Verbesserung der User-Experience von SEE durch ein interaktives Hilfe-System

Bachelorarbeit

Thore Frenzel

Matrikelnummer: 4449639

6. Januar 2022

Fachbereich Mathematik / Informatik
Studiengang Wirtschaftsinformatik

1. Gutachter: Prof. Dr. Rainer Koschke
2. Gutachter: Susanne Putze

Abstract

In dieser Arbeit wurde untersucht, ob der Einsatz eines visuellen, akustischen Hilfesystems in Form vorgespielter Beispielinteraktionen innerhalb einer Anwendung wie SEE eine sinnvolle Alternative zum Einsatz einer herkömmlichen Benutzerdokumentation bieten kann. Bei SEE handelt es sich um eine Applikation zur Softwarevisualisierung. Zunächst wurde ein Hilfesystem entworfen und implementiert sowie eine Benutzerdokumentation geschrieben. In einer vergleichenden Benutzerstudie zwischen den zuvor angesprochenen Varianten wurde daraufhin ermittelt, dass ein Hilfesystem zu einer signifikant besseren Usability von SEE führt. Des Weiteren unterschied sich die Korrektheit der Ergebnisse der in der Studie durchgeführten Aufgaben bei beiden Optionen kaum. Aus weiteren Daten ließ sich jedoch ermitteln, dass ein Hilfesystem sinnvoller als Ergänzung zu einer Benutzerdokumentation dienen kann als als Ersatz, da von beiden Hilfen verschiedene Nutzergruppen angesprochen werden. Abschließend werden Verbesserungen und Entwicklungsmöglichkeiten zum entwickelten Hilfesystem präsentiert, die das Hilfesystem für die Zukunft optimieren sollen.

Erklärung

Ich versichere, die Bachelorarbeit ohne fremde Hilfe angefertigt zu haben. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel benutzt. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen sind, sind als solche kenntlich gemacht.

Bremen, den 6. Januar 2022

.....
(Thore Frenzel)

Gender-Hinweis

In dieser Arbeit wird aus Gründen der besseren Lesbarkeit das generische Maskulinum verwendet. Weibliche und anderweitige Geschlechteridentitäten werden dabei ausdrücklich mitgemeint, soweit es für die Aussage erforderlich ist.

Danksagung

Zunächst möchte ich mich bei meinem Betreuer Prof. Dr. Rainer Koschke bedanken, der bei Fragen und Problemen auch sehr kurzfristig erreichbar war und mir so die Arbeit sehr erleichterte.
Darüber hinaus gilt mein besonderer Dank insbesondere meinen Kommilitonen Falko Galperin und Moritz Blecker, welche mir auch bei fachspezifischen Fragen und Problemen tatkräftig zur Seite standen, sowie vielen weiteren Kommilitonen.
Ein weiterer Dank gilt auch allen Probanden, welche an meiner Benutzerstudie teilnahmen und so die Grundlage für das Ergebnis dieser Arbeit boten sowie allen Korrekturlesern, die mit umfangreichem Feedback dafür sorgten, die Qualität dieser Arbeit zu verbessern. Ohne all diese Unterstützung wäre mir ein Abschluss dieser Arbeit und somit meines Studiums nicht möglich gewesen.

INHALTSVERZEICHNIS

KAPITEL 1

Einleitung

1.1 Softwarevisualisierung mit SEE

Seit Jahren steigt die Anzahl an Softwareentwicklern auf der Welt zunehmend an. [Gee21] So gab es im Jahr 2014 ca. 18,5 Millionen Softwareentwickler auf der Welt, 2019 waren es bereits 26,5 Millionen - Prognosen behaupten, dass es im Jahr 2030 ca. 45 Millionen Softwareentwickler geben wird. Hieran ist die Relevanz von Software und ihrer Entwicklung in der heutigen Welt und auch in der Zukunft eindeutig zu erkennen. Ebenfalls ersichtlich ist es, dass Software immer komplexer und umfangreicher wird. Dieses Problem thematisiert auch Arthur (1993) in seinem Artikel "On the Evolution of Complexity". Das stellt Softwareentwickler und den Softwareentwicklungsprozess vor neue Herausforderungen: Es müssen Wege und Lösungen gefunden werden, die Übersichtlichkeit von Software zu erhöhen, die Einarbeitung in Quellcode und Softwarearchitekturen zu vereinfachen und den kollaborativen Softwareentwicklungsprozess zu unterstützen. Einen möglichen Lösungsansatz für das zuvor angesprochene Problem und die erwähnten Konzepte versucht die Software **Software Engineering Experience**, kurz **SEE**, umzusetzen.

SEE ist eine Anwendung zur Softwarevisualisierung, die seit einigen Jahren von der AG Softwaretechnik der Universität Bremen entwickelt wird. SEE[1] bedient sich hierfür einer Stadt als Metapher.[Kos20] Es werden Klassen einer eingelesenen Software auf Häuser innerhalb einer Stadt, einer sogenannten Code-City abgebildet, welche wiederum in Stadtviertel, also Komponenten und Pakete unterteilt sind. Kommunikationswege innerhalb der Software werden über „Straßen“, also zwischen den Häusern verlaufenden Kanten, repräsentiert. Ein Beispiel für dieses Konzept der Softwarevisualisierung ist in Abbildung 1.1 zu erkennen.

Durch diese Darstellungsweise soll die eingelesene Software übersichtlicher präsentiert werden. SEE bietet verschiedene Möglichkeiten zur Interaktion, die Hauptfunktionen lassen sich jedoch in vier Hauptbereiche unterteilen[Ble+21]: Das Software-Debugging, die Evolution von Software, der Architekturvergleich und die Auswertung von Qualitätsmetriken. Die in SEE abgebildeten Städte, oder auch Code-Cities, können in verschiedensten Layouts gerendert werden. Einige Beispiele sind das TreeMap-Layout, Circle-Packing, Rectangle-Packing und Evo-Streets, welche alle unterschiedliche Schwerpunkte in ihrer Darstellungsweise haben. Ein

[1] Quellcode einsehbar unter: `https://github.com/uni-bremen-agst/SEE` (Zuvor sollte eine Rückfrage an Prof. Dr. Rainer Koschke wegen des Zugangs zum Repository erfolgen)

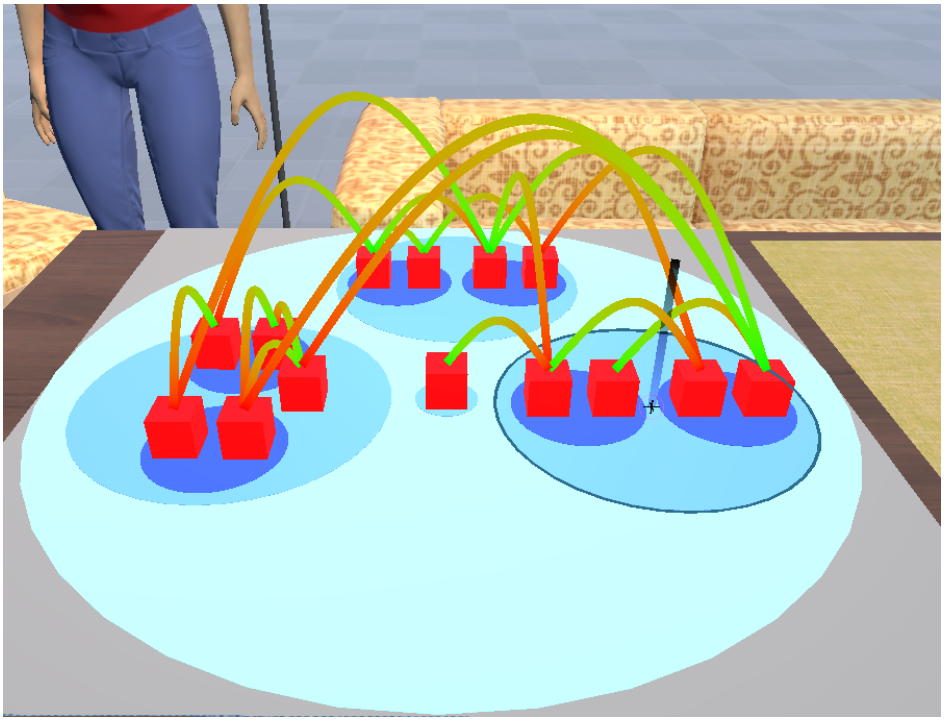


Abbildung 1.1: Ein konkretes Beispiel für eine Code-City

weiteres interessantes Feature von SEE ist die Plattformunabhängigkeit. SEE wurde sowohl für den Desktop, Virtual Reality, als auch für die Augmented Reality entwickelt und ist im Multi-User-Betrieb plattformübergreifend nutzbar[2]. Langfristig soll die eigentliche Softwareentwicklung in SEE möglich sein, Quellcode einzelner Häuser bzw. Klassen kann bereits angezeigt werden. Zur Erreichung dieser Ziele wird und wurde SEE auch in einigen studentischen Projekten sukzessive weiterentwickelt.

1.2 Problemstellung - Ziel der Arbeit

Durch die Anzahl an Möglichkeiten und Anwendungsfällen, welche in SEE ausgeführt werden können, entsteht für Neueinsteiger schnell ein Gefühl von Überforderung, da viele Anwendungsfälle aufgrund ihrer Komplexität nicht zwangsläufig intuitiv zu finden sind. Auch regelmäßige Nutzer könnten jedoch das Bedürfnis haben, sich bei einigen Interaktionen erneut über deren Funktionsweise und Details zu informieren. Für diesen Fall besitzen viele andere Applikationen und Softwarelösungen Benutzerhandbücher in unterschiedlichen Detailgraden, Sprachen und Ausfertigungen. Ein solches Benutzerhandbuch ist bei SEE bisher allerdings nicht vorhanden, es existiert lediglich ein Entwicklerwiki, welches dem Entwickler den Einstieg bei der Mitentwicklung an SEE erleichtern soll. Dieses wird jedoch nicht für den Endnutzer freigegeben. Um eine effektivere Einarbeitung in SEE sowohl für den Endnutzer, als auch für den Entwickler zu ermöglichen, müsste also langfristig entweder ein umfangreiches, detaillier-

[2]Augmented Reality kann z.B. über Hardware wie die HoloLens 2 realisiert werden und transferiert die Code-City mithilfe einer brillenähnlichen Apparatur in die reale Welt, wohingegen VR mithilfe von Hardware wie der HTC VIVE abgebildet wird und durch diese Hardware eine virtuelle Realität erzeugt

tes Benutzerhandbuch geschrieben werden, oder es müssten andere Möglichkeiten gefunden werden, Nutzern den Einstieg in SEE zu vereinfachen. Der Nachteil eines Benutzerhandbuchs ist dadurch gekennzeichnet, dass dessen Hauptaugenmerk auf einer textuellen Beschreibung der Interaktion beruht, ablaufende Prozesse können nur durch kleinschrittige Screenshots oder umfangreiche, schrittweise textuelle Beschreibungen dargestellt werden. Eine Frage, die sich daraufhin stellt, ist, ob es effektivere Methoden zur Einarbeitung von Benutzern in Software bzw. zur Benutzerunterstützung gibt als ein herkömmliches Benutzerhandbuch. Diese Frage soll im Rahmen dieser Arbeit untersucht werden.
Ein Ziel dieser Arbeit ist die Entwicklung eines in SEE integrierten Hilfesystems, dass den Nutzer über die Möglichkeiten und Ausführung von Aktionen aufklärt. Es gilt jedoch zu berücksichtigen, dass SEE sich im laufenden Entwicklungsprozess befindet und eine vollständige, dauerhaft korrekte Entwicklung somit nicht möglich ist, da sich Interaktionen laufend ändern und neue Funktionen hinzukommen. Deshalb ist das Ziel konkreter die Erstellung einer Architektur und Struktur, die eine laufende Weiterentwicklung dieses Hilfesystems ermöglicht, ohne dass ein Entwickler sich erst detailliert in diese Domäne einarbeiten muss.
Aus den vorangegangenen Überlegungen resultierend lautet der Titel dieser Arbeit **Verbesserung der User-Experience von SEE durch ein interaktives Hilfe-System**, in dessen Rahmen folgende Leitfragen beantwortet werden sollen:

- Wie wirkt sich die Integration eines Hilfe-Systems in Bezug auf die Usability von SEE im Vergleich zu einem herkömmlichen Benutzerhandbuch aus?
- Wie effizient ist die Einarbeitung in die Applikation mithilfe des Hilfesystems im Vergleich zu einem Benutzerhandbuch?
- Kann ein Hilfesystem eine Alternative zu einem Benutzerhandbuch bieten bzw. dieses vollumfänglich ersetzen?

1.3 Methodik

Zur Beantwortung dieser Fragen wird in Kapitel 2 zunächst eingehende Recherche zu den fachbezogenen Themen betrieben. Es muss sich in die SEE-Domäne mit Unity und C# eingearbeitet werden und Basiswissen im Umgang mit diesen Technologien erläutert werden. Danach wird der Bereich der Benutzerdokumentationen genauer beleuchtet, welche Anforderungen an diese bestehen und wie Best Practices hierfür konkret ausfallen. Im Zuge dessen wird innerhalb eines Exkurses Literatur zum Lernverhalten von Menschen ausgewertet, damit das Hilfesystem auf dieser Grundlage entwickelt und konzipiert werden und den Nutzern optimal Wissen vermitteln kann.

Auf Grundlage der dargestellten Informationen wird in Kapitel 3 ein Konzept mit Prototypen und Design-Ideen für das Hilfesystem entworfen, welches im nächsten Schritt implementiert wird. Die konkreten Umsetzungen innerhalb der Implementierung und die Herausforderungen, welche diese bot, werden in Kapitel 4 thematisiert.

Nach Abschluss der Implementierung wird in Kapitel 5 das Vorgehen und verschiedene Methoden bei der Nutzerevaluationen sowie deren Auswertungen studiert, woraus ein Fragebogen anhand einer ausgewählten Evaluationsmethode konzipiert wird, der als Grundlage für die folgende Begutachtung und Bewertung dient. Daraufhin wird eine umfangreiche Benutzerstudie durchgeführt, dessen Ergebnisse abschließend ausgewertet und präsentiert werden.

Die Arbeit schließt in Kapitel 6 mit einem zusammenfassenden Fazit ab und gibt in Kapitel Kapitel 7 einen Ausblick auf Weiterentwicklungsmöglichkeiten des Hilfesystems und dessen Zusammenhänge mit SEE.

KAPITEL 2

Grundlagen

Im folgenden Kapitel sollen alle fachbezogenen Grundlagen geschaffen werden, welche zum Verständnis der weiteren Arbeit benötigt werden. Das betrifft insbesondere die Domäne von SEE, d.h. die Grundsätze und Grundlagen, welche der Entwickler oder Leser benötigt, um SEE als Anwendung und die Game-Engine Unity als Laufzeit- und Entwicklungsumgebung für dessen Entwicklung zu verstehen. Weiterhin wird auch das Konzept und der Aufbau von Benutzerdokumentationen thematisiert, um einen gewissen Standard zu definieren und Richtlinien zur Erstellung ebendieser festzulegen. Abschließend findet ein kurzer Exkurs über das Lernverhalten von Menschen statt, welches ebenfalls in der Implementierung des Hilfesystems berücksichtigt werden soll.

2.1 Game-Engine Unity

Die Grundlage für die Entwicklung von SEE ist die Game-Engine Unity. Eine Game-Engine ist eine Entwicklungsumgebung für Spiele bzw. ein Framework zur Spieleentwicklung, welches viele grundlegende Funktionen wie Navigation, Interaktion und auch physikalische Effekte bereitstellt und somit die Entwicklung erleichtert. Dabei trifft Unity einige Grundannahmen und geht von gewissen Strukturen aus, die im Folgenden genauer erläutert werden. Alle hier erarbeiteten Grundlagen basieren auf dem Buch „Spiele entwickeln mit Unity 5“ von Carsten Seifert und Jan Wislauf [Sei15]. Unity ermöglicht die Entwicklung an einer Anwendung innerhalb eines Editors, der das Starten und Verändern der Anwendung während des Entwicklungsprozesses ermöglicht, ohne zuvor eine Executable-Datei aus den vorhandenen Dateien und dem Quellcode erstellen zu müssen. Diese Executable kann der Entwickler jedoch auch jederzeit erstellen und so anderen Nutzern, die Unity nicht installiert haben, die Ausführung zu ermöglichen. Wie eine Anwendung, die mit Unity erstellt wurde, aufgebaut ist und was es bei der Entwicklung zu beachten gilt wird nun genauer erläutert.

