfügt es über Attribute wie den Boolean `I AM A TARGET` und spezifischen Definitionen für den ID.
Dazu werden zwei *float*-Werte verwendet: `ID_Config_Degree` bezeichnet den Winkel, in dem der `Activator` über dem Objekt stehen soll und `ID_Config_Distance` den Abstand beider.
Diese Werte sind für jedes Objekt vorab fix gesetzt, sodass diese in etwa einem gewünschten ID, `my_Target_ID`, entsprechen.

Wenn nun der Nutzer in der Ruheposition den Trigger betätigt, so wird mithilfe der Objektskalierung eine Approximation des IDs ausgeführt.

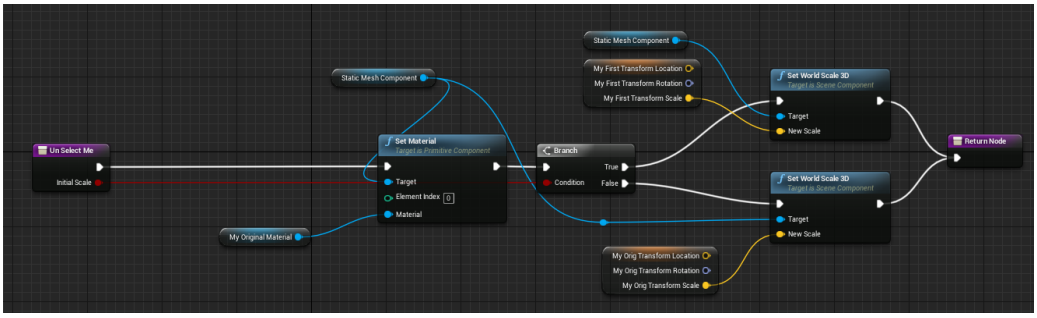


Abbildung 4.18.: Blueprint `UnSelect Me`

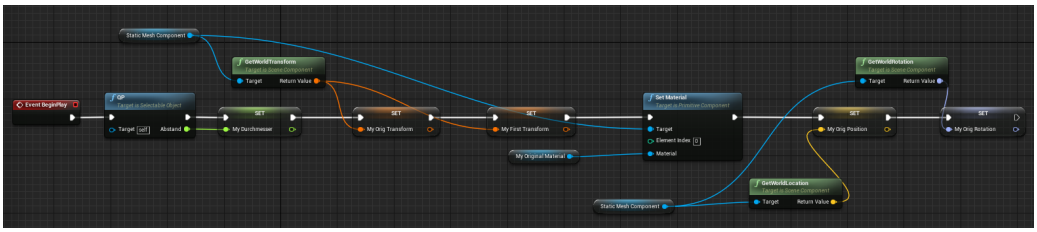


Abbildung 4.19.: Blueprint `SelectableObject`

4.3. Speicherung

Die Speicherung ist als CSV-Export in `C++` implementiert worden und befindet sich in Listing 7.1 in Anhang B.

Die Methode wird aus dem Blueprint `Save System State` aufgerufen und erhält die erzeugte Konkatenation von Variablen als Parameter.

Der entstehende String wird in einer Schleife aufbereitet, indem beispielsweise Punkte durch Kommata ersetzt werden, da von Unreal standardmäßig die englische Notation verwendet wird. Diese wird jedoch von der (zur Datenaufbereitung) genutzten Software Microsoft Excel nicht in jedem Fall als Zahl erkannt.
Um dadurch resultierenden Problemen vorzubeugen, wird erst danach der Wert an die Funktion `SaveStringToFile` des `FFileHelper` übergeben.
Letzterer ist Bestandteil der Unreal Engine.

5. Evaluation

Ziel der Studie war es herauszufinden, wie die vorgestellten Selektionsmethoden im direkten Vergleich abschneiden. Dabei wurden Effizienz und Effektivität in der zuvor entwickelten Evaluationsumgebung gemessen und abschließend bewertet. Weiterhin wurde mithilfe eines QUESI [20] - Fragebogens die Intuitivität der Interaktion untersucht.

Folgende Hypothesen wurden zu diesem Zweck aufgestellt:

Hypothese 1 (H1): *Die vorgestellten Selektionsmetaphern weisen statistisch signifikante Unterschiede in der durchschnittlichen Fehlerrate auf.*

Die `Fehlerrate` beschreibt die Anzahl falscher Selektionen in Form einer Betätigung der Trigger-Taste durch den Probanden, also jede Auswahl von Objekten, die nicht das Ziel der Aufgabe darstellen.

Hypothese 2 (H2): *Die vorgestellten Selektionsmetaphern weisen statistisch signifikante Unterschiede in der durchschnittlichen task completion time auf.*

Die durchschnittliche *task completion time* bezeichnet die Zeit, die zwischen der Aktivierung des Starters und der erfolgreichen Selektion des Ziels vergeht, gemessen in Sekunden.

Hypothese 3 (H3): *Die vorgestellten Selektionsmetaphern sind unterschiedlich intuitiv nutzbar.*

Das Ergebnis einer subjektiven Bewertung anhand eines QUESI-Fragebogens wird signifikante Unterschiede aufzeigen.

Die ersten beiden Hypothesen werden anhand der jeweiligen Messwerte geprüft, welche sowohl in der Vorstudie als auch der eigentlichen Studie die abhängigen Variablen darstellen. Bei Letzteren handelt es sich um die Messwerte, die sich durch den Einfluss der unabhängigen Variablen (in diesem Fall der ID und die jeweilige Selektionsmethode) ändern.

5.1. Studienaufbau

Ein HTC Vive HMD (*head-mounted-display*) mit einer Auflösung von 1520 x 1680 Pixeln pro Auge bei 90 Hertz wurde verwendet, um den Nutzer in eine virtuelle Szene eintauchen zu lassen.

Das head-mounted-display war mit einem Computer verbunden, welcher mit einer 8GB Gainward GeForce GTX 1080 Phoenix Golden Sample sowie einem Intel Core i7-7700K Prozessor mit 4 x 4,20 GHz ausgestattet war und unter Microsoft Windows 10 Education 64-bit mit dem Steam-Client sowie SteamVR lief. Weiterhin verfügte das System über 16 GB Arbeitsspeicher sowie eine Samsung M.2 SSD.

Die Teilnehmer standen in einem 3x3 Meter großen Bereich eines Raumes, der mit Klebeband gekennzeichnet war (siehe Abbildung 5.1). In diesem konnten sie sich frei bewegen. Um zu verhindern, dass die Probanden sich instinktiv in Richtung Ziel bewegen, wurde in diesem Fall automatisch gestoppt. Nachdem wieder der vorgegebene Bereich betreten wurde, startete die Aufgabe erneut.

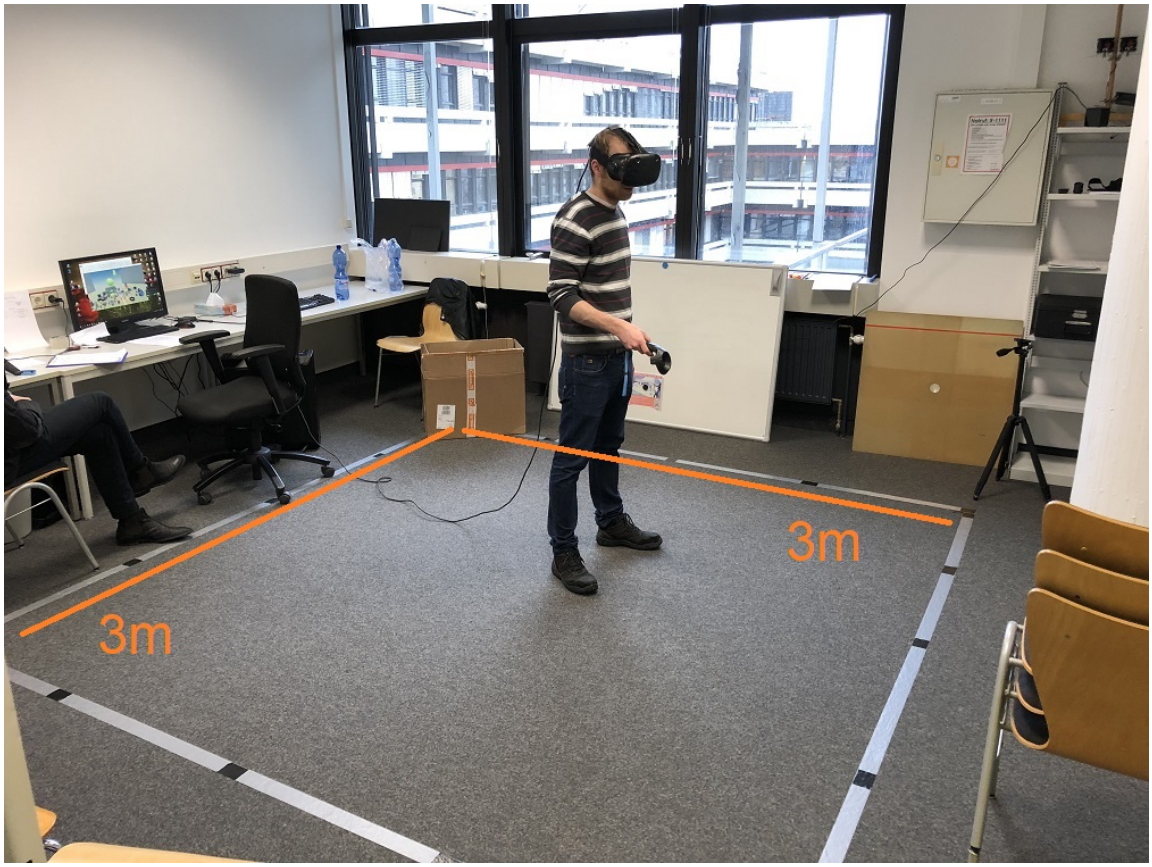


Abbildung 5.1.: Einblick in die Durchführung der Evaluation

5.2. Vorstudie

Vor der Durchführung der eigentlichen Evaluation fand eine Vorstudie statt, welche als Ziel die Optimierung spezifischer Variablen der verschiedenen Selektionsmetaphern verfolgte:

> ***VOTE* Stack Size** (`int`, Abbildung 3.7 auf Seite 14)
> *Die Anzahl der Objekte innerhalb der von `VOTE` mitgeführten Bewertungsliste (im Folgenden auch als "Queue" bezeichnet).*
>
> ***lenSelect* Radius** (`float`), Abbildung 3.5 auf Seite 13)
> *Radius in cm um den Strahl, in dem der Vergrößerungseffekt auftritt.*
>
> ***lenSelect* Max D** (`int`, Abbildung 5.5 auf Seite 37)
> *Maximaler Durchmesser vom Effekt betroffener Objekte.*
>
> **PRECIOUS View** (Abbildung 5.3 auf Seite 36)
> *Die unterschiedlichen Darstellungsmöglichkeiten der drei Zonen für die Stretch-Go-Go-Implementierung.*

Für die Vorstudie wurden 5 Probanden (3 männlich und 2 weiblich) zwischen 20 und 25 Jahren (Durchschnitt 23) rekrutiert. Darunter befand sich ein Linkshänder.

Die Vorstudie wurde im within-subject-design [8] durchgeführt. Jeder Proband hatte zunächst die Aufgabe, die vorgestellten Darstellungen für die Stretch-Go-Go-Zonen von *PRECIOUS* anhand einer fünfstufigen Likert-Skala zu bewerten. Anschließend wurden zufällige Objekte als Ziel vorgegeben und pro möglichem Variablenwert 15 Selektionen ausgeführt.

5.2.1. Ergebnis

Durch die Ergebnisse der Vorstudie konnten die Werte für die metapherspezifischen Variablen bestimmt werden, welche jeweils einen Kompromiss aus möglichst geringer Zeit in Kombination mit möglichst niedriger Fehlerrate darstellen.

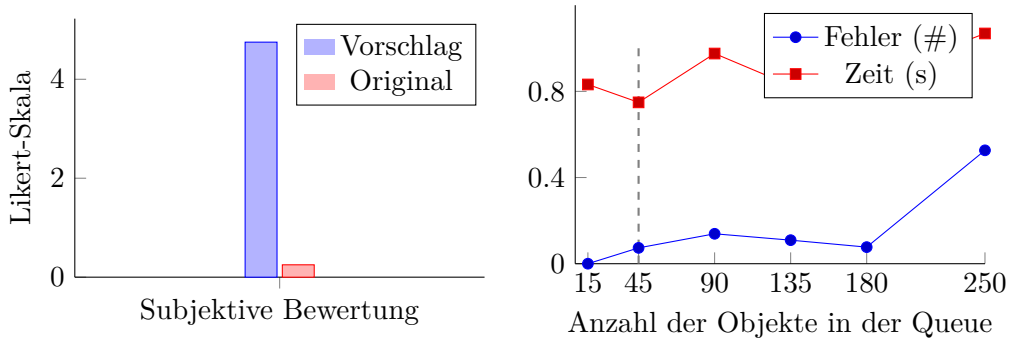


Abbildung 5.2.: Vorstudienergebnisse *PRECIOUS* und *VOTE*

Das Ergebnis für die Darstellung der *Go-Go Zonen* von *PRECIOUS* war eindeutig. Alle Probanden sagten aus, dass sie die Darstellungsoption, wie sie von Mendes et al. im Paper zu *PRECIOUS* vorgestellt wurde (Abbildung 5.3, links), missverständlich sei: Instinktiv wurde "rot" (und auch "gelb") auf der Anzeige mit "schlecht" gleichgesetzt.

Es wird daher der Alternativvorschlag (Abbildung 5.3, rechts) in der Evaluation verwendet.

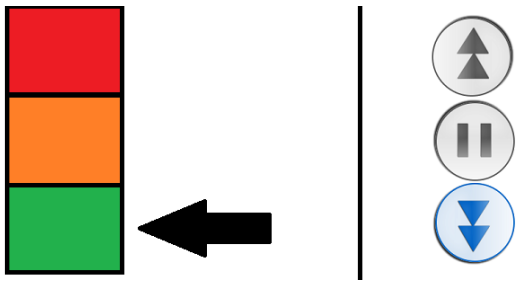


Abbildung 5.3.: Optionen für die Darstellung der *Stretch Go-Go*-Zonen bei *PRECIOUS*

Die *VOTE* -Bewertungsliste wurde mit einer Länge von `45` Elementen initialisiert, da sich mit diesem Wert die *task completion time* und Fehlerrate in Kombination innerhalb der niedrigsten möglichen (und somit besten) Werte bewegen.