Unity-Szene Eine Szene in Unity ist vergleichbar mit einem „Level“ oder einem Ausschnitt eines Spiels. Szenen können kombiniert werden und hintereinander ablaufen, sie können allerdings auch völlig isoliert stehen. Innerhalb dieser Szene befinden sich alle GameObjects und Prefabs etc., aus denen sich eine Anwendung zusammensetzt.

GameObjects Die Klasse GameObject ist die Basisklasse aller Objekte, die sich in einer Unity-Szene befinden. Instanzen dieser Klasse sind sogenannte „Szenen-Objekte“. Das bedeutet, dass jedes Objekt in Unity wie z.B. jedes Gebäude in SEE, der Tisch, auf welchem sich die Code-Cities befinden und auch die Kamera, welche die Sicht des Nutzers darstellt, durch mindestens ein GameObject repräsentiert wird. GameObjects können sowohl im Unity-Editor via Drag and Drop erzeugt werden, als auch zum Anwendungsstart oder zur Laufzeit über den Source-Code der Applikation. Einem GameObject können Komponenten oder weitere GameObjects hinzugefügt werden.

Komponente Eine Komponente ist ein Bestandteil eines GameObjects, die z.B. eine Verhaltensweise dieses GameObjects repräsentiert. Als Beispiel wäre hier ein so genannter Collider zu nennen, eine transparente Fläche, welche sich um ein GameObject legt und Berührungen bzw. Kontakte erkennt und darauf reagieren kann, oder auch ein Rigidbody, der physikalische Eigenschaften wie die Erdanziehungskraft eines Objektes simuliert. Auch ein C#-Script kann eine Komponente eines GameObjects sein.

Prefabs Ein Prefab ist eine Vorlage für ein GameObject, welches in einer Unity-Szene instanziiert werden kann. Das bedeutet, dass eine Veränderung an dem Prefab von GameObjects auch das eigentliche GameObject verändern. Die Nutzung bzw. Erstellung von Prefabs ist dann sinnvoll, wenn der Entwickler dasselbe Objekt regelmäßig benötigt, beispielsweise bei dem Einsatz von Bäumen, Pflanzen oder NPC´s[1] innerhalb eines Spiels oder auch ein wiederkehrendes Menü, in welchem sich nur die Unterpunkte ändern. Die GameObjects, welche aus den Prefabs erzeugt werden, sind isoliert ebenfalls veränderbar. Diese Veränderungen haben keine Auswirkungen auf andere Instanzen bzw. das Prefab selbst. Ein Prefab bzw. die Instanz eines Prefabs kann in Unity entweder manuell via Drag and Drop in die Spieleszene gezogen werden, oder es kann über Quellcode bei Bedarf erzeugt werden.

C#-Scripting Um die Fachlogik, das Verhalten hinter den erstellten GameObjects, zu erstellen, muss man dem entsprechenden GameObject eine Script-Komponente hinzufügen. Unity unterstützt drei Programmiersprachen in diesen Scripts, Javascript, BOO und C#. Als Entwicklungs-Konvention wird SEE mithilfe von C#-Skripten entwickelt. Bei C# handelt es sich um eine objektorientierte, von Microsoft entwickelte Programmiersprache, welche der Programmiersprache Java in ihrer Syntax sehr ähnlich ist. Man muss jedoch zwischen der eigentlichen Programmiersprache C# und C# im Kontext von Unity unterscheiden, da die erstellten Skripte in besonderer Form behandelt werden müssen und besondere Verhaltensweisen bereits vorgegeben sind (siehe Abschnitt 2.1). Jedes einem GameObject hinzugefügte Skript muss dabei von der Klasse MonoBehaviour erben (siehe Abschnitt 2.1).

[1] Non player character - Charaktere, die in ihrer Verhaltensweise automatisiert agieren und nur implementiertes Verhalten beherrschen

MonoBehaviours Jedes C#-Script, das einem GameObject als Komponente hinzugefügt werden soll, muss von der Klasse MonoBehaviour erben. Das begründet sich unter Anderem darin, dass MonoBehaviour einige Funktionen mit sich bringt, welche für die Nutzung des Scripts in Unity elementar sind. So gibt es z.B. Start-, Stop- und Awake-Methoden, welche zum Start bzw. Stop des Spiels aufgerufen werden und eine Update-Methode, welche jeden Frame[2] aufgerufen wird. Start-Methoden werden häufig zur Initialisierung von Objekten genutzt, Update-Methoden sind die meistgenutzten Methoden und enthalten zumeist die eigentliche Logik. Sie sorgen dafür, dass im Spiel zur Laufzeit Dinge geschehen können und Verhalten des Nutzers verarbeitet werden kann. Innerhalb von MonoBehaviours kann auch auf das Eltern-GameObject zugegriffen werden, um dieses zu manipulieren, neue Komponenten hinzuzufügen, zu entfernen oder auch Positionen und Skalierungen zu verändern, es kann jedoch auch auf alle anderen GameObjects der Szene zugegriffen werden[3].

Unity-Store Unity verfügt über einen Asset-Store, in welchem Assets, also Prefabs, Bilddateien, 3D-Modelle etc. zu verschiedenen Preisen angeboten werden. Diese Asset-Pakete können erworben und in ein Projekt integriert werden. Sie verfügen häufig über eine Dokumentation, es steht jedoch nicht immer ein Unternehmen hinter den jeweiligen Paketen, es können auch engagierte Privatentwickler Pakete einreichen und anbieten. In SEE befinden sich ebenfalls einige dieser Assets wie z.B. das Modern UI Pack [Mic21], RTVoicePro [Uni21b] , CurvedUI [Uni21a] und einige weitere. Innerhalb dieser Arbeit ist speziell der Umgang mit dem Modern UI Pack relevant, weil mithilfe dessen alle Menüs innerhalb von SEE erstellt wurden. Da im Rahmen der Entwicklungsaufgabe dieser Arbeit ebenfalls Menüs bzw. Dialogfenster implementiert wurden, wurden diese zur Gewährleistung von Konsistenz innerhalb der Applikation SEE auf Basis des Modern UI Pack entwickelt.

2.2 Benutzerdokumentationen

Die Erstellung einer Benutzerdokumentation ist bei der Entwicklung einer Software von immenser Bedeutung. So bietet diese viele Vorteile für neue, aber auch erfahrene Nutzer. Sie ermöglicht eine effiziente, ressourcenschonende Einarbeitung für neue Nutzer und stellt so ein zusätzliches Qualitätsmerkmal dar. Des Weiteren dient sie als Lernquelle für neue Nutzer bei der detaillierteren Auseinandersetzung mit einer Applikation und kann erfahrenen Nutzern beim Recherchieren nach einzelnen Interaktionen oder Anwendungsfällen helfen.
Dabei ist die Generierung einer Benutzerdokumentation keinesfalls trivial, viele Faktoren müssen berücksichtigt werden[Mra04]:
Zunächst ist die Frage des Adressaten zu klären. Der Adressat der Dokumentation, dessen Erfahrungen mit der Domäne, sein Alter und weitere Charakteristika beeinflussen den Aufbau, die Sprache und die Struktur der Dokumentation. Über die genutzten Termini und Vorkennt-

[2] Ein Videospiel besteht aus vielen, hintereinander abgespielten Bildern, so genannten Frames, die Bewegung und Videos ermöglichen.

[3] GameObjects können über sogenannte Tags gesucht werden, oder auch über dessen Namen per String-Vergleich

nisse muss zuvor recherchiert werden, um zu definieren, welche Begrifflichkeiten als bekannt und welche als unbekannt angenommen werden müssen. Hintergrundinformationen müssen so kurz wie möglich, aber so lang wie nötig verfasst werden. Zuletzt ist die Präsentation der Anwendungsfälle eine relevante Größe in der Erstellung einer Benutzerdokumentation. Dabei sind vorrangig Fragen relevant, die die mediale Unterstützung und die Struktur des Dokumentes betreffen.
Es gibt verschiedene Arten von Benutzerdokumentationen wie z.B. die Embedded Help, die Online-Hilfe, ein pdf-Manual und eine Printversion. Alle bieten Vor-und Nachteile, für eine Anwendung wie SEE sind jedoch besonders Hilfen wie ein pdf-Manual und eine Embedded Help relevant. Ein pdf-Manual ist eine herunterladbare pdf-Datei, welche die Applikation erklärt und Anwendungsfälle in Bild und Schrift beschreibt. Als eine Embedded Help wird eine Hilfe innerhalb der Anwendung bezeichnet, ohne diese schließen zu müssen und dabei weiterhin interagieren zu können.
Da zuvor eingehend die Relevanz einer Benutzerdokumentation beschrieben worden ist, wurde, nachdem all diese Punkte bedacht wurden, mit der Erstellung einer Benutzerdokumentation begonnen. Potentiell kann jedoch zuvor auf individuelle Bedürfnisse der Nutzer eingegangen werden, damit dieser im späteren Verlauf den Umgang mit einer Anwendung wie SEE schneller, effektiver und effizienter erlernen kann. Welche Kriterien hierbei weiterhin relevant sein können wird im folgenden Kapitel thematisiert.

2.3 Exkurs: Lernen und Lernkanäle

Die folgenden Ausarbeitungen beruhen auf dem Werk „Bilden mit Bildern“(2011) von Bergedick, Rohr und Wegener [BRW11]. Dieses Buch befasst sich ausführlich mit Lernprozessen, Lernstilen, Lerntypen und lernförderlichen Faktoren insbesondere im Kontext von visuellem Lernen.
Zunächst wurde ermittelt, dass „verschiedene Sinneskanäle eine wichtige Rolle im Lernprozess spielen“ [BRW11, p. 16]. Es gibt verschiedenste Lernstile und Lerntypen. So lassen sich Lernstile auf circa 13 Modelle reduzieren, woraus sich wiederum zehn Lerntypen herausbilden [BRW11, p. 16]:

- der visuelle Typ
- der auditive oder akustische Typ
- der diskutierende Typ
- der haptische oder motorische Typ
- der psychomotorische Typ
- der olfaktorische oder gustatorische Typ
- der Einsicht anstrebende Typ

- der kontakt- oder personenorientierte Typ
- der abstakt-verbal denkende Typ
- der medienorientierte Typ

Besonders relevant und umsetzbar für das Design eines Hilfesystems wären in diesem Fall also Spezifikationen, die den visuellen Typen, den auditiven Typen, den haptischen oder motorischen Typen und den medienorientierten Typen unterstützen. Diese lassen sich innerhalb eines Hilfesystem am ehesten darstellen.

Darüber hinaus wurden Studien zur Erinnerbarkeit von Informationen in Bezug auf die Übermittlungsart von Informationen durchgeführt. Das Resultat dieser Studien ist in Abbildung 2.1 zu sehen.

Übermittlungsart	Erinnerbarkeit
Hören	ca. 20%
Sehen	ca. 30%
Hören + Sehen	ca. 50%
Hören + Sehen + Reden	ca. 70%
Hören + Sehen + Reden + Tun	ca. 90%

Abbildung 2.1: Übermittlungsart und Erinnerbarkeit (Quelle: Bergedick, Rohr, Wegener (2011))

Die Schlüsse, die sich aus diesen Studien ziehen lassen, sind, dass eine Kombination von mehreren Übermittlungsarten signifikante Auswirkungen auf die Erinnerbarkeit von Informationen hat. So hat die Übermittlungsart des Hörens eine Erinnerbarkeit von 20%, die Übermittlungsart des Sehens hingegen schon eine Erinnerbarkeit von 30%. Die Kombination dieser beiden Übermittlungsarten allerdings führt sogar zu einer Erinnerbarkeit von 50%. Diese Erkenntnisse werden darin zusammengefasst, dass „die beim Lehren und Lernen eingesetzten Aktivitäten, Methoden und Medien abwechslungsreich sein sollten, dass sie verschiedene Sinne einbeziehen sollten“ [BRW11, p. 19]. Für das Design des Hilfesystems bedeutet das, es ist sinnvoll, mindestens die Übermittlungsarten des Hörens und des Sehens anzusprechen. Die Nutzung des Redens wird in der Umsetzung nicht möglich sein, allerdings ist eventuell die Integration des „Tuns“ möglich.

Eine besonders wichtige Rolle nimmt das visuelle Lernen beim Menschen ein. Die visuelle Wahrnehmung ist bei der Mehrheit der Menschen die ausgebildetste Wahrnehmungsart [BRW11, p. 21]. So ist es sinnvoll, zu erlernende Dinge nicht bloß textuell zu präsentieren bzw. zu diktieren, sondern darüber hinaus „in Bildsprache zu übertragen“ [BRW11, p. 19].

Die Begründung dafür, dass die Kombination der Übermittlungsarten sinnvoll ist, ist in der Biologie des Menschen zu finden. So verfügt das menschliche Gehirn über zwei Gehirnhälf-

ten, welche bei effektivem Lernen im Idealfall beide angesprochen werden [BRW11, p. 20]. Da die linke Gehirnhälte aber vorwiegend für das logische Denken zuständig ist, wohingegen die rechte Gehirnhälfte über die kreativen Prozesse verfügt, ist es am effektivsten, „rational-analytische Inhalte mit bildhaften Formen“ [BRW11, p. 20] zu verknüpfen und so das komplette Gehirn zu stimulieren.
Es genügt jedoch nicht nur die Kombination mehrerer Medien und Methoden, besonders relevant ist eine sinnvolle Abstimmung dieser aufeinander [BRW11, p. 21].

Zusammenfassend sind folgende Informationen für die Konzeption des Hilfesystems nutzbar:

- Es gibt verschiedene Lerntypen und Lernstile, bei denen es sinnvoll ist, viele gleichzeitig abzudecken, um möglichst vielen Nutzern optimales Lernen zu ermöglichen. Vorwiegend visuelle, auditive, haptische und medienorientierte Stimulation sind sinnvoll zu integrieren.
- Übermittlungsarten wie Sehen und Hören haben unterschiedliche Erinnerbarkeits-Raten, eine Kombination von Übermittlungsarten erhöht die Erinnerbarkeit signifikant.
- Die Stimulation des gesamten Gehirnes ist für optimales Lernverhalten wichtig. Das bedeutet konkret, dass sowohl die Gehirnhälfte für logisches Denken als auch die Gehirnhälfte für kreatives Denken stimuliert werden sollten.
- Eine schlüssige Abstimmung mehrerer Medien und Methoden aufeinander ist elementar für eine erhöhte Aufnahmefähigkeit.

KAPITEL 3

Entwurf

Im folgenden Abschnitt soll die Entwurfsidee des Hilfesystems thematisiert werden. Hierbei wird zunächst grob zwischen der Navigation innerhalb des System und dem eigentlichen Hilfesystemseintrag unterschieden. Bei der Navigation handelt es sich um ein Navigationsmenü, das den Nutzer zu dem gewünschten Eintrag navigiert, wohingegen der Hilfesystemseintrag den gewünschten Anwendungsfall präsentiert.

3.1 Navigationsmenü

Die erste Herausforderung an das Hilfesystem ist es, den Nutzer bei der Navigation durch eine beliebige Menge an Einträgen zu unterstützen, ohne dass dieser bei der Suche nach seinem spezifischen Eintrag die Übersicht verliert. Er soll innerhalb möglichst kurzer Zeit den Eintrag erreichen können, den er aufrufen möchte. Zu diesem Zweck muss also zunächst ein Konzept erdacht werden, wie dieses Menü strukturiert werden kann. Hierfür müssen einige Optionen der Menüstrukturierung verglichen werden, um das in diesem Kontext optimale Navigationsmenü erstellen zu können.