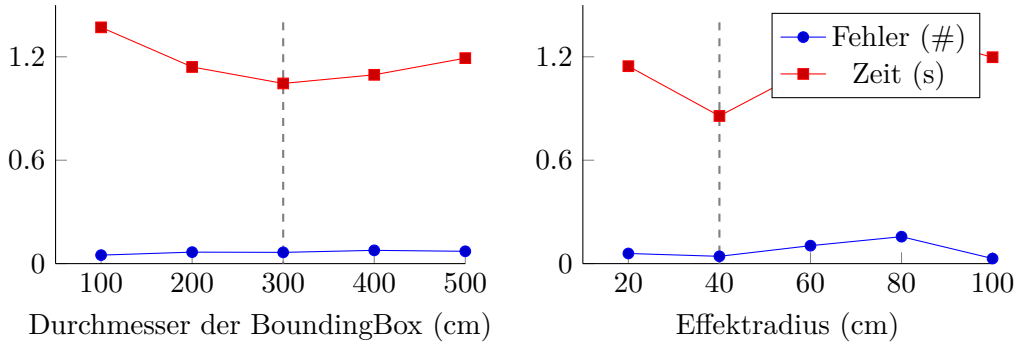


Abbildung 5.4.: Vorstudienergebnisse *lenSelect*

Bei *lenSelect* wurden aus selbigem Grund für den maximalen Objektdurchmesser, bis zu dem der Vergrößerungseffekt eintritt, `300 cm` und als maximaler Radius dieses Effektes `20 cm` festgelegt. Berechnet wurde der Durchmesser anhand der Bounding Box. Diese bezeichnet einen "Rahmen" um das Objekt und dient so als Approximation für die Außenpunkte.

Abbildung 5.5.: Beispielhafte Darstellung einer *Bounding Box*

5.3. Studie

Ziel der Studie war die Untersuchung der Auswirkung von Änderungen der folgenden Variablen auf die *Fehlerzahl* sowie *task completion time*:

Index of Difficulty (ID, `float`)
Der ID beschreibt die Komplexität der Selektionsaufgabe (Abbildung 3.3 auf Seite 11).

Selektionsmethode
Die in den vorherigen Kapiteln vorgestellten vier Methoden.

5.3.1. Teilnehmer

Mithilfe mehrerer Aushänge auf dem Campus und Ankündigungen in einigen Vorlesungen konnten Interessierte angesprochen werden, die sich über einen QR-Code (bzw. den zugehörigen Link) für einen Zeitslot im Terminplanungssystem *youcanbook.me* eintragen konnten.

Die Zeitplanung umfasste 15 Minuten pro Proband, zusätzlich wurden Termine als Puffer für Verzögerungen bzw. Teilnehmer ohne Anmeldung freigehalten.

28 Freiwillige (21 männlich und 7 weiblich) im Altersbereich von 18 bis 46 Jahren (Durchschnitt 23.6) nahmen an der Studie teil. Sie erhielten keine finanzielle Entschädigung.
Es waren 3 Linkshänder und 25 Rechtshänder darunter, die während des Experimentes die jeweils dominante Hand nutzten.

Zu Beginn erfolgte eine Einteilung in vier Gruppen, welche die Selektionsmethoden in unterschiedlichen Reihenfolgen durchliefen, um einen Lerneffekt zu verhindern. Die Zuordnung wurde fortlaufend anhand des Reihenfolge der Teilnahme vorgenommen.

Durchschnittlich dauerte ein Durchlauf inklusive Einweisung und Fragebogen 17 Minuten.

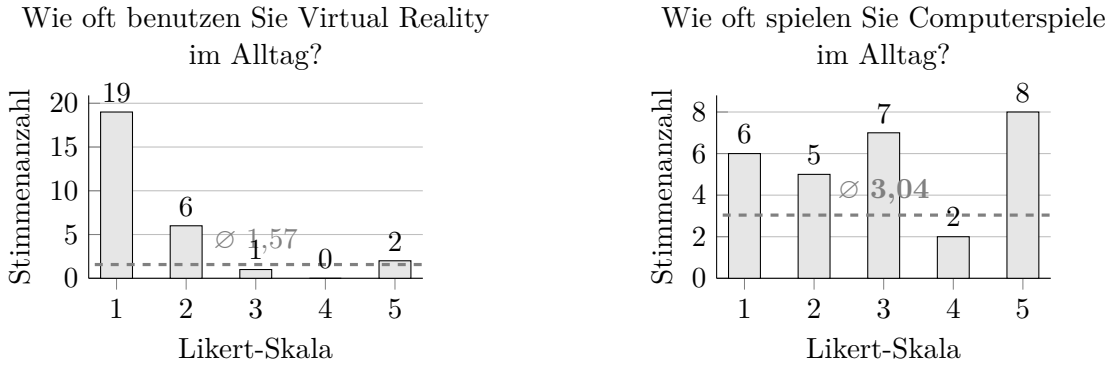


Abbildung 5.6.: Häufigkeitsverteilung der Nutzung von VR und Computerspielen

Aus Abbildung 5.6 wird ersichtlich, dass sowohl die Erfahrung im Umgang mit Virtual Reality (VR), als auch die Nutzung von Computerspielen stark variierte. Im Durchschnitt lässt sich sagen, dass die meisten Probanden wenig oder keine Erfahrung mit VR hatten und den Computer nicht täglich, aber regelmäßig zum Spielen benutzten.

5.3.2. Durchführung

Die Evaluation fand vom 30. - 31.01.2018 an der Universität Bremen im Zeitraum von 8:00 bis 15:00 Uhr statt.

Die eigentliche Durchführung erfolgte wie im folgenden Flowchart dargestellt, wobei für die Reihenfolge der Methoden zusätzlich ein lateinisches Quadrat verwendet wurde. Weiterhin hat die Hälfte der Teilnehmer die Selektionen zuerst mit normaler und danach mit erhöhter Distanz ausgeführt, bei der anderen Hälfte war es umgekehrt. Beides diente der Minimierung von Lerneffekten. Die Durchführung der Studie erfolgte im *within-subject-design*: Jeder Proband absolvierte daher nacheinander alle experimentellen Bedingungen oder Möglichkeiten.

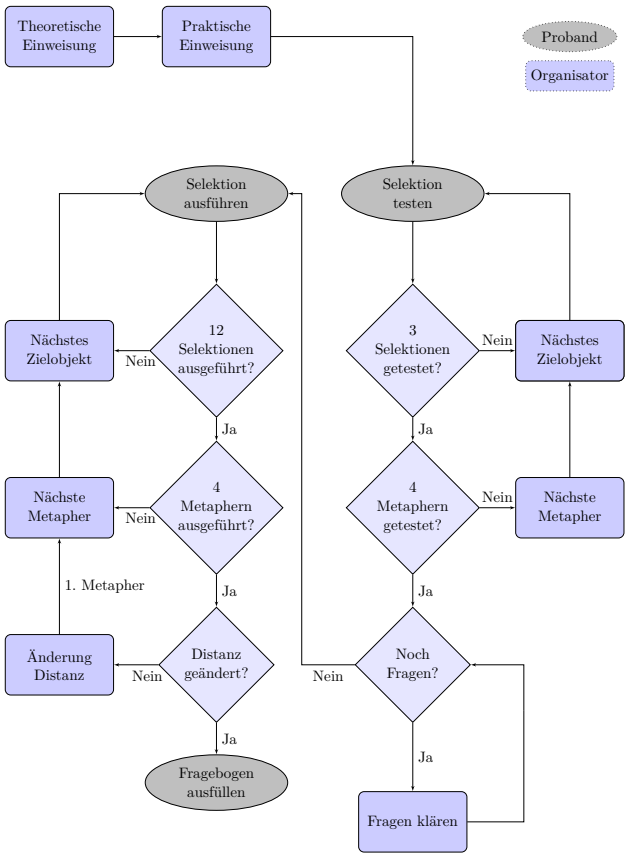


Abbildung 5.7.: Ablauf der Evaluation als Flowchart

5.3.3. Ergebnis

Es wurden für *Fehlerzahl* und *task completion time* insgesamt je 2688 Datensätze gesammelt. Nach der Entfernung von Ausreißern und ungültigen Werten verblieben 2411 gültige Messwerte zur weiteren Auswertung.

Da mehrfach die gleichen Probanden gemessen wurden, wurde eine ANOVA mit Messwiederholung (rmANOVA) verwendet [27][19]. Dabei handelt es sich um ein Verfahren zur Bewertung der statistischen Signifikanz von Varianzen.

Die dazu notwendige Normalverteilung darf gemäß zentralem Grenzwerttheorem, aufgrund ausreichend großer Stichprobe, angenommen werden [4].

Da eine Verletzung der Sphärizität vorlag, wurde nach Girden [17] das *Greenhouse–Geisser*-Verfahren zur Korrektur der Freiheitsgrade eingesetzt.

Signifikante Unterschiede zwischen den Methoden werden in den nachfolgenden Diagrammen durch rote Balken visualisiert.

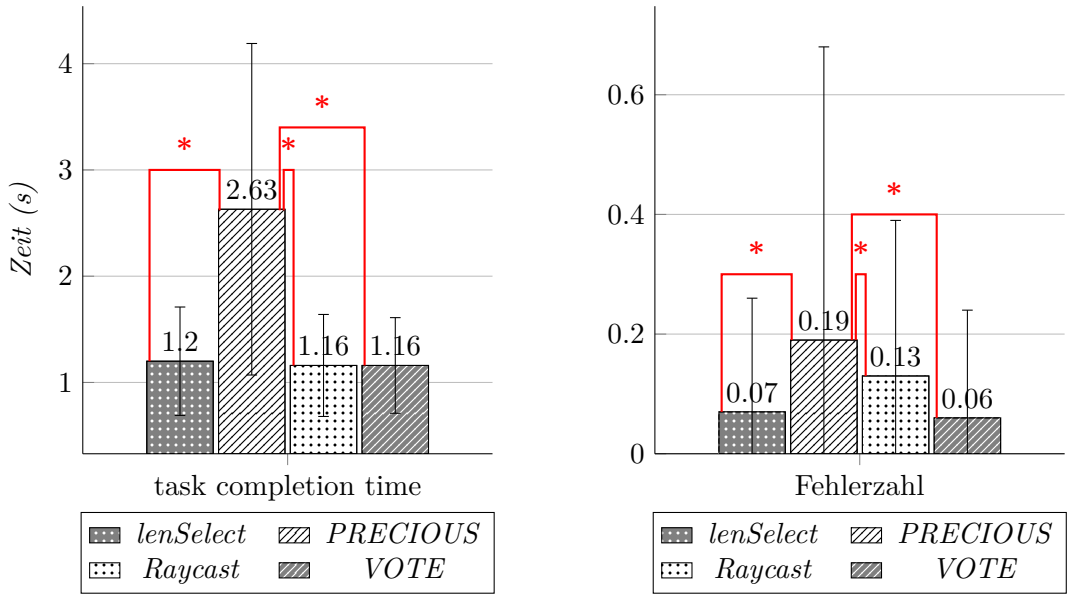


Abbildung 5.8.: Durchschnittliche *task completion time* und *Fehlerzahl*

Die Analyse zeigte, dass die Methoden in der *task completion time* und auch der *Fehlerzahl* über alle IDs hinweg statistisch signifikante Unterschiede aufwiesen,

$$F_{task\ completion\ time}\ (3,\ 2407) = 303.18,\ p_{task\ completion\ time} < 0.001$$
$$F_{Fehlerzahl}(3,\ 2407) = 21.92,\ p_{Fehlerzahl} < 0.001$$

lenSelect (M=1.20, SD=0.51), *Raycast* (M=1.16, SD=0.48) und *VOTE* (M=1.16, SD=0.45) waren signifikant schneller als *PRECIOUS* (M=2.63, SD=1.56). Unter den drei Erstgenannten gab es keine signifikanten Unterschiede.

Bei der Fehlerzahl ergab sich ein ähnliches Ergebnis, wobei die Probanden mit *VOTE* (M=0.06, SD=0.18), gefolgt von *lenSelect* (M=0.07, SD=0.19) und *Raycast* (M=0.13, SD=0.26) deutlich besser als mit *PRECIOUS* (M=0.23, SD=0.49) abschnitten. Die durchschnittlichen Messungen der beiden erstgenannten Methoden wiesen zusätzlich signifikant weniger Fehler als *Raycast* auf.

Der Vergleich der Werte in Abhängigkeit vom *index of difficulty* kommt zu einem ähnlichen Ergebnis. Die Aufzählungen der Methoden innerhalb der folgenden Auswertung erfolgt jeweils in absteigender Reihenfolge (je später die Methode in der Liste, desto besser das Ergebnis).

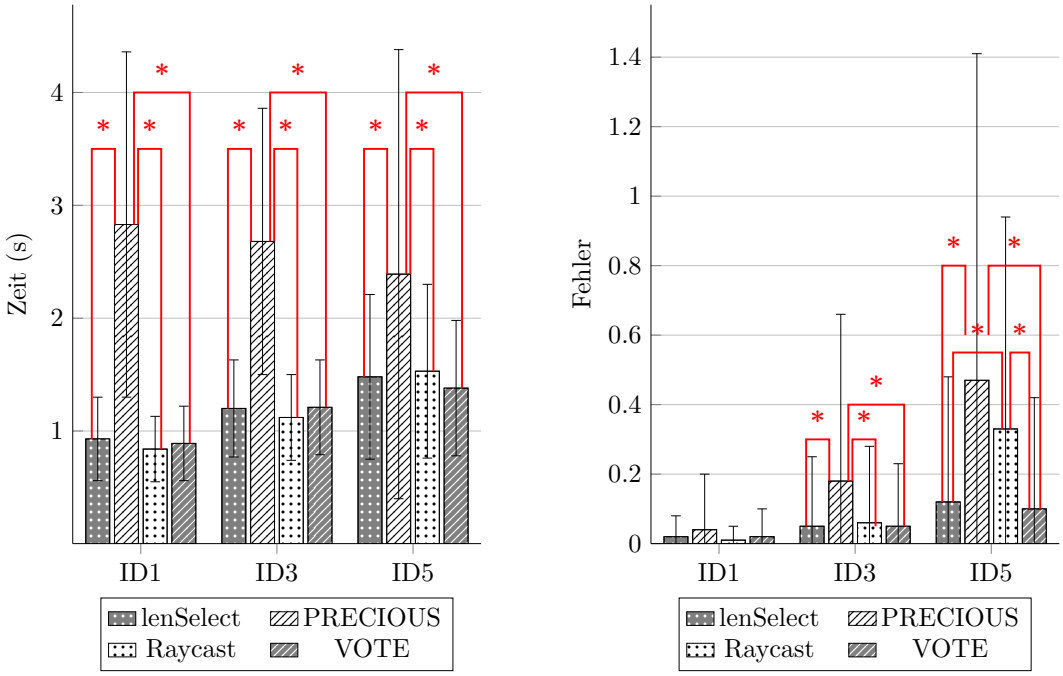


Abbildung 5.9.: *task completion time* und *Fehlerzahl* nach ID

ID 1 ($N = 802$)

$$F_{task\ completion\ time}\ (3,\ 798) = 252.09,\ p_{task\ completion\ time} < 0.001$$
$$F_{Fehlerzahl}(3,\ 798) = 1.86,\ p_{Fehlerzahl} = 0.135$$

PRECIOUS (M=2.81, SD=1.61) war signifikant langsamer als *lenSelect* (M=0.93, SD=0.38), *VOTE* (M=0.89, SD=0.34) und *Raycast* (M=0.85, SD=0.30), wobei es zwischen den letzteren Dreien keinen signifikanten Unterschied gab. In der Fehlerzahl gab es bei ID 1 keinerlei signifikante Unterschiede.

ID 3 ($N = 802$)

$$F_{task\ completion\ time}\ (3,\ 798) = 192.3,\ p_{task\ completion\ time} < 0.001$$
$$F_{Fehlerzahl}(3,\ 798) = 8.71,\ p_{Fehlerzahl} < 0.001$$

Die *task completion time* wies, äquivalent zum vorherigen Ergebnis, signifikante Unterschiede zwischen *PRECIOUS* (M=2.69, SD=1.36) und *VOTE* (M=1.21, SD=0.44), *lenSelect* (M=1.19, SD=0.44) sowie *Raycast* (M=1.12, SD=0.39) auf.
Auch die Fehlerzahl wies signifikante Unterschiede zwischen *PRECIOUS* (M=0.18, SD=0.51) und den Methoden *Raycast* (M=0.05, SD=0.23), *lenSelect* (M=0.05, SD=0.21) und *VOTE* (M=0.04, SD=0.21) auf. Zwischen den letzteren Dreien gab es weiterhin keine statistisch signifikanten Unterschiede.

ID 5 ($N = 807$)

$$F_{task\ completion\ time}\ (3,\ 803) = 26.55,\ p_{task\ completion\ time} < 0.001$$
$$F_{Fehlerzahl}(3,\ 803) = 16.32,\ p_{Fehlerzahl} < 0.001$$

Auch hier fanden sich signifikante Unterschiede zwischen *PRECIOUS* (M=2.39, SD=2.22) und *Raycast* (M=1.53, SD=0.88), *lenSelect* (M=1.48, SD=0.73) sowie *VOTE* (M=1.38, SD=0.69).
In der Fehlerzahl unterscheiden sich *lenSelect* (M=0.12, SD=0.36) und *VOTE* (M=0.1, SD=0.32) zwar nicht signifikant voneinander, aber von *PRECIOUS* (M=0.47, SD=0.94) und *Raycast* (M=0.33, SD=0.61).

5.3.3.1. Einfluss von Erfahrungen

Im Fragebogen wurde die Häufigkeit der Nutzung von Virtual Reality auf einer fünfstufigen Likert-Skala abgefragt.

Bei der Auswertung der durchschnittlichen Messwerte in Abhängigkeit von dieser Angabe wurde deutlich, dass mit zunehmender Erfahrung tendenziell bessere Werte einhergingen.

Die Tatsache, dass bei der maximal möglichen Angabe (4) im Vergleich zur vorherigen ein Anstieg verzeichnet wurde, hing damit zusammen, dass lediglich zwei Probanden ihre Erfahrung derart hoch einschätzten,und dann verhältnismäßig schlecht abschnitten.

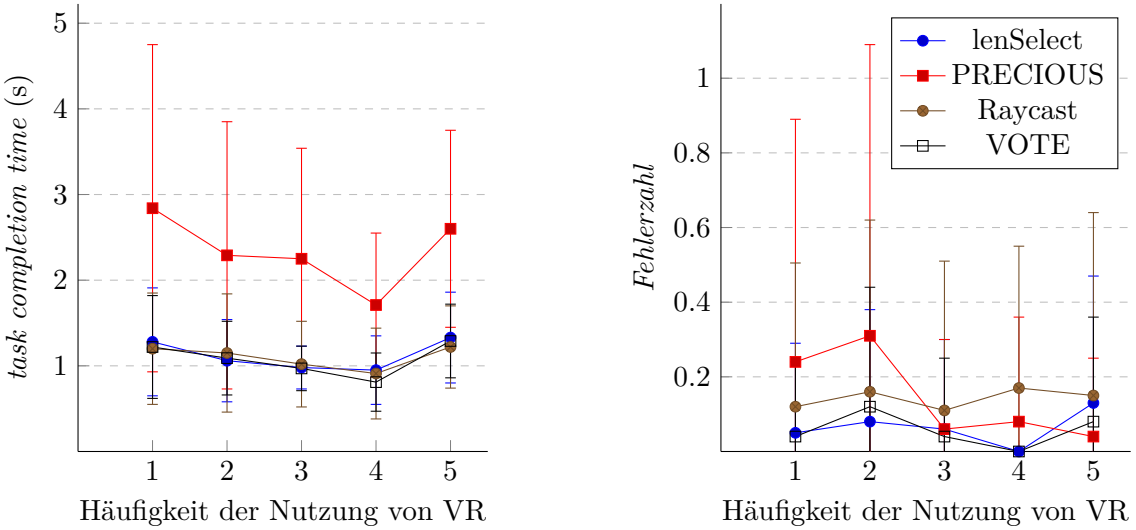


Abbildung 5.10.: Vergleich der Ergebnisse nach Häufigkeit der Nutzung von VR

Da der Großteil der Probanden wenig Erfahrung aufwies (im Durchschnitt 1.57 auf der Likert-Skala), es aber auch erfahrenere Teilnehmer gab, kommt es zwischen den Messungen zu einer hohen Varianz.

5.3.3.2. Vergleich von Kopf- und Handbewegungen

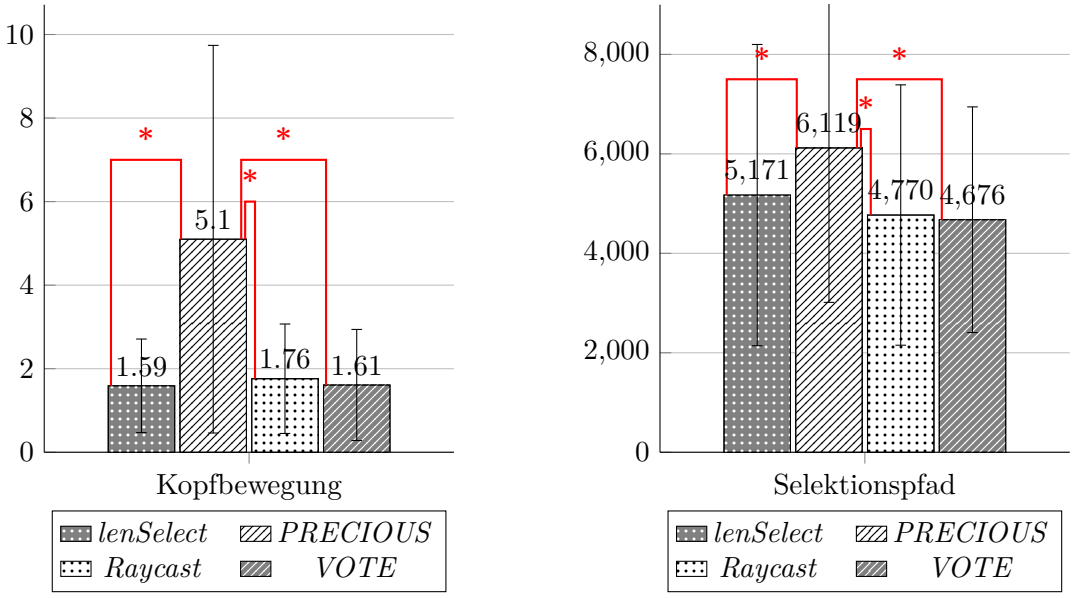


Abbildung 5.11.: Vergleich der Kopfbewegung und des Selektionspfades

Es wurden bei jeder Selektion Werte für die Kopf- sowie Controllerbewegung gesammelt.

Bei Ersterer handelte es sich um die Summe der Distanzen des sogenannten `Forward Vector`, wodurch sich ein Maß für die Kopfbewegung im Sinne seiner Rotation ergab. Der `Forward Vector` gibt den Vektor an, welcher die Blickrichtung des Probanden repräsentiert. Er hat die Länge 1 und bezeichnet somit in jedem Frame einen Punkt vor der HTC Vive, dessen Abstand zum vorherigen bis zum Zeitpunkt der korrekten Selektion summiert wurde.

Bei dem Selektionspfad handelt es sich um den Betrag des dreidimensionalen Pfades, welcher durch die Summe der Distanzen zwischen den *Hits* des *LineTrace* beschrieben wird. Die statistische Bewertung ergab signifikante Unterschiede:

$$F_{Kopfbewegung}\ (3,\ 2407) = 272.370,\ p_{Kopfbewegung} < 0.001$$
$$F_{Selektionspfad}(3,\ 2407) = 33.99,\ p_{Selektionspfad} < 0.001$$

Die durchschnittliche Kopfbewegung war am Höchsten bei *PRECIOUS* (M=5.10, SD=4.64) und unterschied sich signifikant von *Raycast* (M=1.76, SD=1.32), *VOTE* (M=1.61, SD=1.34) sowie *lenSelect* (M=1.59, SD=1.13).

Auch der Selektionspfad unterschied sich signifikant, dieser bei war *PRECIOUS* (M=6119, SD=2618) wieder am längsten, gefolgt von *lenSelect* (M=5171, SD=3029), *Raycast* (M=4770, SD=2618) und *VOTE* (M=4676, SD=2268).

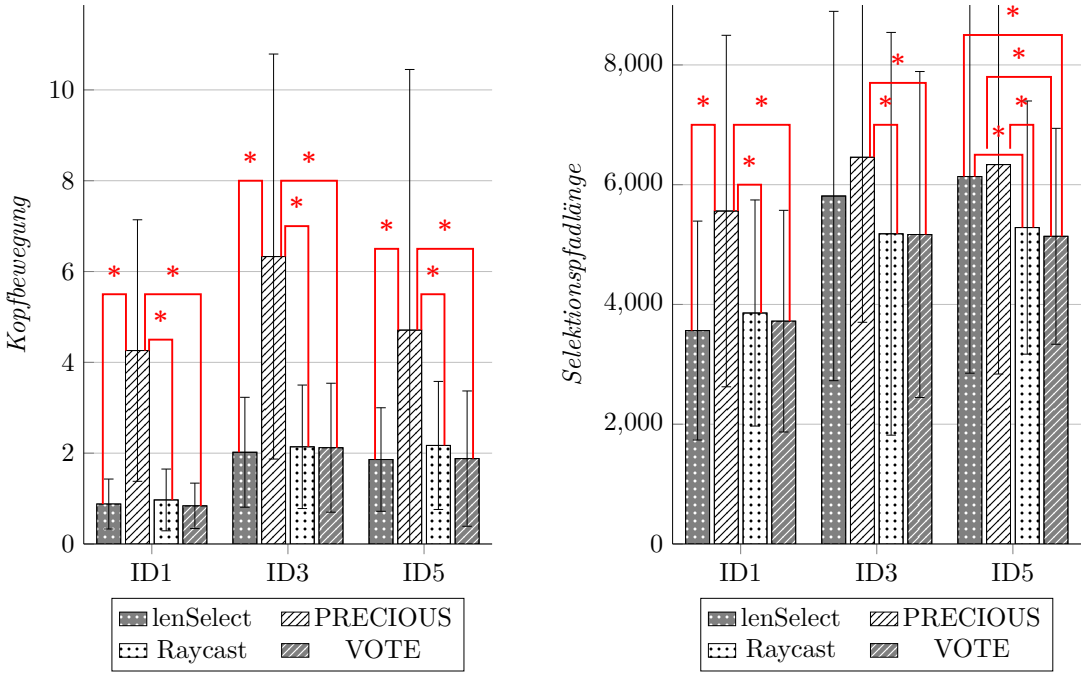


Abbildung 5.12.: Kopfbewegung und Selektionspfad nach ID

ID 1 ($N = 802$)

Sowohl bei der Kopfbewegung als auch beim Selektionspfad gab es signifikante Unterschiede zwischen *PRECIOUS* (M_S=5560, SD_S=2953, M_K=4.26, SD_K=2.88) und *Raycast* (M_S=3857, SD_S=1888, M_K=0.97, SD_K=0.68), *lenSelect* (M_S=3564, SD_S=1828, M_K=0.88, SD_K=0.55) sowie *VOTE* (M_S=3721, SD_S=1851, M_K=0.84, SD_K=0.50).

$$F_{Kopfbewegung}\ (3,\ 798) = 36.41,\ p_{Kopfbewegung} < 0.001$$
$$F_{Selektionspfad}\ (3,\ 798) = 242.87,\ p_{Selektionspfad} < 0.001$$

ID 3 ($N = 802$)

Bei der Kopfbewegung blieb es bei obiger Aussage, allerdings unterschieden sich nur noch *Raycast* (M_S=5180, SD_S=3363, M_K=2.14, SD_K=1.36) und *VOTE* (M_S=5168, SD_S=2722, M_K=2.12, SD_K=1.42) im Selektionspfad signifikant von *PRECIOUS* (M_S=6458, SD_S=2756, M_K=6.33, SD_K=4.46). *lenSelect* (M_S=5811, SD_S=3084, M_K=2.02, SD_K=1.21) hingegen wies zu keiner der anderen Methoden signifikante Unterschiede auf.

$$F_{Kopfbewegung}\ (3,\ 798) = 142.59,\ p_{Kopfbewegung} < 0.001$$
$$F_{Selektionspfad}(3,\ 798) = 8.46,\ p_{Selektionspfad} < 0.001$$

ID 5 ($N = 807$)

Zusätzlich zum Ergebnis von ID 3 unterschieden sich *VOTE* (M_S=5138, SD_S=1803, M_K=1.88, SD_K=1.49) und *Raycast* (M_S=5284, SD_S=2114, M_K=2.17, SD_K=1.41) von *lenSelect* (M_S=6136, SD_S=3284, M_K=1.86, SD_K=1.14) und *PRECIOUS* (M_S=6336, SD_S=3498, M_K=4.71, SD_K=5.74) nicht mehr signifikant in der Länge der Selektionspfade.

$$F_{Kopfbewegung}\ (3,\ 803) = 38.68,\ p_{Kopfbewegung} < 0.001$$
$$F_{Selektionspfad}(3,\ 803) = 1.86,\ p_{Selektionspfad} = 0.135$$

5.3.3.3. Lerneffekt

Zur Beurteilung der Existenz eines Lerneffekts wurden die Messwerte für *task completion time* sowie *Fehlerzahl* in Abhängigkeit von einem Selektionscounter, welcher den Durchlauf (1-12) angibt, ausgewertet.

$$F_{task\ completion\ time}\ (13,\ 2397) = 0.64,\ p_{task\ completion\ time} = 0.820$$
$$F_{Fehlerzahl}(13,\ 2397) = 1.31,\ p_{Fehlerzahl} = 0.195$$

Ein Lerneffekt kann ausgeschlossen werden, da es weder in der *task completion time* noch in der *Fehlerzahl* im Laufe der zwölf durchgeführten Selektionen signifikante Unterschiede gab.