- **Naiver Ansatz: Sortierte Auflistung** Ein naives Konzept wäre die bloße Auflistung aller Einträge auf derselben Hierarchiestufe. Diese könnte dann auf verschiedenste Arten sortiert sein. Intuitiv würde sich hierfür eine alphabetische Auflistung eignen, eine andere Option wäre eine thematische Gruppierung. Der Vorteil bei dieser Struktur ist seine Einfachheit, dessen Schwäche jedoch die Unübersichtlichkeit speziell bei großen Menüs mit vielen Einträgen und - zumindest bei der alphabetischen Variante - die fehlende thematische Zusammengehörigkeit der Einträge. Der Nachteil der thematischen Gruppierung ist die Zuordnung der jeweiligen Einträge zu einem spezifischen Thema, über welches der Nutzer ebenfalls Kenntnis besitzen muss, um genau dort nach dem gewünschten Eintrag suchen zu können. Diese Gruppierung wird speziell Erstnutzern nicht ohne weiteres möglich sein.
- **Suchfeld** Als ein weiteres Konzept ist der Einsatz eines Input-Fields denkbar, mithilfe dessen ein spezifischer Eintrag gefunden werden kann. Um diese Suche zu optimieren, böte sich an dieser Stelle der Einsatz eines Fuzzy-Search-Suchalgorithmus[1] an, welcher

[1] Ein Suchalgorithmus, der Suchergebnisse nach Übereinstimmungen zum eingegebenen Begriff filtert, so

fehlerverzeihend ist. Darüber hinaus gibt er dem Nutzer mehr Möglichkeiten, falls dieser nicht konkret weiß, wonach er sucht, allerdings bereits über Vorwissen zur Anwendung verfügt und eine grobe Orientierung im Menü besitzt. Dies ist aber auch das wesentliche Problem bei dieser Art von Suche: Der Nutzer muss eine grobe Vorstellung von dem Inhalt haben, den er sucht, um irgendetwas in das Suchfeld eintragen zu können.
Für einen Neueinsteiger ist eine solche Struktur ungeeignet, da er kontextbezogen noch gar nichts über diese Anwendung weiß und daraus resultierend auch nicht weiß, nach welchen Termini er suchen muss, um bestimmte Anwendugsfälle zu finden.
Als erweiterndes Feature kann ein Suchfeld allerdings viele Vorzüge bieten und Suchzeiten verkürzen.

- **Baumhierarchie** Für eine detailliertere Strukturierung eines Menüs ist ein weiteres sich anbietendes Konzept eine Baumstruktur (siehe Abbildung 3.1). Das Menü beginnt bei einem Wurzelknoten, welcher weitere Einträge enthalten kann. Diese Einträge können dann entweder Hilfesystem-Einträge (also die Einträge, die den Inhalt des Hilfesystems anzeigen) sein, oder Navigationseinträge, die weitere Einträge enthalten. Dies ermöglicht eine Strukturierung in Punkte und Unterpunkte, wodurch der Nutzer zu jedem Zeitpunkt weiß, wo er sich befindet. Es muss allerdings auch eine Rückwärtsnavigation möglich sein, um eine Hierarchiestufe zurück zu gehen, um zu anderen Punkten bzw. Unterpunkten navigieren zu können. Vorteilhaft an dieser Struktur ist, dass der Nutzer viel ausprobieren kann und eine Gruppierung trotzdem möglich ist, ein Nachteil ist die Dauer der Suche eines Eintrages, wenn der Nutzer bereits genau weiß, wonach er sucht.

Auf Grundlage - und somit als Resultat - dieses Vergleichs wird eine Baumstruktur als Menüführung für das Hilfesystem ausgewählt.
Da ein erfahrenerer Nutzer optional jedoch wie zuvor beschrieben nur nach bestimmten Kernausdrücken suchen könnte, ist die Integration einer ergänzenden Stichwortsuche ebenfalls sinnvoll. Zuletzt ist auch der Aspekt der Farbe ein Faktor, der die Übersicht des Menüs verbessern kann. Ein Menüeintrag kann sich in seiner Farbe von einem anderen Menüeintrag unterscheiden und so optisch einen semantischen Unterschied suggerieren. Das mag sinnvoll erscheinen, wenn es sich um andere Menüpunkte handelt, manchmal kann es jedoch auch Verwirrung erzeugen. Deshalb wurde festgelegt, dass auf den oberen Hierarchieebenen farbliche Unterscheidungen zwischen den in dem Menü enthaltenen inneren Knoten gemacht werden, die Blätter jedoch immer der Farbe des Elternknotens entsprechen, sodass hier ebenfalls eine eindeutige Zuordnung zum jeweiligen Oberthema möglich ist.

Ein weiterer relevanter Punkt ist die Steuerung eines solchen Menüs auf Geräten, in welchen eine Steuerung über eine Maus schwierig erscheint wie z.B. in der VR bzw. AR. Auch diese Geräte bieten Optionen zur Klick-Steuerung wie z.B. die Trigger[2] bei der HTC Vi-

aber auch Suchergebnisse mit falschen Buchstaben oder Teilsuche ermöglicht. Ergebnisse dieses Algorithmus werden absteigend nach einem Matching-Score angezeigt.

[2]Buttons auf den HTC-Vive Controllern, siehe: `https://www.logando.de/wp-content/uploads/`

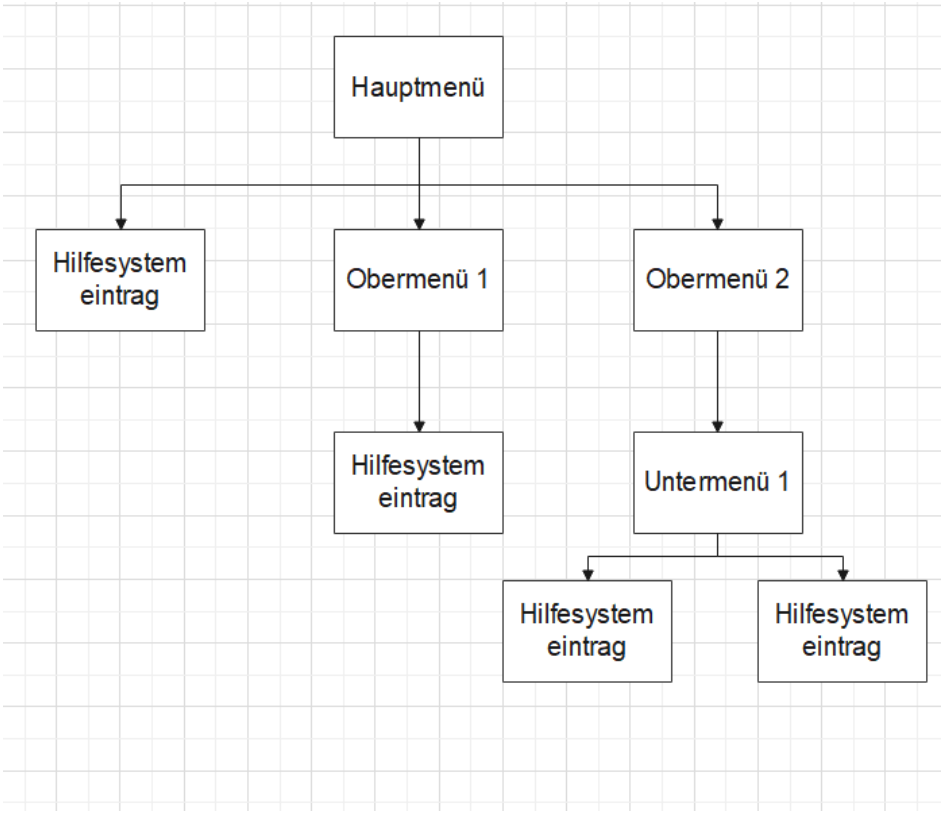


Abbildung 3.1: Ein konkretes Beispiel für ein als Baum strukturiertes Menü

ve[3]. Darüber hinaus unterstützen sie aber auch eine Sprachsteuerung, die für ein solches Menü ebenfalls integriert werden sollte, um die Interaktion mit dem System zu vereinfachen. SEE implementiert für das normale Spielermenü bereits eine Funktionalität, die die Sprachsteuerung von Menüeinträgen ermöglicht. Diese soll also auch für das neue Navigationsmenü übernommen werden. Weiterführende Überlegungen zu Entwicklung in VR und AR sind allerdings nicht Teil dieser Arbeit und werden somit auch weder in der Evaluation, noch in der Implementierung berücksichtigt.

3.2 Hilfesystem-Eintrag

Um nun ein Konzept für den eigentlichen Hilfesystem-Eintrag zu erstellen, muss zunächst ermittelt werden, wie man einem Nutzer optimal Inhalte und Wissen vermittelt. Ferner soll spezifiziert werden, wie man am besten und effizientesten einen großen Bereich an Nutzern mit den entsprechenden Informationen versorgen kann, ohne dass diese Verständnisprobleme entwickeln. Hierfür können die Erarbeitungen aus Abschnitt 2.3 herangezogen werden. Um möglichst viele Lernkanäle zugleich abzudecken und so viele Nutzer optimal zu unterstützen, soll ein Hilfesystemeintrag aus einer textuellen Komponente, einer visuellen Komponente und einer akustischen Komponente bestehen. Diese werden zunächst isoliert entwickelt und

`2017/04/VR-Brille-HTC-Vive-mieten_Handbuch-Deutsch.pdf`[zuletzt abgerufen am 03.01.2022]

[3]Die für SEE genutzte Hardware zur Realisierung von Virtual Reality

betrachtet und final kombiniert.

3.2.1 Visuell

Zur visuellen Präsentation eines Anwendungsfalls in SEE gibt es mehrere Optionen:
Zunächst könnten Bilder oder auch Screenshots verwendet werden, die nacheinander angezeigt werden. Das Problem hierbei ist jedoch, dass wichtige Informationen verloren gehen können wie z.B. einzelne Mausklicks oder Perspektivenwechsel.
Eine weitere Variante wäre der Einsatz eines Videos. Der Vorteil hiervon ist sowohl seine Einfachheit bei der Erstellung, seine einfache Integration, als auch dessen große Menge an Informationen, da ein Anwendungsfall von Anfang bis Ende systematisch vorgeführt werden kann. Ein wesentlicher Nachteil ist in der Dynamik des Videos zu erkennen. Sobald sich der Interaktionsweg eines Anwendungsfalles durch den fortschreitenden Entwicklungsprozess verändert, muss das Video ersetzt werden, da es keine Möglichkeit zur Veränderung bietet.
Eine letzte Variante wäre der Einsatz einer Beispielszene bzw. eines Beispielfalls innerhalb der Applikation, der abgespielt wird. Der Informationsgehalt zum Video ist identisch, der erkennbare Vorteil ist aber die Weiterentwickelbarkeit des Eintrages. Sobald sich ein Interaktionsweg verändert, verändert sich dieser auch im Hilfesystem. Ein wesentlicher Nachteil ist in der Erstellung solcher Szenen bzw. Anwendungsfälle, der Erstellung von Beispielbenutzern für Navigationshilfen und der Erstellung von Hilfesystemeinträgen von Entwicklern ohne Kontexterfahrungen ersichtlich, da dort spezifischer Code ausgeführt werden müsste, über dessen Funktionsweise der Entwickler des Hilfesystemeintrages spezifisches Fachwissen benötigt.

Nach der Abwägung von Vor- und Nachteilen der einzelnen Präsentationsstile wird deshalb ein Video als visueller Präsentationsweg genutzt, da dieses verhältnismäßig einfach in seiner Erstellung ist, einen hohen Informationsgehalt besitzt und kaum implementierungsspezifisches Wissen des Erstellers benötigt. Die Abstriche in Bezug auf dessen Flexibilität werden nach sorgfältiger Abwägung aufgrund der großen Vorteile der anderen Aspekte in Kauf genommen.

3.2.2 Akustisch

Um Nutzer zu unterstützen, die gerne über Akustik lernen, soll das Video während des Abspielens mit einer gesprochenen Erklärung fundiert werden. Zunächst muss hierfür entschieden werden, ob die Sprache innerhalb von SEE über z.B. das bereits integrierte RT-Voice PRO abgespielt werden soll, oder ob sie in das abgespielte Video integriert sein kann. Hierzu müssen erneut Vor- und Nachteile abgewogen werden. Eine Integration in das Video ist bei der Erstellung wesentlich einfacher, ermöglicht jedoch kaum Flexibilität und erzwingt beim Ersatz des Videos eine vollständig neue Tonaufnahme.
Dagegen bietet die Sprachausgabe mithilfe von RT-Voice PRO den Vorteil der Veränderbarkeit, erfordert aber mehr Aufwand bei der Erstellung, da Zeitstempel gesetzt werden müssen, die die Zeitpunkte der Ausgabe einzelner Sätze synchron zur Handlung im Video gewährleisten.

Da eine größere Flexibilität in diesem Fall einige Vorteile bietet und dieselbe Sprachausgabe und Stimme genutzt werden kann wie an anderen Stellen in SEE, wodurch eine größere Konsistenz innerhalb der Anwendung sichergestellt wird, wurde sich für die Ausgabe der Sprache durch RT-Voice PRO entschieden. Die gesprochenen Sätze müssen dabei zuvor als Zeichenketten an die Anwendung übergeben werden, damit sie daraufhin vom RT-Voice PRO wiedergegeben werden können. Es wird also ein Datentyp benötigt, der in einer verlinkten Liste abgespeichert wird, wodurch alle Einträge nacheinander zu bestimmten Zeitpunkten abgespielt werden. Eine verlinkte Liste genügt an dieser Stelle, da immer nur der vorherige bzw. darauffolgende Eintrag abgespielt wird. Dabei ist jedoch wichtig zu beachten, dass nach der Wiedergabe des letzten Eintrages erneut der erste Eintrag abgespielt werden soll. C# bietet an dieser Stelle die LinkedList als Konzept, welche dort als doppelt verkettete Liste implementiert wird.

3.2.3 Textuell

Als letzter wesentlicher Bestandteil für den zu erstellenden Hilfesystem-Eintrag ist die textuelle Ausgabe zu nennen. Durch den in Abschnitt 3.2.2 entworfenen LinkedListEntry existiert bereits ein Erklärungstext, der hierzu genutzt werden kann. Es muss lediglich dafür gesorgt werden, dass diese Stichpunkte zur richtigen Zeit angezeigt werden, was ebenfalls mit der Synchronisierung von Video, Sprache und Text im vorherigen Eintrag organisiert werden kann. Der Benutzer soll also zum Zeitpunkt der Sprachausgabe durch das RT-Voice PRO eine Abschrift des diktierten Textes im Hilfesystemeintrag erhalten. Um eine bessere Übersicht über bereits Gesprochenes zu behalten sollen allerdings immer der aktuell gesprochene Satz und all seine Vorgänger angezeigt werden. Das ermöglicht, dass der Nutzer bereits abgeschlossene Handlungen im Video weiterhin lesen kann, ohne das Video erneut starten zu müssen. Konkret bedeutet das, dass ein Eintrag mit fünf diktierten Stichpunkten bei einem aktuell vorgetragenen Stichpunkt „drei“ die Stichpunkte eins, zwei und drei abbilden soll. Konzeptionell muss hierbei noch bedacht werden, dass am Ende der Auflistung bei Neustart des Videos auch alle Stichpunkte gelöscht werden müssen, damit diese dann erneut abschnittweise angezeigt werden können.

3.2.4 Weitere Anforderungen

Dieser Abschnitt listet weitere Anforderungen auf, die zwar alle für sich eine Relevanz für den Entwurf besitzen, jedoch nicht sinnvoll innerhalb der oberen Struktur gruppiert werden konnten.

3.2.4.1 Skalierbarkeit

Unter Umständen ist es dem Benutzer wichtig, einen Anwendungsfall gleichzeitig zum Abspielen des Hilfesystem-Eintrages auszuführen, um diesen besser nachvollziehen zu können. Aus diesem Grund ist eine Skalierbarkeit des gesamten Eintrages bzw. des Fensters, in wel-

chem sich der Eintrag befinden soll von großer Relevanz. Die Herausforderung hierbei besteht darin, dass alle Komponenten innerhalb dieses Fensters, also sowohl die angezeigten Texte als auch das Video sinnvoll mitskalieren müssen, sobald der Eintrag vergrößert bzw. verkleinert wird. Eine Beweglichkeit des gesamten Eintrages muss auch implementiert werden, sollte aber in Relation zum vorigen Problem einfacher umzusetzen sein, da hierbei keine Größenverhältnisse verändert werden und alle Relationen innerhalb des Eintrages gleich bleiben, sich nur ihre absolute Position verschiebt.