5.4. QUESI

Nach der Evaluation wurde für alle Methoden ein Fragebogen nach QUESI [20] von den Probanden beantwortet. Dieser diente der Messung der subjektiven Konsequenzen intuitiver Nutzung.

Eine digitale Version wurde mithilfe von *Google Documents* erstellt und in 4 äquivalente Seiten unterteilt.

Auf der letzten Seite wurden persönliche Daten des Probanden wie Probandengruppe, Körpergröße, Geschlecht, Alter, Beruf, Links-/Rechtshänder sowie Häufigkeit der Nutzung von VR und Computerspielen abgefragt.

Jede Seite des Fragebogens enthielt 14 Thesen, die auf einer fünfstufigen Likert-Skala (1 = trifft gar nicht zu, 5 = trifft völlig zu) zu bewerten waren. Ein höherer Wert geht dabei mit einer erhöhten Wahrscheinlichkeit für intuitive Benutzung einher [20].

5.4.1. Ergebnis

Der QUESI-Fragebogen zeigte ein ähnliches Ergebnis, wie die vorherigen Vergleiche: *PRECIOUS* (M=3.67, SD=0.02) wurde schlechter (weniger intuitiv) bewertet als die anderen Methoden. Zwischen diesen gab es lediglich kleine Unterschiede: *Raycast* (M=4.62, SD=0.03) führte die Bewertung an, gefolgt von *VOTE* (M=4.57, SD=0.02) und *lenSelect* (M=4.41, SD=0.01).

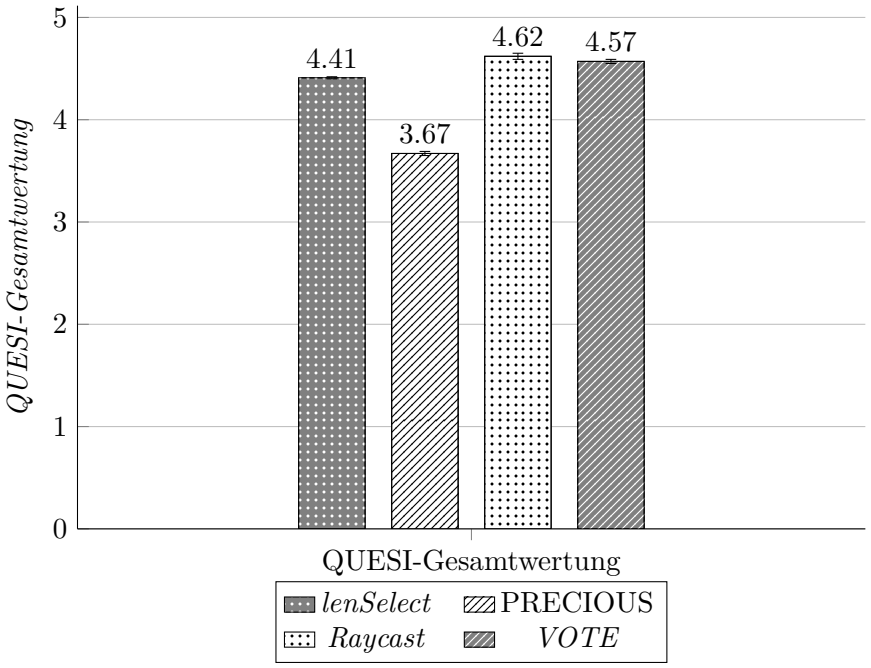


Abbildung 5.13.: Ergebnis der QUESI-Auswertung

Subskala	***lenSelect***		***PRECIOUS***		***Raycast***		***VOTE***	
Wahrgenommene(r)...	M	SD	M	SD	M	SD	M	SD
Kognitive Beanspruchung	4,38	0,01	3,53	0,00	4,64	0,03	4,65	0,02
Zielerreichung	4,44	0,00	3,92	0,06	4,56	0,05	4,5	0,03
Lernaufwand	4,45	0,01	3,54	0,00	4,70	0,00	4,60	0,01
Vertrautheit/Vorwissen	4,45	0,03	3,70	0,03	4,70	0,02	4,61	0,03
Fehlerrate	4,32	0,00	3,66	0,00	4,54	0,02	4,5	0,00
QUESI-Gesamtwertung	**4,41**	0,01	**3,67**	0,02	**4,62**	0,03	**4,57**	0,02

Tabelle 5.1.: Auswertung der Fragebögen nach QUESI

6. Zusammenfassung

Es hat sich herausgestellt, dass die Messwerte eine ausgeprägte Varianz aufwiesen. Nach Analyse der in Kapitel 5.3.1 vorgestellten Rahmendaten der Evaluation schien dies aus der nicht-normalverteilten Erfahrung der Probanden in Bezug auf Virtual Reality und insbesondere auch bei der Häufigkeit der Nutzung von Computerspielen zu resultieren.

Die Hypothesen aus Kapitel 5 konnten teilweise bestätigt werden.
Alle geprüften Methoden wiesen signifikant bessere Werte als *PRECIOUS* in der *Fehlerzahl* und *task completion time* auf.

Bei der Anzahl der Fehler stellte sich außerdem heraus, dass *VOTE* und *lenSelect* signifkant bessere Ergebnisse als *Raycast* lieferten.

Die Auswertung der Intuitivität anhand des *QUESI*-Fragebogens kam im Kern zum gleichen Ergebnis.

Keine statistisch signifikanten Unterschiede konnten hingegen bei der Bearbeitungszeit zwischen den Methoden mit unmittelbarer Selektion gefunden werden. Bei der Fehlerzahl war dies zwischen *Raycast* und den beiden anderen der Fall: *lenSelect* und *VOTE* stellen demnach eine Alternative zueinander und eine statistisch signifikante Verbesserung von *Raycast* dar.

Die Ergebnisse der Werte für die *Fehlerzahl* sowie *task completion time* bestätigen den Vergleich zwischen *Raycast* und *VOTE* von Moore dadurch, dass dieser in der Kernaussage zum gleichen Ergebnis kommt. Das Résumé von Mendes et al. konnte aufgrund der Verwendung anderer Vergleichsmethoden nicht direkt herangezogen werden.

Es fiel auf, dass *PRECIOUS* in den hier ausgeführten Vergleichen verhältnismäßig schlecht abschnitt. Das QUESI-Ergebniss zeigt, dass diese Methode auch weniger intuitiv als die anderen ist. Es lässt sich vermuten, dass die Probanden lediglich zu wenig Erfahrung im Umgang mit ihr entwickeln konnten.

Zwar hat sich gezeigt, dass bereits vorhandene Erfahrung im Umgang mit VR einen erheblichen Einfluss auf die Ergebnisse hat, jedoch war im Laufe der Evaluation kein statistisch signifikanter Lerneffekt messbar.

Der Vergleich von Kopfbewegung und Selektionspfadlänge zeigte außerdem, dass *PRECIOUS* im Vergleich zu den anderen Methoden im Nachteil zu sein schien.

Über alle IDs hinweg wies die Methode die längsten Pfade und das höchste Maß an Kopfbewegungen auf. Bei Betrachtung dieses Vergleichs pro ID fiel außerdem auf, dass sich *lenSelect* bei der Pfadlänge ab ID 3 nicht mehr deutlich von *PRECIOUS* unterschied, *Raycast* und *VOTE* hingegen auch im Vergleich zwischen ID 3 und 5 keine deutliche Steigerung mehr verzeichneten und so signifikant kürzere Wege als *lenSelect* und insbesondere *PRECIOUS* aufwiesen. Dies schien daran zu liegen, dass die Interaktion als solche bei den letzteren Methoden umfangreicher ausfiel, insbesondere aufgrund der iterativen Verfeinerungsschritte bei *PRECIOUS*. Da *lenSelect* hingegen keine Iteration verwendet, lag die Vermutung nahe, dass der Vergrößerungseffekt bei kleineren Zielen zur Verdeckung dieser durch umliegende Hindernisse führte und so für den deutlich erhöhten Pfad sorgte.

Der Vergleich der Methoden nach ID hat ergeben, dass die Messungen der *Fehlerzahl* bei *PRECIOUS* mit steigendem ID rapide nach oben gingen, während *lenSelect* und *VOTE* eine niedrige Steigungsrate aufwiesen. *Raycast* verzeichnete aber bei ID 5 auch einen sprunghaften Anstieg.

Bei der *task completion time* war dieser Verlauf bei *PRECIOUS* hingegen sogar rückläufig, während die anderen Methoden leichte Steigerungen zeigten, aber dennoch signifikant niedrigere Werte erreichten.

Zusammenfassend lässt sich sagen, dass die Wahl der zu verwendenden Selektionsmethode unter anderem von der Art (sprich dem ID) der Ziele abhängt. Verwendete man Ziele mit einem ID von eins, so gab es in Bezug auf die *Fehlerzahl* keinerlei statistisch signifikante Unterschiede.

Da sich hier lediglich *PRECIOUS* mit längerer *task completion time* von den anderen unterschied, ist es, abgesehen von diesem Unterschied, faktisch egal, welche Methode zum Einsatz kommt, sofern die Zeit keine besondere Priorität hat.

Bei den anderen verwendeten IDs drei und fünf waren hingegen sowohl im Sinne der Zeit als auch der Fehler *lenSelect* und *VOTE* die “besten” Methoden. Da zwischen diesen jedoch kein signifikanter Unterschied festgestellt werden konnte, stellen sie eine Alternative zueinander dar.

Die Bewertung der Methoden durch die Probanden hat ebenfalls gezeigt, dass sie *PRECIOUS* als schwieriger (weniger intuitiv) empfanden, während die Konkurrenten mit ihren Gesamtpunkten nahe am Maximalwert des Fragebogens lagen.

7. Fazit

Ziel der vorgelegten Bachelorarbeit war es, vier neuartige Selektionsmetaphern zu implementieren und miteinander zu vergleichen. Dies wurde umgesetzt, indem eine Windows-Anwendung entwickelt und mithilfe von freiwilligen Probanden eine Evaluation durchgeführt wurde.
Dazu lässt sich die Arbeit in zwei Ergebnisse unterteilen:

Die Software wurde erfolgreich implementiert.
Hierbei musste ein Konzept erarbeitet werden, nach welchem das Programm ablaufen sollte. Weiterhin war es das Ziel, dieses in der Unreal Engine als VR-Evaluationsumgebung umzusetzen. Beide Teilziele wurden erfolgreich abgeschlossen. Die entstandene Software ermöglicht die einfache Durchführung der geplanten Evaluation. Dabei gibt es Vielzahl von Objekten, zwischen denen sich 12 Ziele befinden. Diese Ziele werden in zufälliger Reihenfolge vorgegeben und müssen vom Probanden selektiert werden. Der Evaluator hat die Möglichkeit, mittels einer Tastatur (Tastenbelegungen siehe Abbildung 19 auf Seite D-4) zwischen den Selektionsmethoden zu wechseln, die Distanz zu verändern und auch, sofern der Proband den Task nicht erfüllen kann, eine Aufgabe zu überspringen. Letzteres würde als Fehlversuch gewertet werden, war während der Evaluation aber nicht notwendig. Weiterhin lässt sich der *Demomodus* aktivieren oder deaktivieren, wodurch die Selektionszeiten und -fehler nicht gespeichert werden (sinnvoll für Testselektionen im Rahmen der Einweisung). Die Zielobjekte lassen sich im Unreal Editor beliebig verändern, ersetzen oder erweitern, um das Experiment noch anpassen zu können. Auch lassen sich die Objekte austauschen, vervielfachen oder entfernen. Zusätzlich können beliebige Werte für den ID hinterlegt werden, der für jedes Objekt einzeln definiert ist.

Die Evaluation wurde ebenfalls wie geplant abgeschlossen.
Hierzu wurde ein Terminplanungssystem genutzt, um den Probanden eine Teilnahme möglichst ohne Wartezeit zu ermöglichen. Durch Plakate, die auf dem Gelände der Universität Bremen verteilt wurden, sowie Ankündigungen in Vorlesungen und über Mailverteiler wurden Interessierte angesprochen, die sich zum größten Teil über dieses System anmeldeten. Dennoch gab es auch einzelne Teilnehmer ohne Termin, für die aber Pufferzeiten vorgesehen waren. Effektiv hatte kein Teilnehmer mehr als 8 Minuten Wartezeit. Während der Evaluation traten keine Unregelmäßigkeiten auf, jedoch stellten sich im Nachhinein einige Datensätze als unvollständig heraus, da eine fehlerhafte Speicherung vorlag oder es sich um Extremwerte handelte. Diese wurden daher von der Auswertung ausgeschlossen. Mithilfe der Software *IBM SPSS Statistics* und

der Online-Plattform *StatistikGuru* [19] konnten die verbliebenen Datensätze aufbereitet und ausgewertet werden.

Das Ergebnis der Arbeit macht deutlich, dass ein direkter Vergleich von unmittelbaren Selektionsmethoden mit iterativen Verfahren grundsätzlich schwierig ist.
Dies liegt insbesondere daran, dass für erstere häufig keine zusätzlichen Kenntnisse nötig sind, bei letzteren hingegen die Bedienung erst erlernt werden muss.

Auch wenn die Probanden in der Evaluation alle Methoden vorab testen durften und keine Fragen dazu hatten, ihnen die Bedienung somit hätte bekannt sein müssen, empfanden sie die drei Metaphern *lenSelect*, *Raycast* und *VOTE* gegenüber *PRECIOUS* als deutlich intuitiver, was sich auch deutlich in der Auswertung wiederspiegelte.

Natürlich ist das Ergebnis einer solchen Evaluation abhängig von vielen Faktoren, insbesondere der Stichprobe. In diesem Fall handelte es sich um Studenten der Studiengänge Digitale Medien, Systems Engineering, Informatik, BWL, Medizin, Technomathematik und Wirtschaftsinformatik sowie Auszubildende der Universität. Die Aufzählung erfolgt nach der Anzahl der Teilnehmer in absteigender Reihenfolge. Diese hatten im Durchschnitt wenig Erfahrung mit Virtual Reality (vgl. Kapitel 5.3.1 auf Seite 38), was eine gute Voraussetzung für die Bewertung neuer Techniken ist, insbesondere, wenn es um deren intuitive Bedienung geht.

Literaturverzeichnis

[1] Adrian3dartist. Street cones. http://www.sharecg.com/v/54478/browse/5/3D-Model/Street-cones Abruf am: 09.01.2018.

[2] Animated Heaven. Oil Barrel PBR. http://www.sharecg.com/v/87275/browse/5/3D-Model/Oil-Barrel-PBR Abruf am: 09.01.2018.

[3] Bacim, F., Kopper, R., and Bowman, D. A. Design and evaluation of 3d selection techniques based on progressive refinement. *Int. J. Hum.-Comput. Stud. 71*, 7-8 (July 2013), 785–802.

[4] Bortz, J., and Schuster, C. Statistik für Human- und Sozialwissenschaftler. 7., vollständig überarbeitete und erweiterte Auflage. *Berlin [ua]: Springer* (2010).

[5] Bowman, D. A., and Hodges, L. F. An evaluation of techniques for grabbing and manipulating remote objects in immersive virtual environments. In *Proceedings of the 1997 Symposium on Interactive 3D Graphics* (New York, NY, USA, 1997), I3D '97, ACM, pp. 35–ff.

[6] Bowman, D. A., Kruijff, E., LaViola, J. J., and Poupyrev, I. *3D User Interfaces: Theory and Practice*. Addison Wesley Longman Publishing Co., Inc., Redwood City, CA, USA, 2004.

[7] Cashion, J., Wingrave, C., and Jr., J. J. L. Dense and dynamic 3d selection for game-based virtual environments. *IEEE Transactions on Visualization and Computer Graphics 18*, 4 (April 2012), 634–642.

[8] Charness, G., Gneezy, U., and Kuhn, M. A. Experimental methods: Between-subject and within-subject design. *Journal of Economic Behavior & Organization 81*, 1 (2012), 1–8.

[9] Crumpler. City Trash and Waste Set. https://www.unrealengine.com/zh-CN/marketplace/city-trash-and-waste-set Abruf am: 09.01.2018.

[10] De Haan, G., Koutek, M., and Post, F. H. Intenselect: Using dynamic object rating for assisting 3d object selection. In *IPT/EGVE* (2005), Citeseer, pp. 201–209.

[11] Debarba, H. G., Grandi, J. G., Maciel, A., Nedel, L., and Boulic, R. *Disambiguation Canvas: A Precise Selection Technique for Virtual Environments*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013, pp. 388–405.

[12] Dorland. Watering can. http://www.sharecg.com/v/33034/browse/5/3D-Model/Watering-can Abruf am: 09.01.2018.

[13] Drewes. A Lecture on Fitt's Law. https://www.cip.ifi.lmu.de/~drewes/science/fitts/A%20Lecture%20onFitts%20Law.pdf Abruf am: 09.01.2018.

[14] Epic Games Inc. Blueprint Overview. https://docs.unrealengine.com/latest/INT/Engine/Blueprints/Overview Abruf am: 15.02.2018.

[15] Epic Games Inc. Pawn. https://docs.unrealengine.com/latest/INT/Gameplay/Framework/Pawn Abruf am: 28.02.2018.

[16] Figueroa, P., Kitamura, Y., Kuntz, S., Vanacken, L., Maesen, S., De Weyer, T., Notelaers, S., Octavia, J. R., Beznosyk, A., Coninx, K., Bacim, F., Kopper, R., Leal, A., Ni, T., and Bowman, D. A. 3dui 2010 contest grand prize winners. *IEEE Computer Graphics and Applications 30*, 6 (2010), 86 – 96, c3.

[17] Girden, E. Sage University papers. Quantitative applications in the social sciences, Vol. 84. ANOVA: Repeated measures, 1992.

[18] Grossman, T., and Balakrishnan, R. The design and evaluation of selection techniques for 3d volumetric displays. In *Proceedings of the 19th annual ACM symposium on User interface software and technology* (2006), ACM, pp. 3–12.

[19] Hemmerich, W. StatistikGuru. https://statistikguru.de/spss Abruf am: 17.01.2018.

[20] Hurtienne, J., and Naumann, A. Quesi—a questionnaire for measuring the subjective consequences of intuitive use. *Interdisciplinary College 536* (2010).

[21] Liang, J., and Green, M. JDCAD: A highly interactive 3d modeling system. *Computers & Graphics 18*, 4 (1994), 499–506.

[22] Mendes, D., Medeiros, D., Cordeiro, E., Sousa, M., Ferreira, A., and Jorge, J. Precious! out-of-reach selection using iterative refinement in vr. In *2017 IEEE Symposium on 3D User Interfaces (3DUI)* (March 2017), pp. 237–238.

[23] Mine, M. R. *Virtual Environment Interaction Techniques.* Technical Report. University of North Carolina at Chapel Hill, Chapel Hill, NC, USA, March 1995.

[24] Moore, A. G. *VOTE: Vote-oriented technique enhancements.* Master's Thesis. University of Texas at Dallas, USA, 2016.

[25] Periverzov, F., and Ilieş, H. Ids: the intent driven selection method for natural user interfaces. In *3D User Interfaces (3DUI), 2015 IEEE Symposium on* (2015), IEEE, pp. 121–128.

[26] Poupyrev, I., Billinghurst, M., Weghorst, S., and Ichikawa, T. The go-go interaction technique: Non-linear mapping for direct manipulation in vr. In *Proceedings of the 9th Annual ACM Symposium on User Interface Software and Technology* (New York, NY, USA, 1996), UIST '96, ACM, pp. 79–80.

[27] Schumacher, J. Analysis of variance (anova) in r. http://users.minet.uni-jena.de/~jschum/biostat/ANOVA.pdf Abruf am: 16.02.2018.

[28] Steed, A., and Parker, C. 3d selection strategies for head tracked and non-head tracked operation of spatially immersive displays. In *8th International Immersive Projection Technology Workshop* (2004), pp. 13–14.

[29] Unity Technologies. Manual: Input for OpenVR controllers. https://docs.unity3d.com/560/Documentation/Manual/OpenVRControllers.html Abruf am: 16.02.2018.

[30] Zachmann, G. Distortion selection on the gpu for interactive 3d environments. Unpublished note.

Abbildungsverzeichnis

Tabellenverzeichnis

Glossar

3DUI Jährlich stattfindende Konferenz des IEEE (Institute of Electrical and Electronics Engineering) mit den Themen Virtual Reality und 3D Benutzeroberflächen.

cluttered environment Eine überladene Umgebung, beispielsweise ein Supermarktregal, bei welcher viele Objekte dicht aneinander (und/oder voreinander) stehen.

Control-display ratio Das Verhältnis zwischen der natürlichen Handbewegung (Control) und der visualisierten Bewegung des Controllers/Coursors im virtuellen Raum (Display).

Disambiguierung auch: Begriffserklärung; In der Sprachwissenschaft zur Auflösung von Mehrdeutigkeiten verwendet, in diesem Fall beispielsweise bei nicht-eindeutiger Selektion mehrere Objekte.

Head Mounted Display Darstellungsgerät, welches der Nutzer in Form einer Brille oder als Teil eines Helmes trägt. Das Gerät verfügt über ein kleines Display vor einem (monocular HMD) oder beiden (binocular HMD) Augen.

IEEE Institute of Electrical and Electronics Engineering, ein weltweiter Berufsverband von Ingenieuren aus den Bereichen Elektrotechnik und Informationstechnik.

lateinisches Quadrat Quadratisches Schema mit n Reihen und n Spalten, wobei jedes Feld mit einem von n verschiedenen Symbolen belegt ist, sodass jedes Symbol in jeder Zeile und jeder Spalte jeweils genau einmal auftritt. Durch die Nutzung wird hier verhindert, dass mehrere Probanden die gleichen Methoden in der selben Reihenfolge nutzen.

Pawn Die physische Repräsentation eines Spielers in der virtuellen Welt. Er enthält die programmierte Logik, also die Definition des Aussehens, der Interaktion und Kollision mit der Welt [15].

Raycast Selektionstechnik, bei der ein virtueller Strahl vom Controller ausgehend berechnet wird. Sein Kollisionspunkt wird für die Auswahl des Selektionsobjektes verwendet.

Unreal Engine 4 Spieleentwicklungsumgebung der Firma Epic Games. Diese Plattform wird hier als Entwicklungsumgebung genutzt.

virtual environment Computergenerierte, dreidimensionale Umgebung, in welcher sich der Nutzer "befindet".

Virtual Reality Computergenerierte Simulation eines dreidimensionalen Bildes oder einer Umgebung, mit welcher der Nutzer mithilfe spezieller elektronischer Ausrüstung interagieren kann.

within-subject-design Jeder Proband absolviert nacheinander alle experimentellen Bedingungen oder Möglichkeiten.

Anhang

Anhang A Bilder

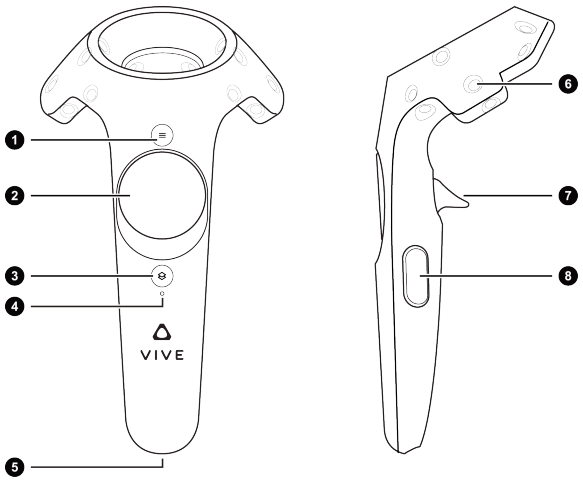


Abbildung 1.: HTC Vive Motion Controller [29]

Abbildung 2.: Ansicht der Supermarktszene vom 3DUI 2010[16]