3.2.4.2 Navigierbarkeit

Mit diesem Aspekt ist die Navigierbarkeit des Videos bzw. der einzelnen Abschnitte der Sprach- und Textausgabe gemeint. Bisher wurde lediglich angenommen, dass ein Eintrag zum Abschluss des Videos erneut starten soll, dies soll jetzt jedoch noch um den Aspekt erweitert werden, dass der Nutzer einen Eintrag und all seine Komponenten pausieren, nach vorne springen und zurück springen können lassen soll. Das erfordert für den Nutzer darüber hinaus eine Anzeige, an welcher Stelle des Eintrages er sich befindet. Hierfür wäre eine Zeitanzeige möglich und spezifische Sprungzeiten von z.B. 10 Sekunden pro Klick, sodass der Nutzer weiß, dass ein Eintrag z.B. 45 Sekunden lang ist und er sich bei Sekunde 15 befindet, sich mit einem Sprung nach vorne bei Sekunde 25 befinden wird bzw. mit einem Sprung zurück bei Sekunde fünf.
Als ein sinnvolleres Konzept erscheint in diesem Kontext jedoch die Anzeige der einzelnen Listeneinträge und Sprungmarken zwischen diesen. Das bedeutet konkret, dass der Benutzer zu jedem einzelnen Listeneintrag springen kann und immer zum Beginn einer Sprachausgabe bzw. Textausgabe startet, wodurch die inhaltliche Struktur und Logik der Teil-Einträge[4] bestehen bleibt. Dies könnte konkret so aussehen, dass der Nutzer die absolute Anzahl an Listeneinträgen sieht und relativ dazu seine Position innerhalb des Eintrages, z.B. 2/5 und mit einem Sprung nach vorne bei 3/5 startet bzw. bei einem Sprung zurück bei 1/5. Für die Implementierung bedeutet das, dass als Sprungmarken jeweils die Zeitstempel der doppelt verketteten Liste genutzt werden.

3.2.5 Zusammenfassung

Nimmt man die zuvor ermittelten Ideen und Entwürfe zusammen, muss hieraus noch ein schlüssiges Darstellungskonzept entwickelt werden, welches alle Komponenten gleichmäßig berücksichtigt:
Im Hilfesystemeintrag soll ein Video abgespielt werden, welches parallel dazu sowohl durch eine textuelle Ausgabe, als auch durch eine Sprachausgabe in Form von Stichpunkten unterstützt wird. Weitere Anforderungen an diesen Hilfesystemeintrag ist die Navigierbarkeit innerhalb der Abschnitte dieser drei Komponenten und die Skalierbarkeit des Eintrages.

[4] Als ein „Teileintrag“ wird ein logischer Abschnitt eines Hilfesystem-Eintrages bezeichnet, welcher einen Abschnitt des Videos, in welchem ein Text abgespielt und diktiert wird, darstellt

In der folgenden Skizze ist ein beispielhafter Entwurf für einen Hilfesystems-Eintrag dargestellt, der als Vorlage für die konkrete Implementierung dienen soll.

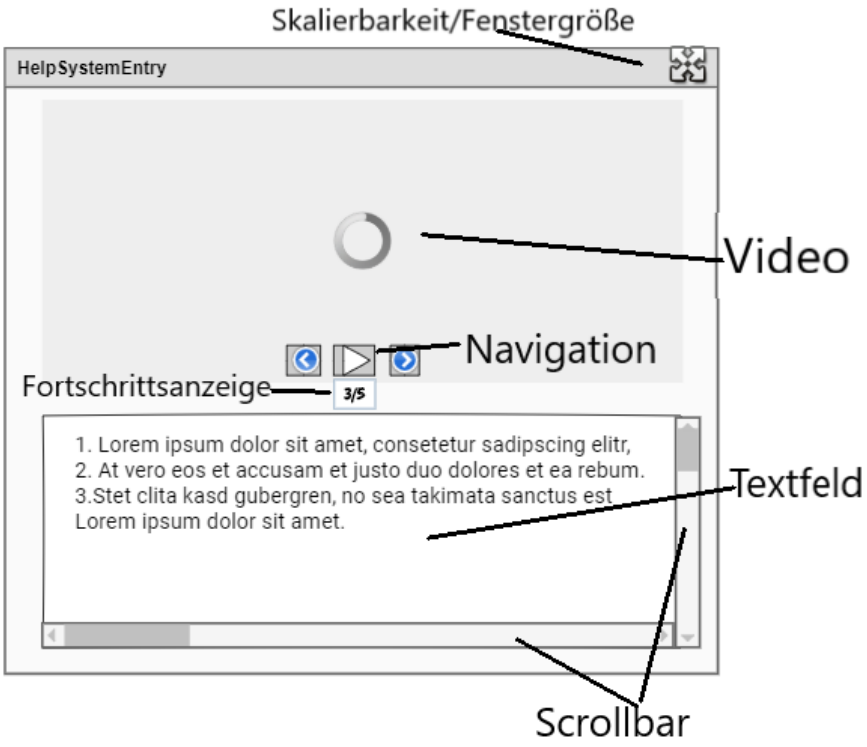


Abbildung 3.2: Beschriftete Skizze eines Hilfesystem-Eintrages

3.3 Weiterentwickelbarkeit

Ein weiterer relevanter Punkt in der Entwicklung des Hilfesystems ist seine Weiterentwickelbarkeit. Im Rahmen dieser Arbeit soll lediglich eine Struktur entstehen, die mit einigen beispielhaften Daten gefüllt ist. Dieses System soll von jedem Entwickler genutzt werden können, um mit geringem Aufwand einzelne Anwendungsfälle zu ergänzen und so einen dauerhaften Einsatz zu ermöglichen. Dafür müssen bereits vorher einige Dinge berücksichtigt werden und entsprechende Strukturen geschaffen werden. Zusätzlich müssen aber auch einige Grundannahmen getroffen werden, die jeder Entwickler kennen muss, um mit dem System zu arbeiten:

- Der Entwickler muss seinen Hilfesystemseintrag in das Navigationsmenü integrieren, damit dieser erreichbar und aufrufbar ist. Hierfür müssen entweder neue Untermenüs erstellt, oder nur der atomare Hilfesystemseintrag in einem Menü bzw. Untermenü ergänzt werden.
- Der Entwickler muss einen Pfad zu einem Video im Format 16:9 bzw. 1,77:1 angeben, welches er zuvor erstellt hat. Diese Annahme muss getroffen werden, um Relationen innerhalb des Hilfesystems sinnvoll erhalten zu können.

- Der Entwickler muss eine Liste an LinkedListEntries erstellen (siehe **??**), welche beim Eintrag-Start abgespielt werden und die Zeitstempel innerhalb dieser händisch mit den in dem Video dargestellten Interaktionen synchronisieren, damit die Text- und Sprachausgabe synchron zu diesen Handlungen abgespielt bzw. angezeigt werden können.

Zur sinnvollen Umsetzung sollen hierfür einzelne Methoden erstellt werden, die lediglich diese Parameter wie Videopfade und LinkedListEntries übergeben bekommen und die restliche Funktionalität abstrahieren.

KAPITEL 4

Implementierung

Die zuvor in Kapitel 3 entworfene Architektur und Struktur soll nun konkret umgesetzt werden. Thematisiert werden hierbei die genutzten Strategien, Frameworks und Features, aufgetretene Probleme sowie weitere Herausforderungen. Abschließend wird die Umsetzung auch in grafischer Form präsentiert. Um die zuvor erarbeiteten Aspekte sinnvoll aus dem Entwurf auf die Implementierung übertragen zu können, wird dieses Kapitel in seiner Struktur dem vorherigen nahezu gleich aufgebaut sein und die zuvor konzeptionell entworfenen Teilaspekte konkretisieren.

4.1 Navigationsmenü

Innerhalb der Anwendung SEE werden bereits an einigen Stellen Menüs zur Navigation benutzt, wie z.B. das PlayerMenu (siehe Abbildung 4.1). Dieses besteht aus mehreren Reitern, die jeweils für eine Benutzerinteraktion stehen und ermöglicht dem Benutzer, durch das Anklicken des jeweiligen Reiters eine Interaktion auszuführen.

Um eine Konsistenz innerhalb der Anwendung zu gewährleisten, wurde somit ein Design sehr ähnlich zu dem des Player Menus verwendet, welches vorwiegend auf Komponenten des ModernUIPacks basiert. Darüber hinaus implementiert dieses Menü bereits viele Funktionen, welche in einem verschachtelten Menü auch benötigt werden, weshalb eine Erweiterung durch eine vom SimpleMenu[1] erbende Klasse als sinnvollste Option erscheint. Diese Kindklasse wird im Folgenden als NestedMenu bezeichnet. Das Navigationsmenü wurde zunächst durch die Erstellung eines Prefabs in Unity designt und daraufhin durch die Erstellung einer C#-Klasse **NestedMenu.cs** mit Fachlogik gefüllt.

Dieses NestedMenu benötigt zwei wesentliche Erweiterungen im Vergleich zum herkömmlichen SimpleMenu:

- Eine Hierarchie-Struktur in Form eines Baumes, welche navigierbar ist.
- Ein Suchfeld für die zuvor bereits erwähnte Fuzzy-Search.

[1] Die Klasse, auf der auch das PlayerMenu basiert

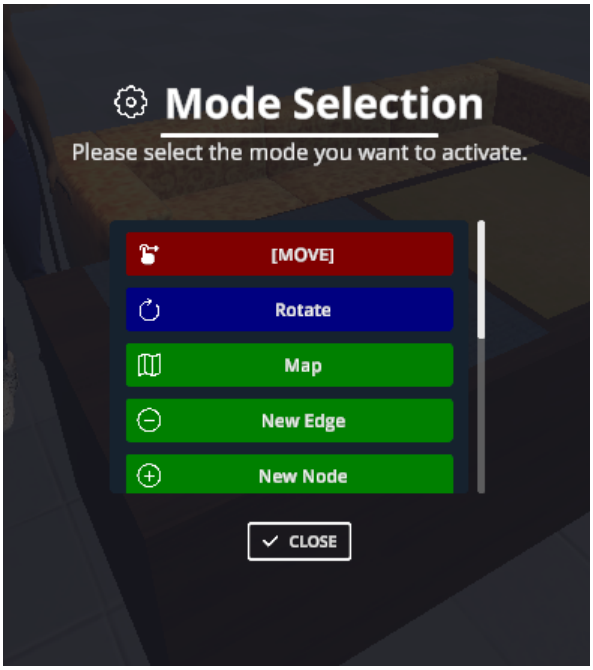


Abbildung 4.1: Das bereits in SEE verwendete Player Menu

Das Design des neu erstellten NestedMenus ist in Abbildung 4.2 zu erkennen. Zur Veranschaulichung wurde dieses mit einigen exemplarischen Werten und Einträgen gefüllt. Die zuvor aufgelisteten Besonderheiten im Vergleich zum SimpleMenu werden in den nachfolgenden Abschnitten genauer beschrieben.

Baumstruktur Die Herausforderung, dass ein Eintrag auch weitere Einträge enthalten kann, wurde so gelöst, dass eine weitere Klasse **NestedMenuEntry.cs** erstellt wurde, welche eine Liste an darin enthaltenen Einträgen enthält. Diese erbt von der Klasse **MenuEntry.cs**, die innerhalb der anderen Menüs hinter jedem Eintrag für dessen Ausführung verantwortlich ist. Die in dem Menü enthaltenen Einträge können dann entweder NestedMenuEntries sein, oder auch herkömmliche MenuEntries. NestedMenuEntries übernehmen die Navigation durch die Struktur des Menüs, wohingegen MenuEntries eine Aktion ausführen, in diesem Fall die Erklärung der Suche nach einem Knoten. Auch hier ist das Konzept eines Baumes anwendbar - so handelt es sich bei einem NestedMenuEntry um einen inneren Knoten, der weitere Knoten enthält, bei einem MenuEntry hingegen um einen Blattknoten, der dann einen Hilfesystems-Eintrag öffnet.
Eine Unterscheidung dieser beiden Arten von Einträge wurde durch die Auswahl des Icons des Eintrages sichtbar gemacht: So wird ein referenzierender Eintrag durch ein Plus gekennzeichnet, während ein Hilfesystemseintrag durch ein Auge dargestellt wird. Sobald der Benutzer einen Referenzeintrag bzw. NestedMenuEntry innerhalb des Menüs anklickt wird dieses mit den darin enthaltenen Werten überschrieben und ist somit eine Hierarchiestufe tiefer als noch zuvor. Als Blatt der bereits thematisierten Baumstruktur dienen dann wiederum die herkömmlichen MenuEntries zum Aufruf des Hilfesystem-Eintrages. Über einen **Back**-Button ist diese Funktion bis zur obersten Hierarchiestufe rückgängig zu machen, sodass eine Navi-

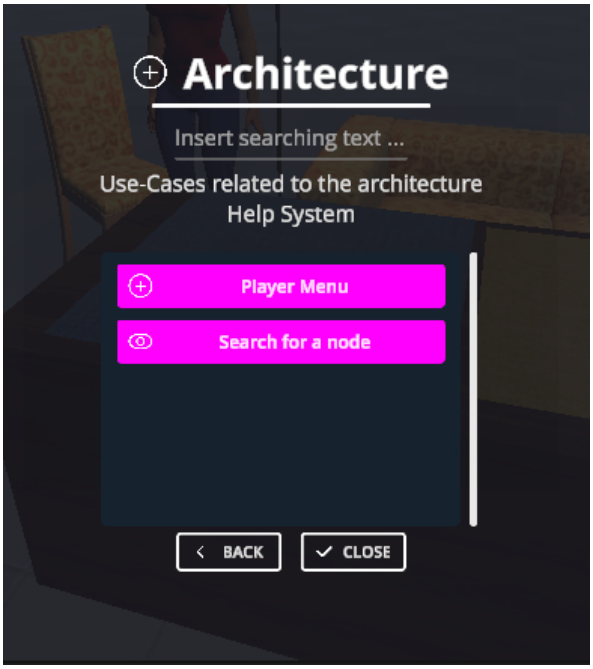


Abbildung 4.2: Das neu erstellte NestedMenu für das Hilfesystem

gation in beide Richtungen ermöglicht wird. Beim Schließen des Menüs wird das Menü, egal auf welcher Hierarchiestufe sich der Nutzer zu diesem Zeitpunkt befindet, auf die oberste Stufe zurückgesetzt.
Ein weiteres Feature, welches aufgrund der Baumstruktur des Menüs hinzugefügt wurde, ist eine Art Breadcrumb[2], die dem Nutzer den bisher verfolgten Pfad anzeigt. So erhält ein Benutzer, der im Help System den Eintrag Architecture - Player Menu ausgewählt hat einen Hinweis Help System / Architecture oberhalb der Einträge des Untermenüs.

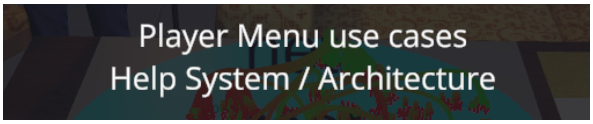


Abbildung 4.3: Ein Beispiel für Breadcrumbs

Suchfeld Um dem Nutzer ergänzend die Suche über ein Suchfeld zu ermöglichen, wurde dem Prefab des NestedMenu´s ein Eingabefeld hinzugefügt, das über das hinterlegte NestedMenu.cs angesteuert werden kann. Dieses wartet auf Eingaben und wertet diese nach jedem Tastendruck aus, sodass dann die am besten übereinstimmenden Einträge im Navigationsmenü angezeigt werden. Um zu definieren, wann ein Eintrag mehr bzw. weniger übereinstimmt, wurde ein Suchalgorithmus genutzt, der die Funktionalität der Fuzzy-Search bereits implementiert und die Resultate nach einem Matching-Score anzeigt. Die zuvor erwähnten Resultate werden im Hilfesystem absteigend nach ihrem Matching-Score geordnet. FuzzyS-

[2]Bei dem Entwurfsmuster der Breadcrumbs soll im herkömmlichem Sinne dem Nutzer die Navigation durch eine Baumstruktur erleichtert werden, indem durch die Breadcrumbs der Pfad abgebildet wird, welchen der Nutzer durch mehrere Hierarchiestufen hindurch aufgerufen hat.

harp[3] stellte sich hier als eine geeignete Bibliothek heraus und ist unter der **MIT-LICENSE**[4] lizensiert, weshalb sie innerhalb von SEE nutzbar ist. Darüber hinaus wird diese Bibliothek bereits an anderen Stellen innerhalb von SEE genutzt, wie z.B. bei der Knotensuche. Nachdem nun die Navigation zu den Hilfesystems-Einträgen implementiert wurde, muss jetzt der eigentliche Hilfesystems-Eintrag konkretisiert werden.

4.2 Hilfesystem-Eintrag

Diese Sektion beschreibt die Implementierungsdetails und die Umsetzung des eigentlichen Hilfesystem-Eintrages. Auch hier wird eine isolierte Betrachtung der einzelnen Komponenten durchgeführt. Zunächst wurde ein Prefab designt, welches dem Wireframe[5] in Abbildung 3.2 ähnelt. Das Resultat hieraus ist in Abbildung 4.4 zu erkennen.

Abbildung 4.4: Das Design des Hilfesystem-Eintrages in SEE

4.2.1 Visuell

Das Abspielen des Videos wurde mithilfe des in Unity integrierten VideoPlayers[6] durchgeführt. Dieser verfügt über alle nötigen Funktionen zum Abspielen des Videos und auch über

[3] Einzusehen und herunterzuladen über: `https://github.com/JakeBayer/FuzzySharp` [zuletzt abgerufen am 21,11.2021]

[4] Eine Software-Lizenz, einzusehen unter: `https://mit-license.org/` [zuletzt abgerufen am 21.11.2021]

[5] Als ein Wireframe (Drahtmodell) wird ein Grundgerüst bzw. die grafische Konzeption einer (Teil-) Anwendung bezeichnet.

[6] Dokumentation einsehbar unter: `https://docs.unity3d.com/Manual/class-VideoPlayer.html` [zuletzt abgerufen am 21.11.2021]

Navigationsmöglichkeiten innerhalb des Videos. Die eigentliche Abbildung des Videos muss allerdings über eine grafische Komponente vollzogen werden. Hierfür wurde ein RawImage[7] genutzt. Dieses wird als Ziel zur grafischen Präsentation in der VideoPlayer-Komponente hinzugefügt und ermöglicht so eine Anzeige des abgespielten Videos. Der Video-Player benötigt dann lediglich noch einen Pfad zu dem Video, das dann direkt zum Aufruf des Eintrages mithilfe einer **Play()**-Funktion abgespielt wird. Dieses Video muss zuvor in einem bestimmten Ordner hinterlegt werden und erfordert ein Seitenverhältnis von ca. 1,77:1, damit die Relationen innerhalb des Eintrages passend bleiben. Die weiteren Navigationsmöglichkeiten, die der VideoPlayer ermöglicht, werden in Kapitel 4.2.4.2 aufgegriffen.

4.2.2 Akustisch

Das in SEE integrierte Package RTVoice-PRO bietet eine statische Klasse **Speaker**, die Funktionen wie **Speak()**, **PauseOrUnpause()** oder **Pause()** bereitstellt. So ist die Aussprache einer Zeichenkette für dieses Framework mit einem Funktionsaufruf problemlos möglich. Damit jedoch bestimmt werden kann, welcher Eintrag der in Kapitel 3.2.2 konzipierten Liste abgespielt werden soll, muss der jeweilige Zeitstempel in einem Eintrag ausgewertet werden. Dafür wird die bereits abgelaufene Zeit des Videos mit dem jeweiligen Zeitstempel des aktuellen Eintrages der Liste verglichen. Aus diesem Grund ist es wichtig, dass jeder Eintrag die kumulierte Zeit seines Abspielzeitpunktes und aller vorherigen Einträge erhält, damit der Vergleich zwischen diesem Zeitstempel und der Videodauer auch sinnvoll ist.
Bei Übereinstimmung wird dieser Eintrag abgespielt und der nächste Listeneintrag betrachtet, bis der Zeitstempel dieses Eintrages mit der aktuellen Zeit des Videos übereinstimmt. Beim Überlauf der Liste, wenn also der letzte Eintrag abgespielt wurde, wird erneut der erste Eintrag betrachtet.

4.2.3 Textuell

Zur textuellen Darstellung der vorgesprochenen Kernaussagen wird die Unity-Klasse und Komponente TextMeshPro[8] genutzt. Diese lässt die Anzeige und das Rendering von Texten innerhalb einer Unity-Szene zu. Auch dieser Teil ist von den Zeitstempeln, dem Video und dem Inhalt der Liste abhängig. Konzeptionell funktioniert dieser Teil sehr ähnlich zu der Sprachausgabe. Sobald ein Zeitstempel erreicht wurde, wird der TextMeshPro-Komponente der jeweilige Text des Listeneintrages, um einen Zeilenumbruch ergänzt, hinzugefügt. Ein wesentlicher Unterschied besteht darin, dass die Anzeige bzw. Löschung der vorherigen Einträge ebenfalls verwaltet werden muss. So muss der neueste Eintrag so lange der Text-Mesh-Pro-Komponente hinzugefügt werden ohne die vorherigen Einträge zu entfernen, bis der letzte Eintrag erreicht ist. Sobald das Video neu startet, muss der hinzugefügte Text vollständig

[7] Dokumentation einsehbar unter: `https://docs.unity3d.com/2018.3/Documentation/ScriptReference/UI.RawImage.html` [zuletzt abgerufen am 21.11.2021]

[8] Dokumentation einsehbar unter: `https://docs.unity3d.com/Manual/com.unity.textmeshpro.html` [zuletzt abgerufen am 21.11.2021]

gelöscht werden. Daraufhin beginnt dieser Prozess erneut. Da dieselben Zeitstempel zur Anzeige wie auch zum Diktieren des Eintrages genutzt werden, ist dies bereits synchronisiert, was bedeutet, dass ein neuer Eintrag immer dann angezeigt wird, wenn er auch diktiert wird. Da die Text-Mesh-Pro-Komponente allerdings nur über einen eingeschränkten Platz verfügen kann, mussten Scrollbars ergänzt werden, die die Anzeige von Text zulassen, der sich außerhalb des Anzeigebereichs befinden würde. Dies gilt sowohl im vertikalen als auch im horizontalen. Hierfür bietet das ModernUIPack Scrollbar-Komponenten an, die einfach dem Prefab hinzugefügt werden konnten.

4.2.4 Weitere Anforderungen

4.2.4.1 Skalierbarkeit

Zur Skalierung des Hilfesystems-Eintrages wurde eine weitere Bibliothek genutzt, welche sich um die dynamische Skalierung von Panels kümmert, die **Dynamic Panels**[9]. Diese Bibliothek wird bereits an anderen Orten in SEE genutzt, wie z.B. den Code-Windows (siehe Abbildung 4.5). Darin integriert sind sowohl die Skalierung des Eintrages über ein Ziehen der jeweiligen Ränder auf der Benutzeroberfläche, als auch die Bewegung des gesamten Eintrages als Ganzes über eine dem Rahmen hinzugefügte Drag-Komponente im oberen, grauen Bereich des Eintrages. Theoretisch bietet es auch die Möglichkeit der Öffnung mehrerer Tabs zur selben Zeit, dies wird allerdings im Kontext des Hilfesystems nicht benötigt und ist somit zu vernachlässigen.

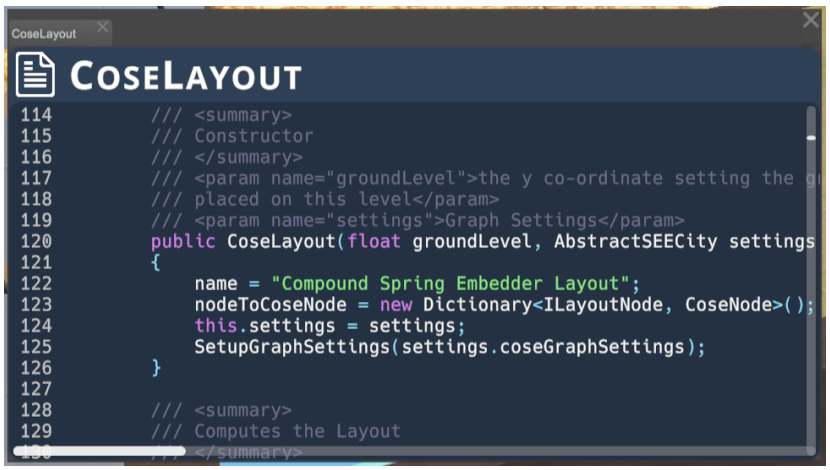


Abbildung 4.5: Die Dynamic Panel - Komponente anhand eines Code-Windows

Um diese Bibliothek für den Hilfesystemeintrag zu nutzen, wurde ihre ausführliche Dokumentation studiert. Da ein Prefab hierarchisch aufgebaut ist, war es von großer Relevanz, dass alle Kindkomponenten wie das Video, der Text und auch die Navigationsbuttons (siehe Abbildung 4.4) auch sinnvoll skalieren und ihre relativen Positionen im Eintrag erhalten. Die Dynamic Panel-Komponente wird daraufhin wie ein Wrapper um das vorherige Prefab gelegt.

[9] Dokumentation einsehbar unter: `https://github.com/yasirkula/UnityDynamicPanels` [zuletzt abgerufen am 21.11.2021]

Um die Skalierung der Kindkomponenten zu gewährleisten, wurden zum einen die von Unity bereitgestellten Komponenten **Vertical Layout Group** und **Horizontal Layout Group** genutzt. Diese organisieren Relationen zwischen mehreren Kindkomponenten. Zum anderen mussten so genannte Ankerpunkte und Pivotpunkte gesetzt werden. Unity bietet hierzu die Möglichkeit, diese absolut bzw. relativ und horizontal bzw. vertikal zu setzen (siehe Abbildung 4.6)[10]. Vereinfacht ausgedrückt wird durch das Setzen des Pivot-Punktes festgelegt, an welchem Punkt des Parent-Objektes sich ein GameObject orientiert bzw. in diesem Fall, um welchen Punkt herum das Objekt skaliert. Dieser Pivot-Punkt muss für jedes GameObject im Prefab passend ausgewählt werden.

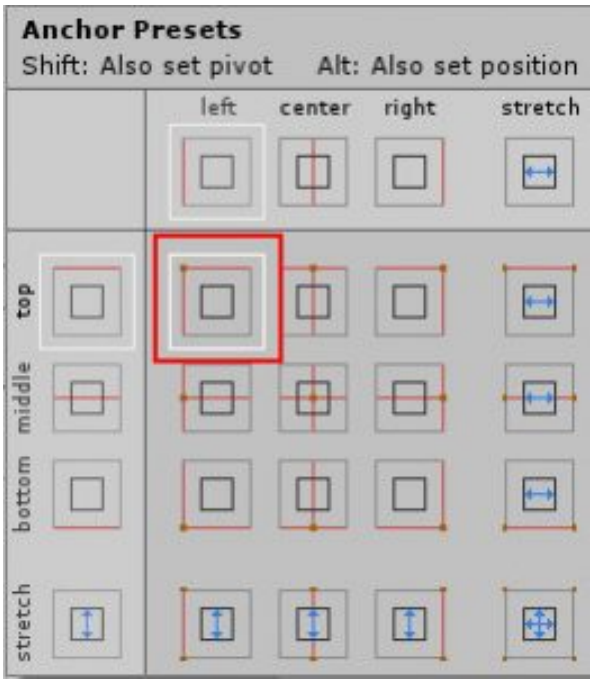


Abbildung 4.6: Das Interface von Unity zur Bearbeitung der Ankerpunkte eines GameObjects

Die Skalierung des Eintrages funktioniert durch die angepassten Anker- und Pivotpunkte ebenfalls für alle Kind-Komponenten des Hilfesystem-Eintrages, wie in Abbildung 4.7 beispielhaft zu sehen ist. So kann der Nutzer sogar parallel zur Betrachtung des Eintrages Interaktionen in SEE ausführen.

4.2.4.2 Navigierbarkeit

Zur Umsetzung einer Navigation musste lediglich noch ein Navigations-Design entwicklelt werden, nachdem das Konzept bereits erdacht war. Hierfür wurde ein aus drei Buttons bestehende Navigationsbar implementiert - ein Button für das Abspielen/Pausieren des Eintrages, ein Button für das Vorspulen, ein Button für das Zurückspulen.

[10] Ein Beispiel anhand eines Shooter-Spiels einsehbar unter: `http://gamecodeschool.com/unity/unity-2d-shooter/` [zuletzt abgerufen am 21.11.2021]

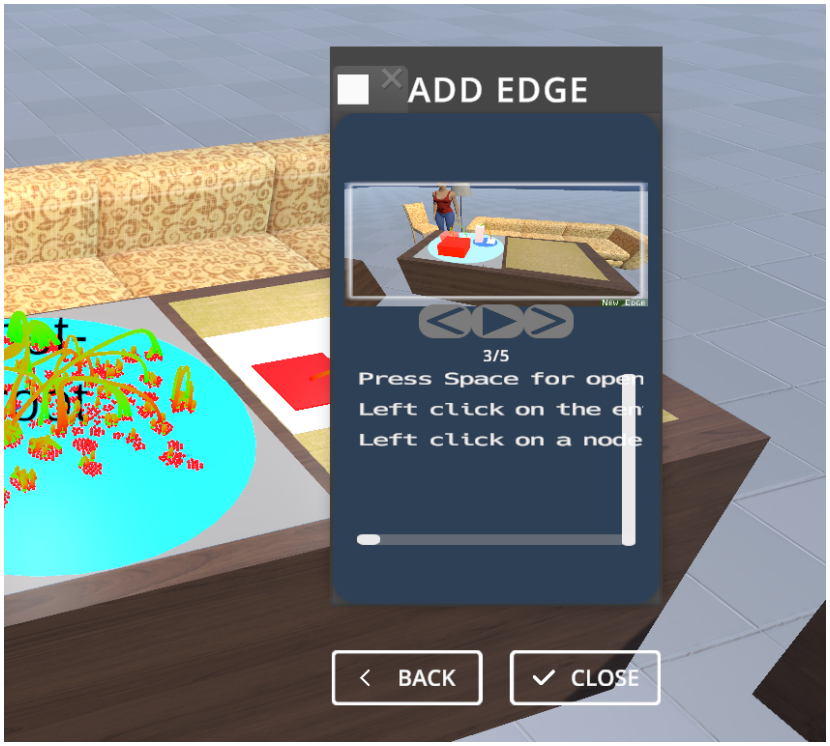


Abbildung 4.7: Ein skalierter Hilfesystems-Eintrag

Abbildung 4.8: Die integrierte Nagivations-Bar

Eine zu lösende Herausforderung hierbei war speziell das Springen zum vorherigen Eintrag. Sobald ein Teileintrag abgespielt wird, man nehme an, Eintrag 3/5 wird abgespielt - muss der „Zurück“-Button zweierlei Funktion haben: Zum einen muss er den Teileintrag von vorne abspielen können (es wird erneut Eintrag 3/5 abgespielt), zum anderen muss er aber auch einen Sprung zum vorherigen Teileintrag ermöglichen (Eintrag 2/5 wird abgespielt).
Ein gängiges Konzept hierfür ist die Wahl einer maximalen Abspieldauer eines Teileintrages, bis dieser erneut abgespielt wird. So hat der Benutzer innerhalb des Hilfesystems die erste Sekunde eines Teileintrages Zeit, auf den vorherigen Eintrag zurückzuspringen, danach spielt er nur noch den gerade laufenden Eintrag erneut ab. Ein Doppelklick hingegen ermöglicht dann auch eine Navigation zum vorherigen Eintrag - zunächst den Sprung zum Anfang des Teileintrages und mit dem zweiten Klick einen Sprung zum vorherigen Eintrag, da die Sekunde noch nicht abgelaufen ist.
Die Überschreitung von Grenzwerten (z.B. das Springen vom ersten zum letzten Eintrag über ein „Zurück“ und das Springen vom letzten zum ersten Eintrag) mussten auch bedacht werden. Die Logik des Vorwärts-springens ist jedoch durch die normale Logik und Funktionalität des Eintrages und die fortlaufende Zeit bereits abgedeckt. Lediglich das Springen vom ersten

zum letzten Eintrag erforderte gesonderte Fachlogik. So muss die vollständige Liste an Teileinträgen angezeigt und der letzte Eintrag abgespielt werden.
Die letzte Herausforderung stellte der PPlayButton dar. Für diesen musste ein Toggle implementiert werden, der sowohl das angezeigte Icon wechselt, als auch den kompletten Eintrag abhängig vom Zustand pausiert bzw. abspielt. Auch hierbei war es wie bei allen vorherigen Navigationen wichtig, immer sowohl den angezeigten Text, den diktierten Text als auch das angezeigte Video in der Navigation zu berücksichtigen und synchron zu halten, sodass alle Komponenten gleichzeitig pausiert bzw. fortgesetzt werden.