Abbildung 3.: Einblick in die Durchführung der Evaluation

Abbildung 4.: Multiple Selektion bei PRECIOUS

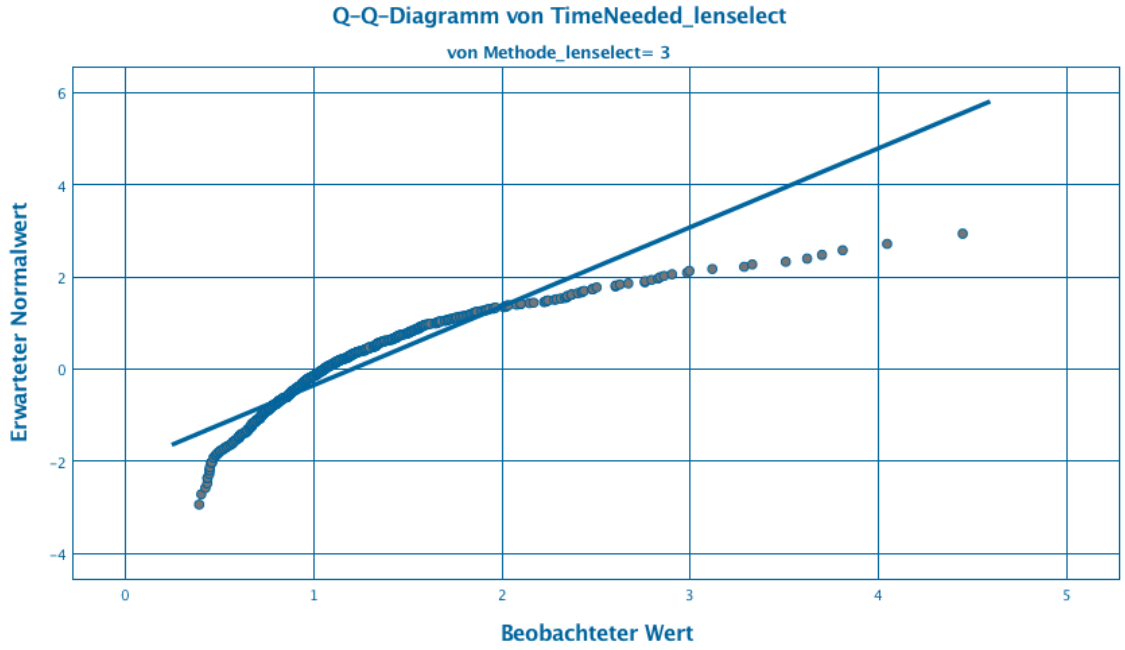


Abbildung 5.: Verteilung der `task completion time` bei *lenSelect*

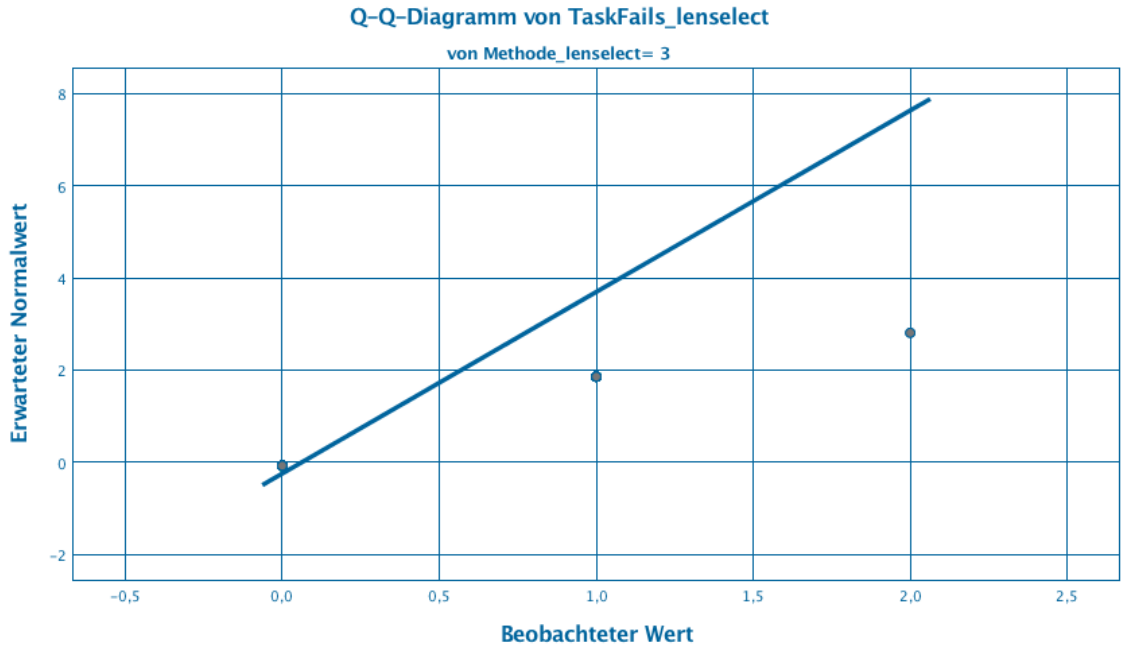


Abbildung 6.: Verteilung der `Fehlerzahl` bei *lenSelect*

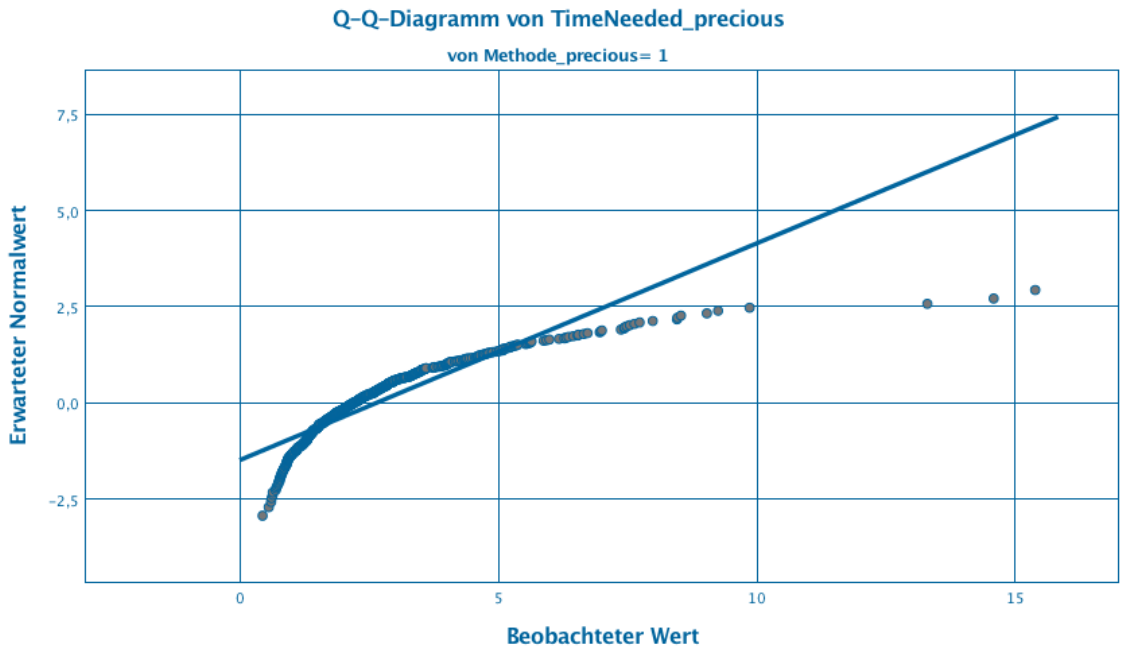


Abbildung 7.: Verteilung der `task completion time` bei *PRECIOUS*

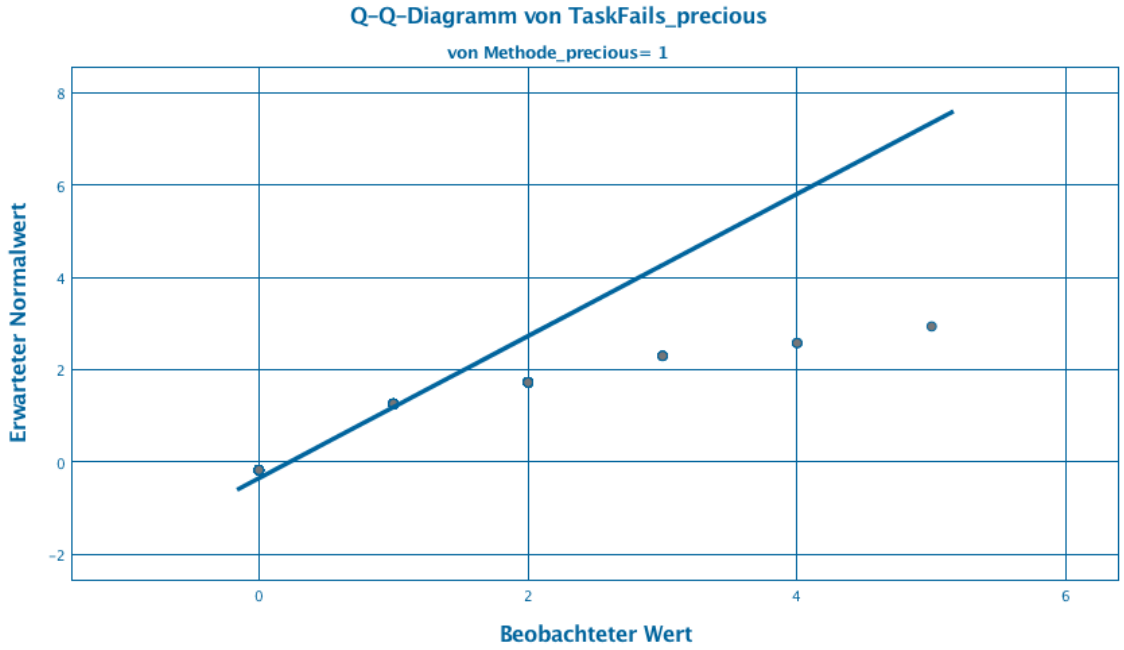


Abbildung 8.: Verteilung der `Fehlerzahl` bei *PRECIOUS*

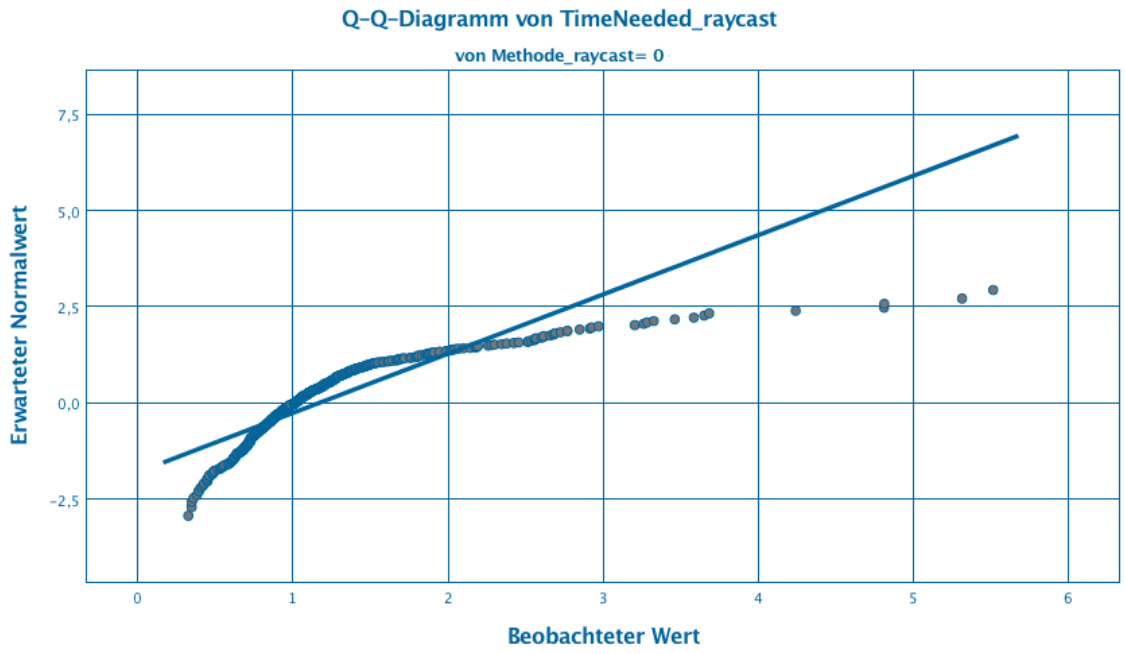


Abbildung 9.: Verteilung der `task completion time` bei *Raycast*

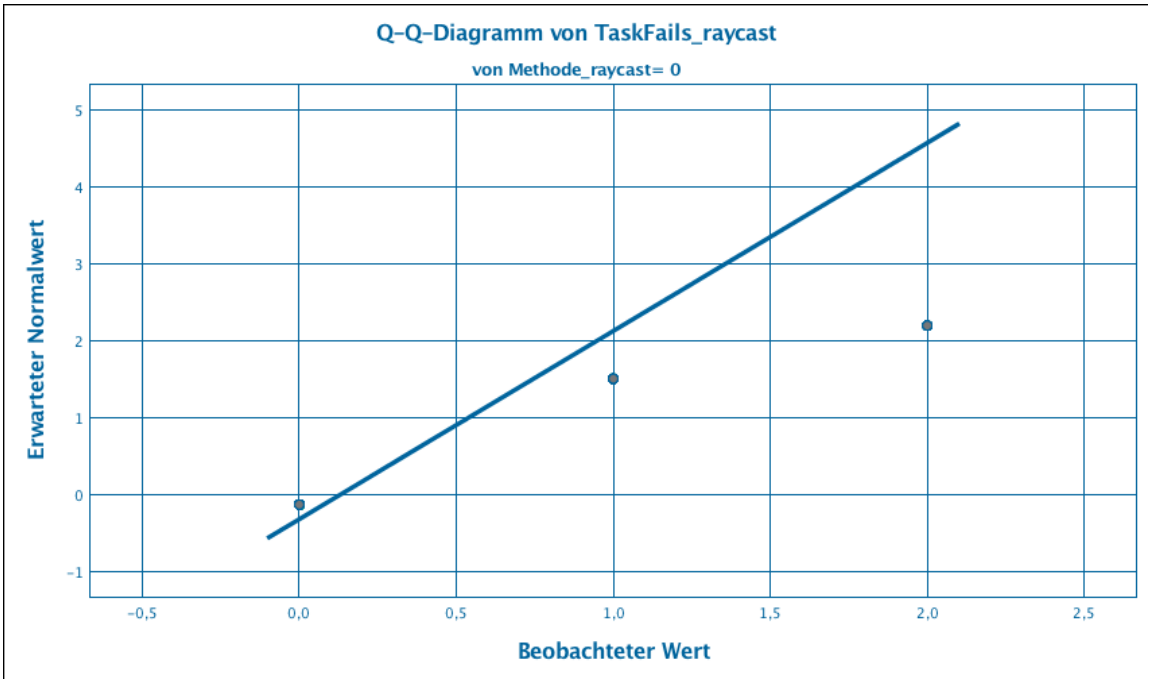


Abbildung 10.: Verteilung der `Fehlerzahl` bei *Raycast*

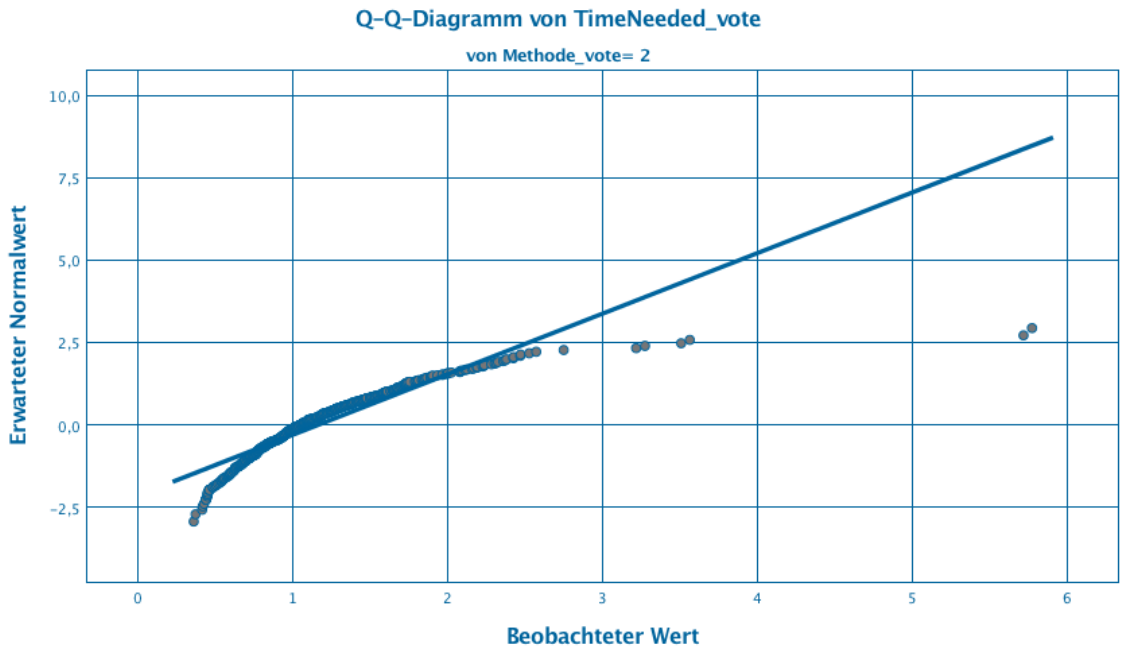


Abbildung 11.: Verteilung der `task completion time` bei *VOTE*

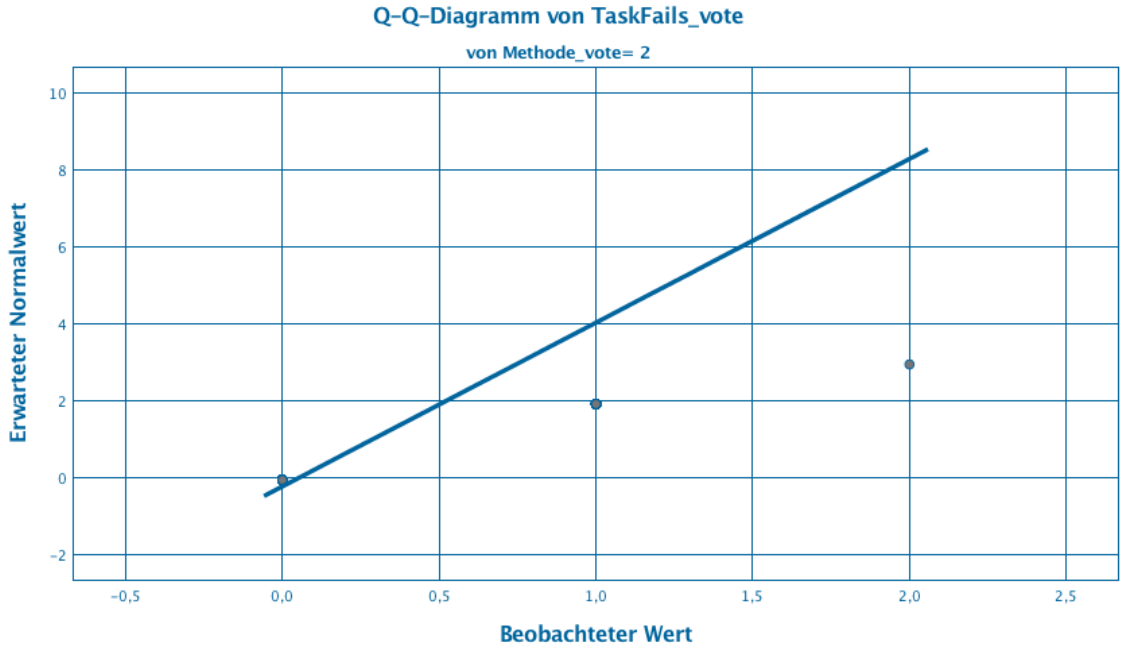
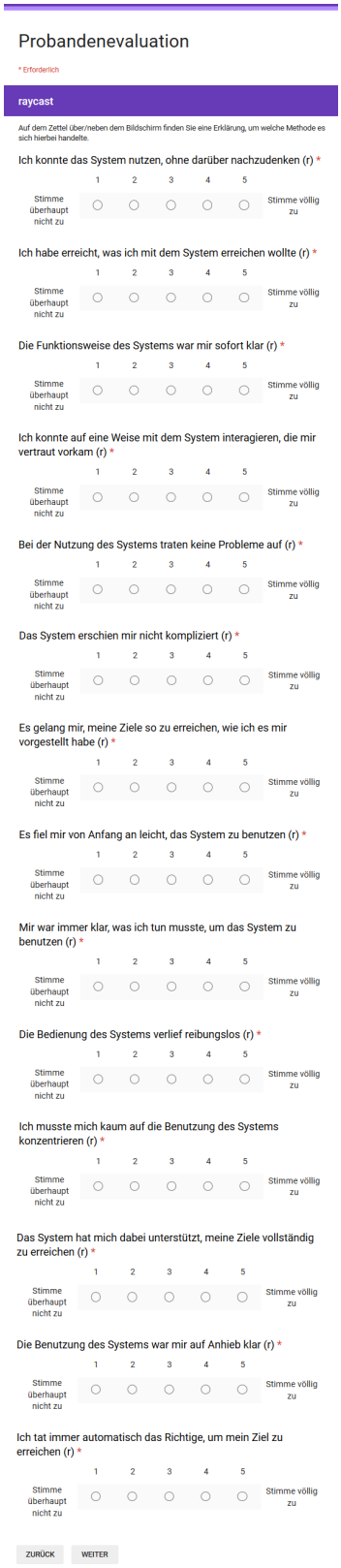


Abbildung 12.: Verteilung der `Fehlerzahl` bei *VOTE*



Probandenevaluation

* Erforderlich

raycast

Auf dem Zettel über/neben dem Bildschirm finden Sie eine Erklärung, um welche Methode es sich hierbei handelte.

Ich konnte das System nutzen, ohne darüber nachzudenken (r) *

	1	2	3	4	5	
Stimme überhaupt nicht zu	○	○	○	○	○	Stimme völlig zu

Ich habe erreicht, was ich mit dem System erreichen wollte (r) *

	1	2	3	4	5	
Stimme überhaupt nicht zu	○	○	○	○	○	Stimme völlig zu

Die Funktionsweise des Systems war mir sofort klar (r) *

	1	2	3	4	5	
Stimme überhaupt nicht zu	○	○	○	○	○	Stimme völlig zu

Ich konnte auf eine Weise mit dem System interagieren, die mir vertraut vorkam (r) *

	1	2	3	4	5	
Stimme überhaupt nicht zu	○	○	○	○	○	Stimme völlig zu

Bei der Nutzung des Systems traten keine Probleme auf (r) *

	1	2	3	4	5	
Stimme überhaupt nicht zu	○	○	○	○	○	Stimme völlig zu

Das System erschien mir nicht kompliziert (r) *

	1	2	3	4	5	
Stimme überhaupt nicht zu	○	○	○	○	○	Stimme völlig zu

Es gelang mir, meine Ziele so zu erreichen, wie ich es mir vorgestellt habe (r) *

	1	2	3	4	5	
Stimme überhaupt nicht zu	○	○	○	○	○	Stimme völlig zu

Es fiel mir von Anfang an leicht, das System zu benutzen (r) *

	1	2	3	4	5	
Stimme überhaupt nicht zu	○	○	○	○	○	Stimme völlig zu

Mir war immer klar, was ich tun musste, um das System zu benutzen (r) *

	1	2	3	4	5	
Stimme überhaupt nicht zu	○	○	○	○	○	Stimme völlig zu

Die Bedienung des Systems verlief reibungslos (r) *

	1	2	3	4	5	
Stimme überhaupt nicht zu	○	○	○	○	○	Stimme völlig zu

Ich musste mich kaum auf die Benutzung des Systems konzentrieren (r) *

	1	2	3	4	5	
Stimme überhaupt nicht zu	○	○	○	○	○	Stimme völlig zu

Das System hat mich dabei unterstützt, meine Ziele vollständig zu erreichen (r) *

	1	2	3	4	5	
Stimme überhaupt nicht zu	○	○	○	○	○	Stimme völlig zu

Die Benutzung des Systems war mir auf Anhieb klar (r) *

	1	2	3	4	5	
Stimme überhaupt nicht zu	○	○	○	○	○	Stimme völlig zu