4.3 Weiterentwickelbarkeit

Zur Gewährleistung der Weiterentwickelbarkeit wurde eine Klasse **HelpSystemBuilder.cs** erstellt, welche als eine Art Erstellungs-Interface dient und vier relevante Methoden bereitstellt, die zur Erstellung eines Eintrages und dessen Integration in das Menü relevant sind:

- `CreateMainMenu(string title, string description, string icon, List<MenuEntry> mainMenuEntries)`

  Dieser Methode ist für die Erstellung des obersten Menüs zuständig, auf welchem bei der weiteren Erstellung aufgebaut wird. Dieses Menü benötigt einen Titel und eine Beschreibung, einen Link auf ein genutztes Icon und eine Liste enthaltener Menüeinträge, bei denen es sich wieder um Blatteinträge oder referenzierende Einträge handeln kann.

- `CreateNewRefEntry(List<MenuEntry> innerEntries, string title, string description, Color entryColor)`

  Diese Methode ist für die Erstellung eines Referenz-Eintrages im NestedMenu, also einem inneren Knoten-Eintrag zuständig. Auch dieser benötigt einen Titel, eine Beschreibung und eine Farbe und zusätzlich eine Liste der in ihm enthaltenen Einträge, seien es nun Blatteinträge oder weitere Referenzeinträge.

- `CreateNewHelpSystemEntry(string title, string description, Color entryColor, string videoPath, LinkedList<LinkedListEntry> keywords, HelpSystemEntry entry = null)`

  Diese Methode ist für die Erstellung eines Hilfesystem-Eintrages im NestedMenu, also eines Blatt-Eintrages zuständig. Hier kann der Titel, die Beschreibung und die Farbe des Eintrages im NestedMenu bestimmt werden. Darüber hinaus muss dann die Liste der abzuspielenden Keywords, also Teileinträge, übergeben werden, genau so wie der Link auf das Video, welches abgespielt werden soll. Zuletzt muss auch das Hilfesystem-GameObject in Unity übergeben werden, damit alle Komponenten auf diesem abgespielt werden können.

- `Execute(HelpSystemEntry helpSystem, string entryTitle, LinkedList<LinkedListEntry> keywords, string videoPath)`

Diese Methode abstrahiert die vollständige Ausführung des Hilfesystem-Eintrages und wird automatisch über die Betätigung eines Blatteintrages aufgerufen. Die übergebenen Parameter wurden bereits mit den vorherigen Methoden mit-übergeben.

All diese Methoden werden in der Klasse **HelpSystemMenu.cs** in einer Methode CreateStartMenu() aufgerufen.
In dieser findet die konkrete Spezifikation des Menüs statt, d.h. die konkreten Ausprägungen der Menüs und Untermenüs, die Erstellung der LinkedListEntries für die Keyword-Übergabe und Videolink-Übergabe sowie die Strukturierung des Menüs.
Da Quellcode innerhalb von Methoden allerdings von oben nach unten ausgewertet wird, ist es wichtig, dass die niedrigste Hierarchiestufe des Menüs, also die untersten Blätter, zuerst definiert werden, da diese Einträge dann den Einträgen eine Ebene höher übergeben werden müssen[11]. Dies zieht sich durch bis zur obersten Ebene des Menüs.

Um nun also das Hilfesystem zu erweitern, benötigt der Entwickler folgende Dinge:

- Das Video im richtigen Format mit dem auf das Video verweisendem Pfad
- Die Methoden der Klasse **HelpSystemBuilder.cs**
- Die Klasse **HelpSystemMenu.cs**, in welcher Methoden des HelpSystemBuilders aufgerufen werden und das konkrete Menü erstellt wird.

Diese Struktur ermöglicht eine vergleichsweise simple Integration weiterer Einträge.

4.4 Zusammenfassung

Das Produkt der Implementierung lässt sich nun folgendermaßen zusammenfassen:
Es wurde ein Menü erstellt, in welchem sich entweder mithilfe einer Hierarchie in Form einer Baumstruktur oder mithilfe einer Fuzzy-Search einzelne Hilfesystemseinträge finden lassen. Darüber hinaus wurde ein Hilfesystem bzw. dessen einzelne Einträge erstellt, welche sowohl eine akustische, eine visuelle, als auch eine textuelle Komponente enthalten, resizable sowie draggable sind und darüber hinaus über eine simple Navigationsbar verfügen. Dieses System ist architekturbedingt für Entwickler ohne umfangreiches Vorwissen erweiterbar und in seiner Struktur auch auf dessen Erweiterung und Ausbau ausgelegt.

[11] Die Konsequenz aus der Nichteinhaltung dieser Konvention wäre ein Verweis auf noch nicht existente Objekte, was zu einer Null-Reference-Exception führt bzw. den Compiler dazu veranlasst, das übergebene Objekt nicht zu erkennen und so Fehler zu produzieren

KAPITEL 5

Evaluation

5.1 Nutzerstudie - Anforderungen

Um die Ausarbeitungen aus der Implementierung sowie die vorangegangenen Fragestellungen dieser Arbeit zu überprüfen, soll im Folgenden eine vergleichende Benutzerstudie durchgeführt werden. Diese soll beantworten, wie gut das implementierte Hilfesystem den Benutzer im Vergleich zu einer herkömmlichen Benutzerdokumentation bei der Interaktion mit SEE unterstützt.

Um diese Resultate messbar zu machen und somit einen Vergleich zu ermöglichen, muss deshalb zunächst ein Fragebogen bzw. eine Evaluationsmethodik bestimmt werden, welche alle eigens formulierten Anforderungen erfüllt. Diese Anforderungen werden deshalb erhoben, damit die Nutzerstudie sowie dessen Auswertung für eine Einzelperson als Teil dieser Arbeit in vertretbarer Zeit umsetzbar bleibt:

- Die Daten, welche im Rahmen der Benutzerstudie ermittelt werden, müssen statistisch verwertbar sein, also Bewertungen über Punkt- oder Rangskalen ermöglichen.
- Für eine Nutzerstudie im Rahmen dieser Arbeit kann nur sinnvoll mit einer kleinen Gruppe an Probanden gearbeitet werden (circa 15-20 Teilnehmer). Das bedeutet, die Aussagekraft der Nutzerstudie muss auch bei kleinen Teilnehmerzahlen statistisch verwertbar sein.
- Die Fragebögen sollen sowohl vorgefertigte, generische Fragen beinhalten, welche zu großen Teilen aus geschlossenen Fragen[1] bzw. Skalenfragen[2] bestehen solle , aber auch Raum für eigen erstellte, präzisere Fragen bieten.
- Es soll sich bei dem Fragebogen vorrangig um einen Post-Study-Fragebogen[3] handeln.

[1] Als geschlossene Frage bezeichnet man eine Frage auf einem Fragebogen, die lediglich mit einem Wort oder Ja oder Nein beantwortet werden kann. Dies ist zumeist dadurch zu begründen, dass sie sehr konkret formuliert sind und keinen Interpretationsspielraum ermöglichen. siehe `https://www.marktforschung.de/wiki-lexikon/marktforschung/Geschlossene%20Fragen/`, zuletzt abgerufen am 04.12.2021

[2] Fragen, welche Antwortmöglichkeiten auf einer Skala von-bis zulassen und somit mehr Möglichkeiten zur Abstufung bieten. Ein Beispiel für eine solche Skala bzw. Skala-Frage ist z.B. die Likert-Skala. siehe `https://wpgs.de/fachtexte/frageboegen/skalierte-fragen/`, zuletzt abgerufen am 04.12.2021

[3] Ein Fragebogen, welcher in Gänze immer erst am Ende der Studie beantwortet wird und nicht, wie z.B. ein Post-Task-Fragebogen, nach jeder Aufgabe

- Der Fragebogen soll in deutscher Sprache verfasst sein oder zumindest über eine validierte, deutsche Übersetzung verfügen. Dies ist darin begründet, dass sprachliche Feinheiten und Nuancen die Auswertung der Evaluation nicht beeinflussen sollen, eine unvalidierte Übersetzung würde jedoch genau in solchen Ungenauigkeiten resultieren.
- Die Evaluation sollte mithilfe von Online-Evaluationstools durchführbar sein. Das bedeutet, dass ein Proband die Studie in Abwesenheit des Entwicklers durchführen können sollte und diesen nur bei Rückfragen kontaktieren muss. Das vereinfacht die Evaluation und sorgt für eine entfernungs- und zeitunabhängige Evaluation, was die Teilnahmeattraktivität an der Studie für die Probanden steigern und somit eine höhere Anzahl an Probanden generieren soll und zusätzlich den Administrationsaufwand während der Evaluation verringert.

Im nachfolgenden Abschnitt werden nun einige Post-Study-Fragebögen vorgestellt und auf ihre Zweckmäßigkeit hin überprüft. Abschließend wird der am Besten passende Fragebogen für die Nutzerstudie ausgewählt.

5.2 Fragebögen und Auswahl

5.2.1 ISONORM 9241/110

Der von Prümper und Anft entwickelte Fragebogen ISONORM 9241/110 ist ein Fragebogen, der Software auf Grundlage der Internationalen Ergonomie-Norm DIN EN ISO 9241-110 beurteilt[Prü]. Innerhalb des Fragebogens soll der Proband einzelne Fragen auf einer Skala von --- bis +++, also insgesamt 7 Stufen, beantworten (siehe Abbildung 5.1).

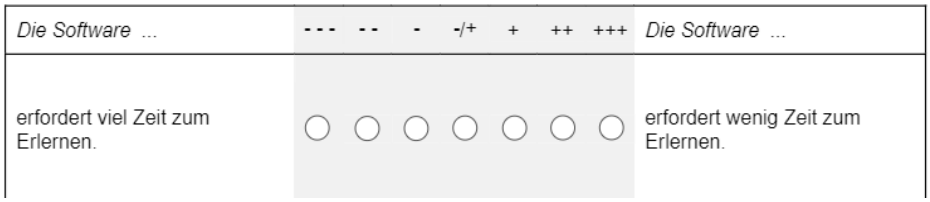


Abbildung 5.1: Eine Frage des ISONORM 9241/110-Fragenbogens

Es existiert eine Kurzform des Fragebogens, die 21 Fragen enthält und darüber hinaus persönliche Fragen zur Auswertung und Einordnung dieser Daten. Erweiternd gibt es jedoch auch eine ausführliche Form des Fragebogens, die 42 Fragen enthält, welche in die Bereiche Aufgabenangemessenheit, Selbstbeschreibungsfähigkeit, Erwartungskonformität, Lernförderlichkeit, Steuerbarkeit, Fehlertoleranz und Individualisierbarkeit unterteilt werden. Der Fragebogen ist in deutscher Sprache verfasst.

Vorteilhaft an diesem Fragebogen ist seine hoher Standardisierungsgrad, der Aufbau der Fragen in Form von Skalenfragen. Die weiteren Anforderungen an den Fragebogen werden ebenfalls von ihm erfüllt. Jedoch kann dieser Fragebogen aus zwei Gründen nicht verwendet werden: Zunächst ist ein Umfang von 21 Fragen immer noch eine zu große Anzahl für eine

verhältnismäßig einfach gehaltene Benutzerstudie, zum Anderen kann über die Lizensierung nach einiger Recherche keine Aussage getroffen werden, weshalb nicht klar ist, ob dieser Fragebogen überhaupt verwendet werden darf oder Einschränkungen unterliegt. Somit wird dieser Fragebogen als Ansatz verworfen.

5.2.2 System Usability Scale (SUS)

Beim System Usability Scale (kurz SUS) handelt es sich um einen der meistgenutzten Fragebögen im Bereich der Usability-Bewertung von Applikationen [Bro96]. Er wurde 1986 von Brooke entwickelt und veröffentlicht. In seiner Originalfassung ist er in der englischen Sprache verfasst, es existieren jedoch einige Übersetzungen auch in die deutsche Sprache, wie z.B. die Übersetzung von SAP [4], welche validiert wurde und somit verwendbar ist. Der Fragebogen besteht aus 10 Fragen, welche alle in Form der Likert-Skala[5] gestaltet sind. Ein Beispiel für eine solche Frage ist in Abbildung 5.2 zu sehen.

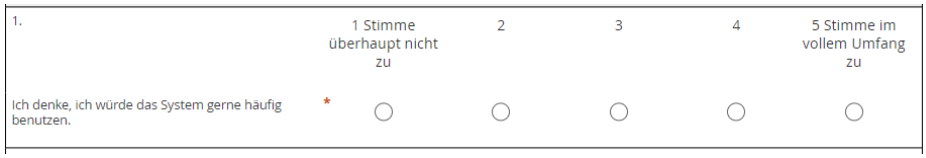


Abbildung 5.2: Die erste Frage des System Usability Scale

Der SUS bietet viele Vorteile:
Er ist leicht zu verstehen und leicht zu erstellen, was sowohl für den Ersteller als auch für den Probanden von Vorteil ist. So wurde er bereits bei der Erstellung von Brooke als „quick and dirty usability scale“ bezeichnet, was in diesem Fall positiv zu verstehen ist, da er keinen Zusatzaufwand erfordert. Diese Aussage impliziert auch den zweiten großen Vorteil dieses Fragebogens. Er ist in seinem Umfang zwar aussagekräftig, jedoch nicht zu komplex, wodurch er ebenfalls schnell zu erstellen und zu bearbeiten ist. Darüber hinaus sind die Werte der Likert-Skala statistisch sinnvoll auswertbar und vergleichbar. Der letzte wesentliche Vorteil ist seine Domänenunabhängigkeit, da die Fragen so generisch sind, dass sie für alle Interface-Anwendungen gültig sind. Dieser Vorteil ist jedoch ebenfalls ein Schwachpuntk des SUS, da er spezifischere Fragen nicht zulässt und lediglich auf diesen vorgefertigten 10 Fragen beruht. Der SUS ist nicht-proprietär und somit zu diesem Verwendungszweck frei nutzbar.

5.2.3 After Scenario Questionaire (ASQ)

Der von Lewis erstellte After Scenario Questionaire, oder auch ASQ, ist ein weiterer Fragebogen, der die Zufriedenheit von Nutzern erfassen soll [Lew91]. Er besteht lediglich aus drei

[4] siehe `https://blogs.sap.com/2016/02/01/system-usability-scale-jetzt-auch-auf-deutsch/`, zuletzt aufgerufen am 04.12.2021

[5] Eine nach dem amerikanischen Psychologen Rensis Likert benannte Skala, welche die Haltung von Probanden zu einem Thema, in unserem Fall einer Software, erfassen soll und hierzu zwischen 5 und 11 Merkmalsausprägungen besitzt, siehe `https://de.statista.com/statistik/lexikon/definition/82/likert_skala/`, zuletzt abgerufen am 04.12.2021

Skala-Fragen auf einer sieben-stufigen Likert-Skala und existiert in validierter Form in der deutschen, der englischen und der französischen Sprache. Durch die Beantwortung der drei Fragen wird dem Durchführer der Studie auf sehr simple Art und Weise ein Eindruck der Probanden zurückgeliefert. Darüber hinaus bezieht sich die letzte Frage des ASQ´s explizit auf Zufriedenheit mit Unterstützung wie z.B. Dokumentationen und Hilfen und wäre somit ideal passend für das Thema dieser Arbeit.

Der große Vorteil des Fragebogens, nämlich seine Einfachheit und Kürze ist auf der anderen Seite der große Nachteil: Eine mit diesem Fragebogen durchgeführte Benutzerstudie basiert auf sehr wenig Werten und vermittelt dem Evaluanten nur einen groben Eindruck über die Resultate. Genauere Ergebnisse können hiermit nicht erzielt werden, da schlicht und einfach weitere Fragen und Informationen fehlen. Eine Benutzerstudie auf lediglich einer Frage beruhen zu lassen liefert zu wenig Informationen zurück. Aus diesem Grund wird der ASQ als Post-Study-Fragebogen ebenfalls verworfen.