Ich tat immer automatisch das Richtige, um mein Ziel zu erreichen (r) *

	1	2	3	4	5	
Stimme überhaupt nicht zu	○	○	○	○	○	Stimme völlig zu

ZURÜCK WEITER

Abbildung 13.: QUESI Fragebogen, beispielhaft für eine Methode

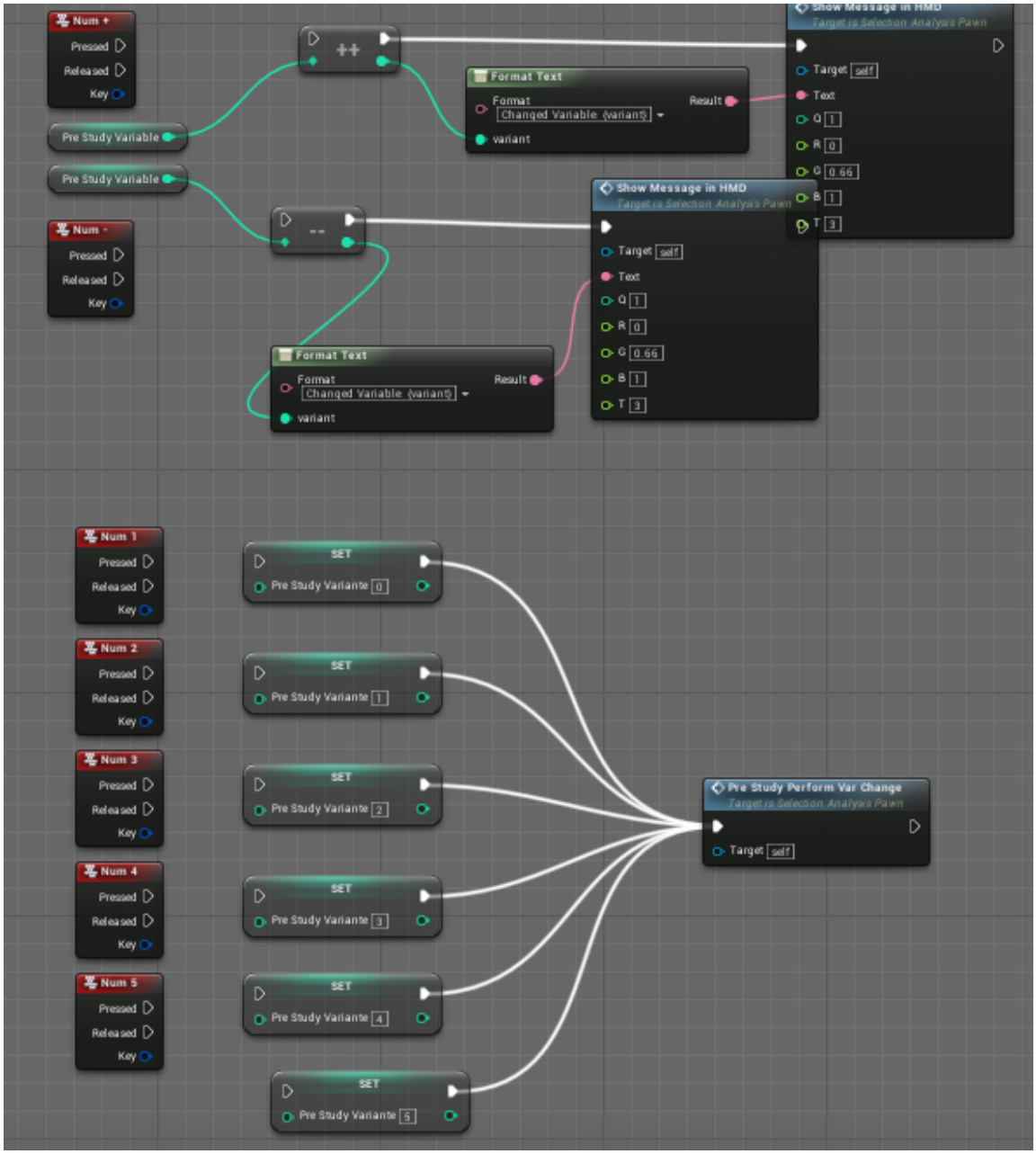


Abbildung 14.: Blueprint `Pre-Study Variants`

Abbildung 15.: Aushang zur Probandensuche

Anhang B Code

```
#include "FileAgent.h"
#include <string>
#include "Engine.h"

// Sets default values
AFileAgent::AFileAgent()
{
  // Set this actor to call Tick() every frame.  You can turn this off to improve performance
     if you don't need it.
  PrimaryActorTick.bCanEverTick = true;

}

// Called when the game starts or when spawned
void AFileAgent::BeginPlay()
{
  Super::BeginPlay();

}

// Called every frame
void AFileAgent::Tick(float DeltaTime)
{
  Super::Tick(DeltaTime);

}

bool AFileAgent::WriteToSaveFile(FString text, int id, FString FileNameParam) {

  FString SaveDirectory = FString("A:/Uni/Wise2017/Bachelorarbeit/Implementierung/
     SimpleSelectionAnalysis/SSA/Saved/SaveGames");
  FString FileName = FString("evaluationdaten-");
  FileName.AppendInt(id);
  FileName.Append("-");
  FileName.Append(FileNameParam);
  FileName.Append(".csv");
  FString TextToSave;

  bool AllowOverwriting = false;
  bool ret = false;
  IPlatformFile& PlatformFile = FPlatformFileManager::Get().GetPlatformFile();

  // CreateDirectoryTree returns true if the destination
  // directory existed prior to call or has been created
  // during the call.
  if (PlatformFile.CreateDirectoryTree(*SaveDirectory))
  {
    // Get absolute file path
    FString AbsoluteFilePath = SaveDirectory + "/" + FileName;
    FFileHelper::LoadFileToString(TextToSave, *AbsoluteFilePath);

    TextToSave.Append("\n");
```

```
51
53      for (char i : text) {
          if (i == ',') {
55
          }
57        else {
            if (i == '.') {
59            TextToSave.AppendChar(',');
            }
61          else {
              if (i == '/') {
63              TextToSave.Append("\n");
              }
65            else {
                TextToSave.AppendChar(i);
67            };
            };
69        };
      }
71
    ret=FFileHelper::SaveStringToFile(TextToSave, *AbsoluteFilePath);
73
    }
75  return ret;
}
```

Listing 7.1: Speicherfunktion

Anhang C Tabellen

Methode	ID	M	SD
lenSelect	1	0,93	0,37
	3	1,20	0,43
	5	1,48	0,73
TOTAL		1,20	0,51
PRECIOUS	1	2,83	1,53
	3	2,68	1,18
	5	2,39	1,99
TOTAL		2,63	1,56
Raycast	1	0,84	0,29
	3	1,12	0,38
	5	1,53	0,77
TOTAL		1,16	0,48
VOTE	1	0,89	0,33
	3	1,21	0,42
	5	1,38	0,60
TOTAL		1,16	0,45

Tabelle 1.: Deskriptive Statistik für die *task completion time*

Methode	ID	M	SD
lenSelect	1	0,02	0,06
	3	0,05	0,20
	5	0,13	0,34
TOTAL		0,07	0,19
PRECIOUS	1	0,04	0,16
	3	0,18	0,48
	5	0,61	0,82
TOTAL		0,23	0,49
Raycast	1	0,01	0,04
	3	0,06	0,22
	5	0,33	0,53
TOTAL		0,13	0,26
VOTE	1	0,02	0,08
	3	0,05	0,18
	5	0,11	0,30
TOTAL		0,06	0,18