5.2.4 Finale Auswahl

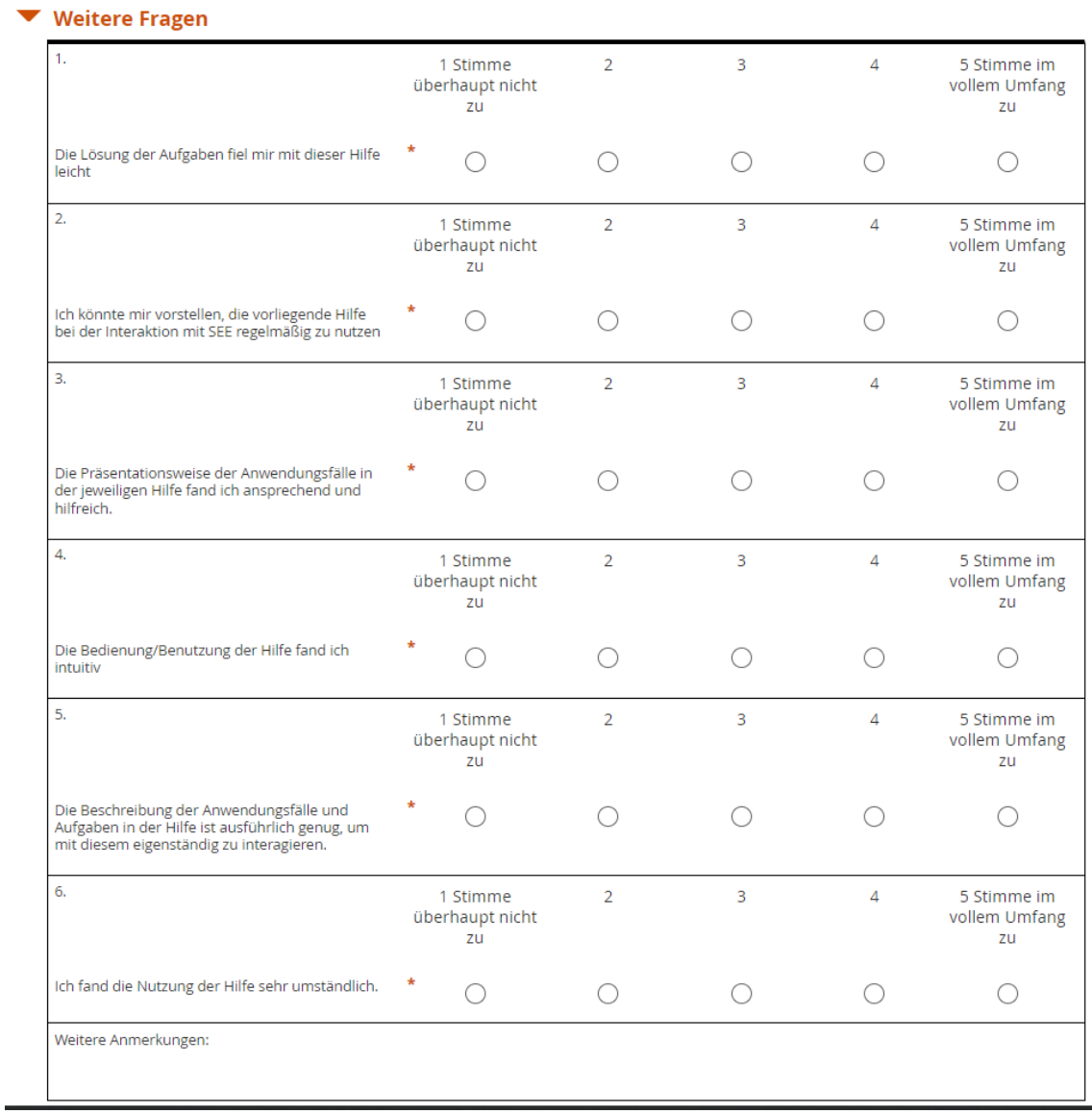
Aus dem Vergleich von ISONORM 9241/110, SUS und ASQ geht hervor, dass der SUS sich am besten zur Durchführung der Evaluation des Hilfesystems eignet. Ergänzend wurden allerdings noch weitere Fragen hinzugefügt, die auch auf einer Likert-Skala-Auswertung beruhen. Diese sind weniger generisch als die Fragen des SUS sind, allerdings selbst entworfen und somit nicht so aussagekräftig und validiert wie die der SUS. Zusätzlich müssen diese getrennt von den Fragen des SUS ausgewertet werden, da diese nicht validiert sind. Die Validierung eines Fragebogens ist wichtig, um seine Aussagekraft und Verlässlichkeit zu gewährleisten. Das bedeutet konkret, dass die Antworten auf die Fragen des SUS wesentlich aussagekräftiger sind als die Antworten auf die eigens formulierten Fragen. Erweiternd sollen diese Ergebnisse jedoch zusätzlich Aufschluss über die Resultate bringen. Vom Aufbau und der Struktur ähneln diese Fragen denen des SUS. Abschließend enthält jeder Fragebogen noch ein Freitextfeld, welches Möglichkeiten zum individuellen Feedback gibt und dem Probanden somit Entfaltungsraum bietet, damit dieser weitere Kommentare abgeben kann.

5.3 Erstellung Evaluationsbogen

Die konkrete Ausprägung der Fragebögen wird im folgenden abgebildet.

		1 Stimme überhaupt nicht zu	2	3	4	5 Stimme im vollem Umfang zu
1. Ich denke, ich würde das System gerne häufig benutzen.	*	○	○	○	○	○
2. Ich fand das System unnötig komplex.	*	○	○	○	○	○
3. Ich fand das System einfach zu benutzen	*	○	○	○	○	○
4. Ich glaube, ich würde die Hilfe einer technisch versierten Person benötigen, um das System benutzen zu können.	*	○	○	○	○	○
5. Ich fand, die verschiedenen Funktionen in diesem System waren gut integriert.	*	○	○	○	○	○
6. Ich denke, das System enthielt zu viele Inkonsistenzen	*	○	○	○	○	○
7. Ich kann mir vorstellen, dass die meisten Menschen den Umgang mit diesem System schnell lernen.	*	○	○	○	○	○
8. Ich fand das System sehr umständlich zu nutzen	*	○	○	○	○	○
9. Ich fühlte mich bei der Benutzung des Systems sehr sicher.	*	○	○	○	○	○
10. Ich musste eine Menge lernen, bevor ich anfangen konnte, das System zu verwenden.	*	○	○	○	○	○

Abbildung 5.3: System Usability Scale



Weitere Fragen

1.	1 Stimme überhaupt nicht zu	2	3	4	5 Stimme im vollem Umfang zu
Die Lösung der Aufgaben fiel mir mit dieser Hilfe leicht *	○	○	○	○	○
2.	1 Stimme überhaupt nicht zu	2	3	4	5 Stimme im vollem Umfang zu
Ich könnte mir vorstellen, die vorliegende Hilfe bei der Interaktion mit SEE regelmäßig zu nutzen *	○	○	○	○	○
3.	1 Stimme überhaupt nicht zu	2	3	4	5 Stimme im vollem Umfang zu
Die Präsentationsweise der Anwendungsfälle in der jeweiligen Hilfe fand ich ansprechend und hilfreich. *	○	○	○	○	○
4.	1 Stimme überhaupt nicht zu	2	3	4	5 Stimme im vollem Umfang zu
Die Bedienung/Benutzung der Hilfe fand ich intuitiv *	○	○	○	○	○
5.	1 Stimme überhaupt nicht zu	2	3	4	5 Stimme im vollem Umfang zu
Die Beschreibung der Anwendungsfälle und Aufgaben in der Hilfe ist ausführlich genug, um mit diesem eigenständig zu interagieren. *	○	○	○	○	○
6.	1 Stimme überhaupt nicht zu	2	3	4	5 Stimme im vollem Umfang zu
Ich fand die Nutzung der Hilfe sehr umständlich. *	○	○	○	○	○
Weitere Anmerkungen:					

Abbildung 5.4: Weitere Fragen

5.4 Durchführung der Nutzerstudie

Für die Durchführung der Evaluation wurde das Online-Evaluationstool KoBo-toolbox[6] genutzt. Die Benutzerstudie ist so aufgebaut, dass dem Nutzer zunächst einige anwendungsbezogene Informationen über SEE vermittelt wurden, bis dieser in eine geführte Evaluation starten konnte. Nachdem einige demografische Fragen und Fragen zur Einschätzung des Erfahrungsschatzes der Benutzer gestellt wurden, musste eine SEE-Executable heruntergeladen werden, welche Online über einen Cloudienst als Downloadlink bereit gestellt wurde. Danach mussten mehrere Aufgaben mit der jeweiligen Hilfe gelöst werden. Nach Abschluss jeder Aufgabe sollte der Nutzer eine Information geben, ob er die Aufgabe lösen konnte oder nicht

[6]siehe: `https://www.kobotoolbox.org/`, zuletzt abgerufen am 04.12.2021

bzw. kleinere, geschlossene Fragen beantworten [7].
Abschließend zu jedem Evaluationsteil sollten die Fragen des SUS und die individuellen Fragen beantwortet werden. Jede Evaluation bestand aus drei Aufgaben, die in ihrer Komplexität unterschiedlich gewählt waren. Im zweiten Schritt sollten nun andere Aufgaben mit der jeweils anderen Hilfe gelöst werden, ansonsten änderte sich an dem Ablauf im Vergleich zum ersten Durchgang nichts. Hier wurden Aufgaben gewählt, die augenscheinlich der Komplexität der zuvor gewählten Aufgaben entsprachen.
Um statistische Probleme zu vermeiden[8], erhielt eine Gruppe zunächst die Software mit dem integrierten Hilfesystem und im zweiten Schritt die Software ohne Hilfesystem mit einer Benutzerdokumentation als .pdf-Datei. Die andere Gruppe erhielt zuerst die Software ohne Hilfesystem mit der Benutzerdokumentation und danach die Software mit dem Hilfesystem. Das Ziel dieser Evaluation war es, mindestens 16 Datensätze pro Hilfe zu erlangen, also insgesamt 32 Datensätze, damit ein statistisch verwertbares Ergebnis erzielt werden kann. Die Evaluation wurde über die Verteilung von Links geregelt und innerhalb von zwei Wochen abgeschlossen. Probleme traten hierbei nahezu keine auf. Dies lässt sich eventuell dadurch begründen, dass die Benutzerstudie zuvor von einigen Testprobanden in einer Pilotstudie getestet wurde. Das Feedback dieser Tests floss danach in die Ausfertigung der eigentlichen Evaluation mit ein. Die Testdaten wurden nicht für die Nutzerstudie verwendet, die Testprobanden waren keine Teilnehmer der Evaluation.

[7] so genannte Post-Task-Fragen, die nach jeder Frage vorkamen. Diese ermöglichen spezifischere Informationen zu jeder Aufgabe

[8] Die Reihenfolge der Bearbeitung kann einen Einfluss auf die Ergebnisse haben, indem der Nutzer z.B. Informationen sammelt, die ihm im zweiten Durchlauf helfen, obwohl dies nicht an dem zu evaluierenden Aspekt liegt

5.5 Auswertung

5.5.1 Demographische Daten

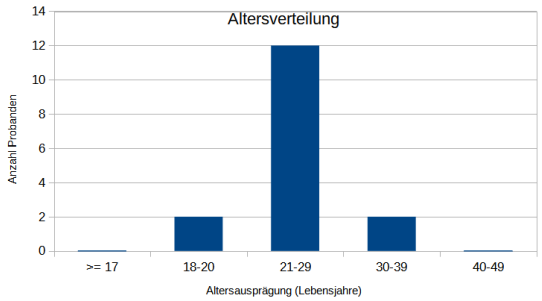


Abbildung 5.5: Altersverteilung

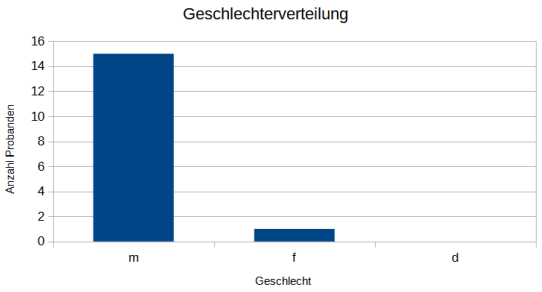


Abbildung 5.6: Geschlechterverteilung

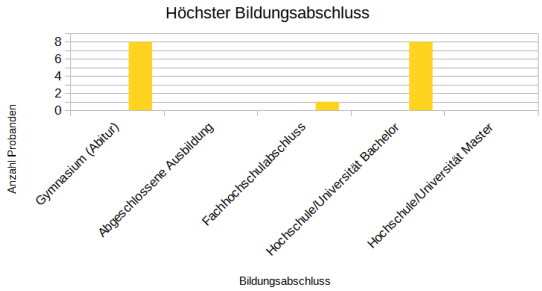


Abbildung 5.7: Bildungsabschluss

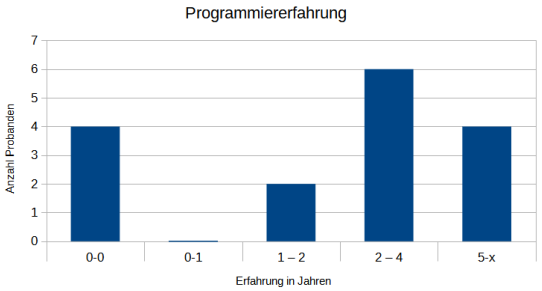


Abbildung 5.8: Programmiererfahrungen

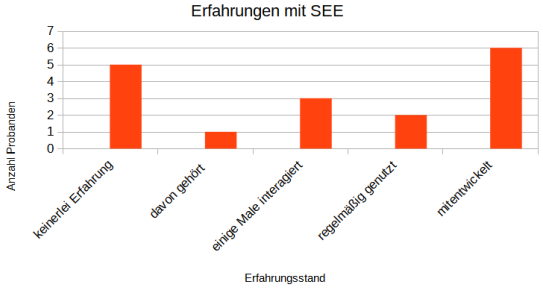


Abbildung 5.9: Erfahrungen mit SEE

Um die statistische Verwertbarkeit der Daten zu gewährleisten, werden zunächst die Antworten auf die demographischen Daten der Teilnehmer ausgewertet. Zur Darstellung dieser Daten wurden Balkendiagramme genutzt. Es wurden Daten zur Altersverteilung, zur Geschlechterverteilung, zum Bildungsabschluss, zur Programmiererfahrung im Allgemeinen und zur Erfahrung mit SEE gesammelt.

Altersverteilung Betrachtet man die Altersverteilung (siehe Abbildung 5.5) der Probanden, fällt schnell auf, dass sich die überwiegende Anzahl dieser - nämlich 75% der Teilnehmer - im Alterbereich von 21-29 Jahren befindet. Weitere vorkommende Gruppen sind die Alters-

gruppen 18-20 (12,5%) und 30-39 (12,5%).

Geschlechterverteilung Die Geschlechterverteilung (siehe Abbildung 5.6) zeigt, dass 15 der 16 Teilnehmer der Studie männlich sind, wohingegen nur eine Person weiblich ist.

Bildungsabschluss Bei der Auswertung des höchsten Bildungsabschlusses der Probanden (siehe Abbildung 5.7) sind zwei Hauptgruppen vorwiegend repräsentiert. Sieben Studienteilnehmer (43,75%) verfügten über einen gymnasialen Abschluss, acht Studienteilnehmer (50%) besaßen sogar einen Bachelorabschluss einer Universität bzw. Hochschule. Ein Teilnehmer (6,25%) hatte als höchsten Bildungsabschluss einen Fachhochschul-Abschluss vorzuweisen. Andere Bildungsabschlüsse waren nicht vertreten.

Programmiererfahrung Bei der Programmiererfahrung (siehe Abbildung 5.8) der Probanden gab es vier auftretende Gruppen: Keine Programmierefahrungen (25%), 1-2 Jahre Programmiererfahrungen (12,5%), 2-4 Jahre Programmiererfahrungen (37,5%) und 5 Jahre Programmiererfahrungen und mehr (25%). Lediglich die Gruppe 0-1 Jahre Programmiererfahrung war nicht vertreten.

Erfahrungen mit SEE Die Erfahrungen mit der Applikation SEE waren bei den Probanden breit gefächert (siehe Abbildung 5.9): Fünf der Sechzehn Teilnehmer (31,25%) hatten keinerlei Erfahrungen mit SEE, ein Teilnehmer (6,25%) hatte lediglich davon gehört, konnte aber keine spezifischeren Informationen vorweisen. Drei Teilnehmer (18,75%) haben in anderen Projekten oder Studien bereits mit SEE interagiert und zwei Probanden (12,5%) haben SEE regelmäßig genutzt bzw. tun dies immer noch. Die größte Anteil der Studienteilnehmer (37,5%) hat bereits an SEE mitentwickelt.

Zusammenfassung Es ist zu erkennen, dass es einige Parameter gibt, bei denen eine relativ geringe Streuung existiert, wie z.B. die Alters- und Geschlechterverteilung und den höchsten Bildungsabschluss, andere Fragen wurden sehr divergierend beantwortet, wie die Programmiererfahrungen und auch die Erfahrungen mit SEE. Inwiefern diese Verteilungen relevant für die Auswertung der Studiendaten ist, wird in den nachfolgenden Kapiteln thematisiert.