Tabelle 2.: Deskriptive Statistik für die Fehlerzahl

Objekt	**ID**	***lenSelect***		***PRECIOUS***		***Raycast***		***VOTE***	
		M	SD	M	SD	M	SD	M	SD
Bleach	3	1,232	0,469	2,720	1,255	1,102	0,298	1,254	0,389
Bottle2-1	3	1,288	0,549	2,734	1,023	1,240	0,492	1,275	0,515
Bottle3	3	1,160	0,378	3,544	1,581	1,022	0,298	1,164	0,380
Broom	5	1,457	0,730	2,449	1,834	1,167	0,461	1,180	0,451
Cigs6	5	1,558	0,776	2,877	3,295	1,890	1,083	1,380	0,553
Cup5	3	1,121	0,339	1,757	0,863	1,133	0,418	1,160	0,415
Detergent	5	1,591	0,728	2,646	2,020	1,863	1,005	1,781	1,004
Magazine	5	1,313	0,670	1,602	0,797	1,216	0,532	1,177	0,407
TrashBag7	1	1,018	0,402	3,347	1,853	0,955	0,331	1,040	0,361
TrashBin6	1	0,979	0,370	3,399	1,501	0,854	0,291	0,881	0,321
TrashBin7	1	0,846	0,382	2,337	1,290	0,779	0,293	0,833	0,323
TrashCan	1	0,866	0,323	2,244	1,479	0,776	0,256	0,825	0,308

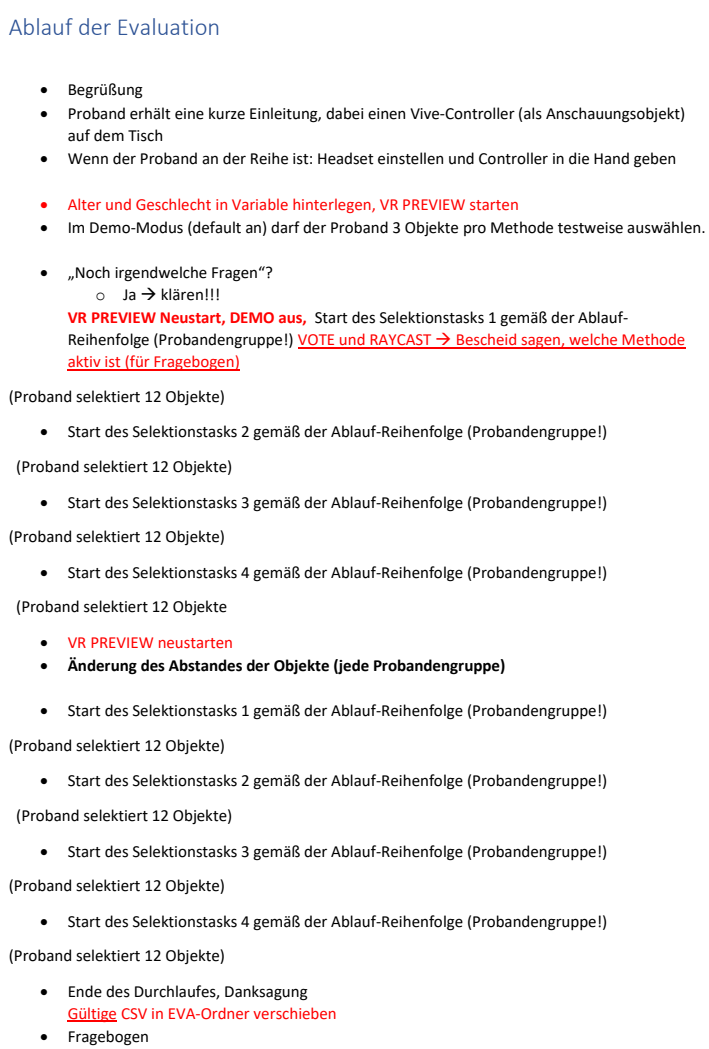
Tabelle 3.: Deskriptive Statistik für die *task completion time* in Abhängigkeit vom Objekt

Objekt	**ID**	***lenSelect***		***PRECIOUS***		***Raycast***		***VOTE***	
		M	SD	M	SD	M	SD	M	SD
Bleach	3	0,02	0,14	0,06	0,32	0,02	0,14	0,06	0,24
Bottle2-1	3	0,02	0,15	0,22	0,50	0,12	0,33	0,06	0,25
Bottle3	3	0,10	0,30	0,12	0,38	0,02	0,14	0,00	0,00
Broom	5	0,12	0,32	0,27	0,60	0,06	0,24	0,02	0,14
Cigs6	5	0,17	0,43	0,35	0,82	0,53	0,73	0,08	0,28
Cup5	3	0,04	0,20	0,31	0,71	0,06	0,25	0,06	0,24
Detergent	5	0,19	0,45	1,10	1,34	0,58	0,79	0,24	0,43
Magazine	5	0,02	0,14	0,18	0,52	0,16	0,37	0,08	0,33
TrashBag7	1	0,00	0,00	0,09	0,29	0,00	0,00	0,02	0,14
TrashBin6	1	0,06	0,25	0,00	0,00	0,00	0,00	0,00	0,00
TrashBin7	1	0,00	0,00	0,04	0,19	0,02	0,14	0,04	0,19
TrashCan	1	0,00	0,00	0,02	0,14	0,00	0,00	0,00	0,00

Tabelle 4.: Deskriptive Statistik für die Fehlerzahl in Abhängigkeit vom Objekt

Anhang D Dokumente

D.1 Ablauf



Ablauf der Evaluation

- Begrüßung
- Proband erhält eine kurze Einleitung, dabei einen Vive-Controller (als Anschauungsobjekt) auf dem Tisch
- Wenn der Proband an der Reihe ist: Headset einstellen und Controller in die Hand geben

- Alter und Geschlecht in Variable hinterlegen, VR PREVIEW starten
- Im Demo-Modus (default an) darf der Proband 3 Objekte pro Methode testweise auswählen.

- „Noch irgendwelche Fragen“?
  - Ja → klären!!!

  VR PREVIEW Neustart, DEMO aus, Start des Selektionstasks 1 gemäß der Ablauf-Reihenfolge (Probandengruppe!) VOTE und RAYCAST → Bescheid sagen, welche Methode aktiv ist (für Fragebogen)

(Proband selektiert 12 Objekte)

- Start des Selektionstasks 2 gemäß der Ablauf-Reihenfolge (Probandengruppe!)

(Proband selektiert 12 Objekte)

- Start des Selektionstasks 3 gemäß der Ablauf-Reihenfolge (Probandengruppe!)

(Proband selektiert 12 Objekte)

- Start des Selektionstasks 4 gemäß der Ablauf-Reihenfolge (Probandengruppe!)

(Proband selektiert 12 Objekte

- VR PREVIEW neustarten
- **Änderung des Abstandes der Objekte (jede Probandengruppe)**

- Start des Selektionstasks 1 gemäß der Ablauf-Reihenfolge (Probandengruppe!)

(Proband selektiert 12 Objekte)

- Start des Selektionstasks 2 gemäß der Ablauf-Reihenfolge (Probandengruppe!)

(Proband selektiert 12 Objekte)

- Start des Selektionstasks 3 gemäß der Ablauf-Reihenfolge (Probandengruppe!)

(Proband selektiert 12 Objekte)

- Start des Selektionstasks 4 gemäß der Ablauf-Reihenfolge (Probandengruppe!)

(Proband selektiert 12 Objekte)

- Ende des Durchlaufes, Danksagung
  Gültige CSV in EVA-Ordner verschieben
- Fragebogen

Abbildung 16.: Ablauf der Evaluation

D.2 Theoretische Einweisung

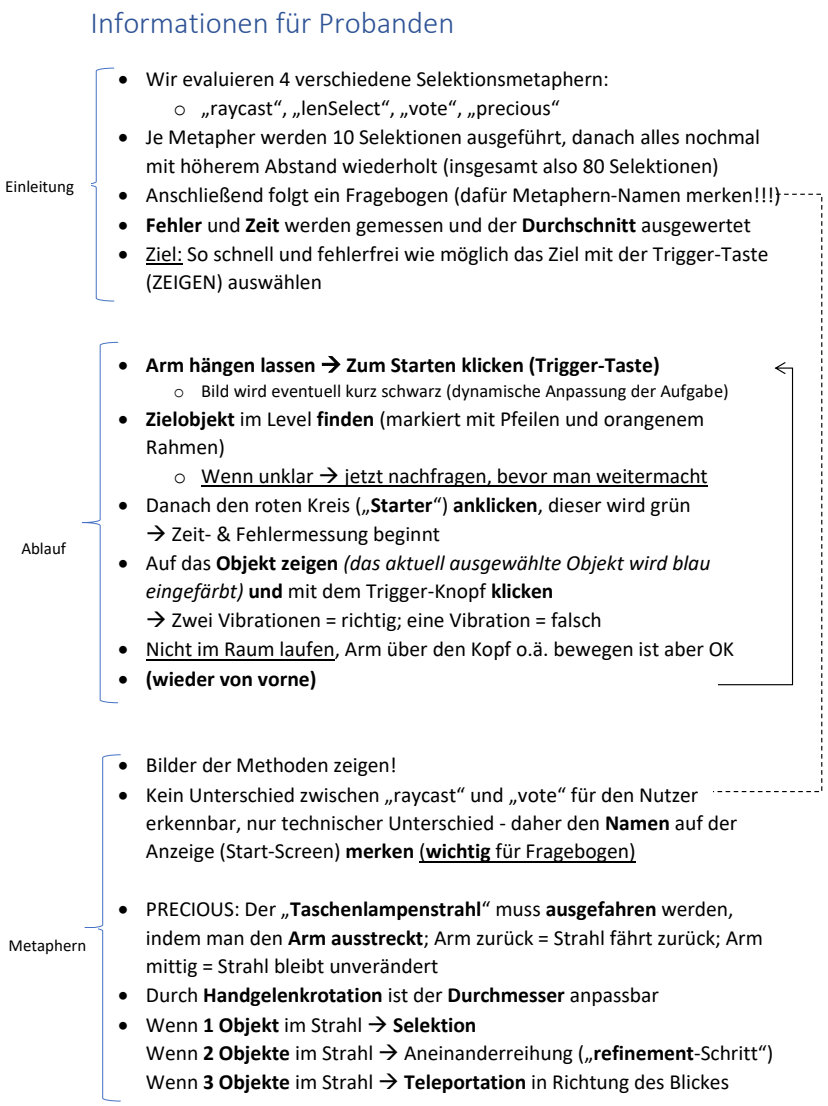


Abbildung 17.: Inhalte der theoretischen Einweisung

D.3 Protokoll

Evaluationsprotokoll

Proband Nr.	P-Gruppe	Tag	Uhrzeit	Zieltyp	Methode 1	Methode 2	Methode 3	Methode 4
1	1	30	9.15	SC	Raycast	PRECIOUS	VOTE	lenSelect
2	2		9.40	SU	PRECIOUS	Raycast	lenSelect	VOTE
3	3		10.00	BC	VOTE	lenSelect	Raycast	PRECIOUS
4	4		10.30	BU	lenSelect	VOTE	PRECIOUS	Raycast
5	1		10.42	SC	Raycast	PRECIOUS	VOTE	LenSelect
6	2		11.00	SU	PRECIOUS	Raycast	lenSelect	VOTE
7	3		11.32	BC	VOTE	lenSelect	Raycast	PRECIOUS
8	4		11.46	BU	lenSelect	VOTE	PRECIOUS	Raycast
9	1		12.35	SC	Raycast	PRECIOUS	VOTE	LenSelect
10	2		13.15	SU	PRECIOUS	Raycast	lenSelect	VOTE
11	3		14.00	BC	VOTE	lenSelect	Raycast	PRECIOUS
12	4		14.17	BU	lenSelect	VOTE	PRECIOUS	Raycast
13	1		14.33	SC	Raycast	PRECIOUS	VOTE	LenSelect
14	2	31	8.25	SU	PRECIOUS	Raycast	lenSelect	VOTE
15	3		9.13	BC	VOTE	lenSelect	Raycast	PRECIOUS
16	4		10.04	BU	lenSelect	VOTE	PRECIOUS	Raycast
17	1		10.15	SC	Raycast	PRECIOUS	VOTE	LenSelect
18	2		10.35	SU	PRECIOUS	Raycast	lenSelect	VOTE
19	3		12.30	BC	VOTE	lenSelect	Raycast	PRECIOUS
20	4		12.50	BU	lenSelect	VOTE	PRECIOUS	Raycast
21	1		13.20	SC	Raycast	PRECIOUS	VOTE	LenSelect
22	2		13.43	SU	PRECIOUS	Raycast	lenSelect	VOTE
23	3		13.55	BC	VOTE	lenSelect	Raycast	PRECIOUS
24	4		14.05	BU	lenSelect	VOTE	PRECIOUS	Raycast
25	1		14.20	SC	Raycast	PRECIOUS	VOTE	LenSelect
26	2		14.40	SU	PRECIOUS	Raycast	lenSelect	VOTE
27	3	30	13.32	BC	VOTE	lenSelect	Raycast	PRECIOUS
28	4		13.50	BU	lenSelect	VOTE	PRECIOUS	Raycast
29	1			SC	Raycast	PRECIOUS	VOTE	LenSelect
30	2			SU	PRECIOUS	Raycast	lenSelect	VOTE

Abbildung 18.: Evaluationsprotokoll

D.4 Tastendefinitionen

TASTENKOMBINATIONEN F. EVALUATION

Methodenwechsel

F	**FishEye (3) /lenSelect**
P	**PRECIOUS (1)**
R	**Raycast (0)**
V	**Vote (2)**

Demo-Modus, Ziele

D	**Demo-Modus Toggle**
M	Negieren der Entfernung, 500 → -500 → 500
N	Entfernung erhöhen
SPACE	Start / Stop Toggle
Shift-Links	Wechsel des Ziels (Überspringen) (zählt als Fehler)

Abbildung 19.: Tastendefinitionen für die Evaluation

Anhang E Datenträger

Auf dem Datenträger, der dieser Arbeit beiliegt, befinden sich folgende Daten:

- Die Implementierung der Software ("SSA")
  - in Form einer ausführbaren *EXE*-Datei
  - in Form einer *UPROJECT*-Datei für Unreal Engine 4.18
- Alle Rohdaten, die während der Evaluation gesammelt wurden
  - in Form von *CSV*-Dateien
- Die statistische Auswertung der Daten
  - in Form von *PDF*-Dateien
  - in Form von *SAV*-Dateien für IBM SPSS Statistics
- Ein Demo-Film
  - in Form einer *MP4*-Datei