5.5.2 Korrektheit

Die Korrektheit der Antworten bzw. die korrekte Lösung der gestellten Aufgaben wurde lediglich über ein Feld mit einer Auswahlbox „Hast du die Aufgabe lösen können?“ überprüft. Tatsächlich handelt es sich bei allen Aufgaben auch um Aufgaben, welche entweder gelöst wurden, oder aber überhaupt nicht gelöst werden konnten und die Resultate somit auch relativ binär waren. Zusätzlich war das Resultat vom Nutzer immer direkt visuell überprüfbar und ersichtlich. Die einzige Frage, welche über eine größere Auswahl an Antworten verfügte, war die Frage für die Aufgabe „Search Node“.

Auf Grundlage der Bewertungen, Kommentare und Feedback existierte bei dieser Frage bzw. bei diesem Anwendungsfall in SEE allerdings ein konzeptionelles Problem, wodurch die Daten aus dieser Frage kaum verwertbar sind, da diese Aufgabe kaum richtig gelöst werden konnte. Auf dieses Problem wird jedoch an dieser Stelle nicht genauer eingegangen, da es nicht durch die Benutzerstudie bzw. die Hilfen verursacht wurde, sondern in einer Funktionsweise von SEE begründet ist.
Die Auswertung der anderen Fragen ergab, dass nur ein Nutzer Probleme bei der Lösung der Aufgaben hatte, diese allerdings sowohl beim Hilfesystem, als auch bei der Benutzerdokumentation. Das lässt sich daran ermitteln, dass nur dieser Proband die Frage, ob er die Aufgaben lösen konnte, mit „Nein“ beantwortete, alle anderen jeweil „Ja“ auswählte. Ansonsten wurden bei beiden Evaluationsteilen gleichermaßen alle Aufgaben gelöst, woraus sich schließen lässt, dass keine der beiden Hilfen größere bzw. kleinere Auswirkungen auf die Korrektheit der Lösung der Aufgaben hat als die andere.

5.5.3 SUS

Um die Daten des System Usability Scale auszuwerten, mussten diese zunächst aus dem Evaluationstool exportiert und tabellarisch aufgelistet werden. Die Resultate dieser Auflistungen sind in Tabelle 5.1 und Tabelle 5.2 zu sehen. Für alle 16 Probanden wurden die Punktwerte der zehn Fragen des SUS abgebildet.

#	q1	q2	q3	q4	q5	q6	q7	q8	q9	q10
p1	4	1	4	3	4	1	4	1	2	2
p2	3	1	4	1	4	1	5	4	5	1
p3	3	2	4	2	3	4	2	3	4	4
p4	5	3	5	3	2	2	3	2	4	2
p5	5	1	5	1	5	1	4	1	5	1
p6	1	5	1	2	1	1	4	5	5	1
p7	1	2	4	2	3	2	4	2	2	2
p8	4	1	5	1	5	1	4	1	5	1
p9	3	1	5	1	3	1	3	4	4	2
p10	3	2	3	1	2	1	4	3	2	3
p11	2	3	1	1	3	3	1	4	2	5
p12	3	3	3	1	2	1	4	4	5	1
p13	3	4	3	2	3	4	3	2	1	2
p14	4	2	3	2	2	3	2	4	3	2
p15	4	1	5	1	4	2	4	1	3	2
p16	1	2	4	1	2	2	4	2	4	2

Tabelle 5.1: Daten Benutzerdokumentation SUS

Die aufgelisteten Resultate des SUS wurden entsprechend der von Lewis [Lew18] beschriebenen Methode nach folgender Formel ausgewertet:

#	q1	q2	q3	q4	q5	q6	q7	q8	q9	q10
p1	4	1	5	1	3	1	4	2	4	1
p2	4	1	5	1	5	1	4	1	5	1
p3	4	1	5	1	5	2	3	2	5	1
p4	2	3	4	1	3	2	4	2	5	1
p5	4	1	5	1	5	1	5	1	5	1
p6	4	4	5	1	5	1	5	2	5	1
p7	2	2	3	4	4	2	3	2	1	2
p8	5	2	3	2	5	1	4	2	3	3
p9	3	1	5	1	3	1	5	2	5	1
p10	2	3	2	2	2	1	4	3	2	3
p11	4	1	5	1	4	3	4	1	4	1
p12	5	1	5	1	5	1	5	1	5	1
p13	3	3	4	1	3	3	3	2	3	2
p14	4	2	4	1	3	1	5	2	4	1
p15	5	1	4	1	4	1	5	1	5	1
p16	2	1	4	1	3	4	4	2	4	2

Tabelle 5.2: Daten Hilfesystem SUS

$$S = 2,5 \cdot (20 + \sum_{i=1}^{10} (-1)^{i+1} \cdot S_i) \tag{5.1}$$

Konkret bedeutet das, dass die Punktwerte aller geraden Fragen invertiert werden. Das ist darin begründet, dass die geraden Fragen negativ formuliert wurden und somit das invertierte Ergebnis zum Vergleich benötigt wird. Aus diesen Werten wird eine Summe gebildet und mit dem Faktor 2,5 multipliziert.
Das Resultat dieser Berechnungen ist eine Zahl zwischen 0 und 100. Das Ergebnis kann nun für sich stehend interpretiert, oder mit einer anderen Auswertung verglichen werden. In diesem Zusammenhang erscheint ein Vergleich zwischen der Auswertung der Resultate des Hilfesystems mit den Resultaten der Benutzerdokumentation sinnvoll. Der Vergleich dieser Werte ist in Tabelle 5.3 dargestellt.

Im nächsten Schritt wurden die ermittelten Scores nun in zwei Violin-Plots[9] zusammengefasst (siehe Abbildung 5.10).

Streuungsparameter Auf Grundlage der präsentierten Daten wurde zunächst die Spannweite der beiden Violin-Plots berechnet. Diese beträgt für die Benutzerdokumentation $R_{Userdoc} = 62,5$ und für das Hilfesystem $R_{HelpSystem} = 50$.

[9]Eine dem Box-Plot ähnliche Darstellung der Verteilung von Ergebnissen, mit dem Unterschied, dass es hierbei weniger um die Quartilsausprägungen, als um einen Gesamteindruck der Verteilung der Resultate geht. Konkret wird die Menge an Ergebnissen in einem Abschnitt durch Punkte dargestellt und durch die Breite des Violin-Plots an dieser Stelle dargestellt. Je breiter der Violin-Plot an einer Stelle ist, desto mehr Ergebnisse werden an dieser abgebildet. Extremwerte bzw. Ausreißer werden in dieser Darstellung durch einen rote Markierung explizit hervorgehoben. [HN98]

	Userdocumentation	HelpSystem	Differenz
p1	75	85	10
p2	82.5	95	12.5
p3	52.5	87.5	35
p4	67.5	72.5	5
p5	97.5	97.5	0
p6	45	87.5	42.5
p7	60	52.5	-7.5
p8	95	75	-20
p9	72.5	87.5	15
p10	60	50	-10
p11	32.5	85	52.5
p12	67.5	100	35.5
p13	47.5	62.5	15
p14	52.5	82.5	30
p15	82.5	95.0	12.5
p16	65	67.5	2.5
Mittelwert	66.125	80.156	14.406

Tabelle 5.3: Vergleich der SUS-Scores

Mithilfe der jeweiligen Mittelwerte $\overline{x}_{Userdoc} = 66,125$ und $\overline{x}_{HelpSystem} = 80,156$ wurden nun die jeweiligen Varianzen mit der Formel $\sigma^2 = \frac{\Sigma(X-\mu)^2}{N}$ berechnet.
Die Varianzen betragen somit $\sigma^2_{Userdoc} = 303,062$ und $\sigma^2_{HelpSystem} = 223,80$.
Um diese besser vergleichen und interpretieren zu können, werden jeweils noch die Standardabweichungen aus diesen berechnet, die $\sigma_{Userdoc} = 17,408$ und $\sigma_{HelpSystem} = 14,96$ betragen.

Aus allen zuvor ermittelten Daten bezüglich der SUS-Scores der Nutzerstudie sind zusammenfassend folgende Erkenntnisse zu ziehen:

- Von den 16 Probanden haben drei Probanden den Umgang mit der Benutzerdokumentation als besser bzw. benutzbarer eingeschätzt.
- Von den 16 Probanden haben 12 Probanden den Umgang mit dem Hilfesystem als besser bzw. benutzbarer eingeschätzt.
- Ein Proband hat beide Systeme gleich effektiv eingeschätzt.
- Der Mittelwert der Einschätzung der Benutzerdokumentation liegt bei 66,125 - der Mittelwert des Hilfesystems hingegen 14,406 Punkte höher bei 80,156.
- Die Betrachtung des Medians unterscheidet sich nicht sonderlich von der des Mittelwertes, eher verstärkt sie die zuvor ermittelten Tendenzen noch: Die Einschätzung der Benutzerdokumentation beträgt 66,25 Punkte, die des Hilfesystems 85,00 Punkte, was einen Unterschied von 18,75 Punkten ausmacht.
- Die Varianz der SUS-Scores ist mit 303,062 Punkten bei der Benutzerdokumentation höher als die bei dem Hilfesystem (223,80). Daraus lässt sich schließen, dass die Einigkeit

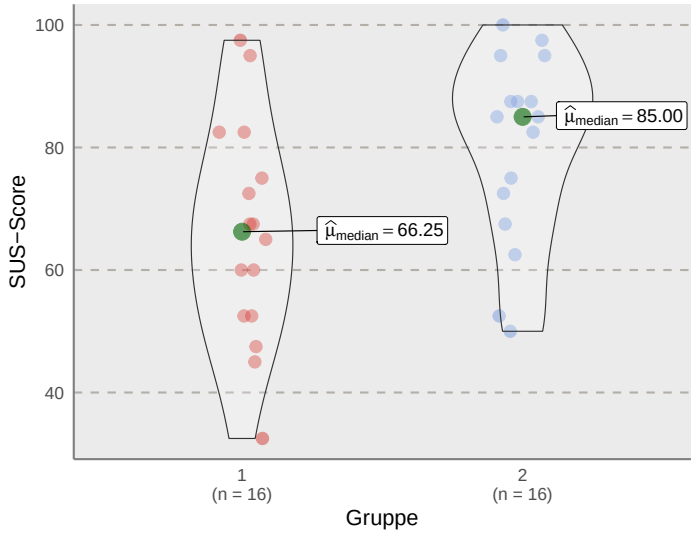


Abbildung 5.10: SUS-Scores

der Probanden bei der Bewertung des Hilfesystems größer war als bei der Bewertung der Benutzerdokumentation – es gab also ein geringeres Streufeld bei der Bewertung.

- Diese Annahme lässt sich durch Hinzunahme der Standardabweichung bestätigen, Die Bewertung der Nutzerdokumentation weicht durchschnittlich um 17,408 Punkte vom Mittelwert ab, die des Hilfesystems nur um 14,96 Punkte. Trotzdem deuten beide Werte auf eine große Streuung hin.

Um abschließend auch die statistische Signifikanz der Ergebnisse nachzuweisen, muss ein statistischer Test verwendet werden. Zur Auswahl des Testverfahrens müssen zunächst einige Grundannahmen über die Evaluation getroffen werden.
Zunächst wird festgestellt, dass es sich um einen nicht-parametrischen Test handeln muss, da es sich bei der Testgruppe nicht um eine normalverteilte Population handelt (siehe Kapitel 5.5.1).
Darüber hinaus wird festgestellt, dass die Stichproben bzw. Evaluationen des Hilfesystems und der Benutzerdokumentation voneinander unabhängig sind. Ein weiterer wichtiger Aspekt der Benutzerstudie ist es, dass die Probanden der verschiedenen Testgruppen aus sehr ähnlichen bzw. gleichen Populationen stammen.
Nimmt man diese Erkenntnisse zusammen und recherchiert dahingehend nach statistischen Testverfahren wird schnell ersichtlich, dass sich der „Wilcoxon-Mann-Whitney-Test“, oder auch „Mann-Whitney-U-Test“ bzw. „U-Test“ am besten zum Nachweis der statistischen Signifikanz eignet[Tea21]. Das grundlegende Konzept dieses statistischen Testverfahrens ist es, die Differenzen zwischen einzelnen Ergebnissen zu vergleichen.
Zunächst wird eine Nullhypothese H_0 aufgestellt, welche durch den Test bewiesen bzw. verworfen werden soll. Dementgegen wird auch die Alternativhypothese H_1 aufgestellt, welche alle anderen Resultate umfasst:

- Nullhypothese H_0: Die Benutzerzufriedenheit und Usability von SEE mit dem Hilfesys-

tem ist kleiner als oder gleich der mit der Benutzerdokumentation, sodass gilt : $U_{HelpSystem} \leq U_{UserDoc}$

- Alternativhypothese H_1: Die Benutzerzufriedenheit und Usability von SEE mit dem Hilfesystem ist größer als die mit der Benutzerdokumentation, sodass gilt: $U_{HelpSystem} > U_{UserDoc}$

Nun muss ein Signifikanzniveau für das Resultat bestimmt werden. Hierfür wird beim Mann-Whitney-U-Test in der Softwareforschung standardmäßig ein Niveau von $\alpha = 0,05$ gewählt.

Die Durchführung dieses statistischen Tests anhand der SUS-Scores beider Testgruppen ergab als Resultat $U = 67,5$ und $p = 0,01171$. Dieser Wert lässt sich so interpretieren, dass es einen signifikanten Unterschied zwischen der Usability von SEE mit dem HelpSystem im Vergleich zur Benutzerdokumentation in Richtung des HelpSystems gibt. Das resultiert daraus, dass der p-Wert der Berechnung kleiner ist als unser festgelegtes Signifikanzniveau von $\alpha = 0,05$. Somit muss die Nullhypothese H_0 verworfen werden und die Alternativhypothese H_1 wird angenommen.

5.5.4 Eigene Fragen

Ein konzeptionell ähnliches Vorgehen wie bei dem System-Usability-Scale wurde bei den sechs eigen formulierten Fragen (siehe Abbildung 5.4) angewandt. Zunächst wurden auch diese Resultate tabellarisch in Tabelle 5.4 und 5.5 festgehalten.

#	q1	q2	q3	q4	q5	q6
p1	5	4	4	5	5	1
p2	5	4	4	4	5	3
p3	4	1	4	2	4	4
p4	4	1	4	2	4	4
p5	4	5	5	5	5	1
p6	1	1	1	1	3	5
p7	4	5	4	4	4	1
p8	5	3	3	5	5	1
p9	5	5	5	5	5	1
p10	5	2	4	4	5	1
p11	3	3	4	3	3	3
p12	3	2	3	2	5	4
p13	4	2	5	5	5	1
p14	4	3	3	2	4	4
p15	4	4	3	3	4	1
p16	4	3	2	3	2	2

Tabelle 5.4: Daten Benutzerdokumentation

#	q1	q2	q3	q4	q5	q6
p1	5	4	5	3	5	2
p2	5	5	5	4	4	1
p3	5	5	4	5	5	1
p4	5	5	4	5	5	1
p5	4	5	5	4	5	1
p6	5	5	3	5	5	1
p7	4	4	4	4	4	1
p8	5	5	5	5	5	1
p9	5	5	5	5	5	1
p10	4	5	5	4	4	1
p11	5	4	5	4	5	1
p12	5	5	5	5	5	1
p13	4	3	3	2	4	2
p14	5	3	4	4	4	2
p15	5	5	5	5	5	1
p16	5	1	4	4	3	2

Tabelle 5.5: Daten Hilfesystem

Im nächsten Schritt musste die Formel zur Berechnung des SUS angepasst werden: Die eigenen Fragen beinhalten lediglich eine negativ formulierte Frage (Frage Nr. 6), sodass nur diese Frage invertiert werden musste und für alle anderen Fragen eine ähnliche Formel wie die des SUS gilt. Final muss das Ergebnis nun nicht mit 2,5 mulitpliziert werden, sondern mit $(2,5)^2$, damit ebenfalls eine Skala von 0-100 erzielt wird und die Werte somit sowohl untereinander, als auch zum SUS vergleichbar sind. Die Ergebnisse dieser Berechnungen sind in Tabelle 5.6 dargestellt.

	Userdocumentation	HelpSystem	Differenz
p1	87.5	87.5	0
p2	93.75	100	6.25
p3	68.75	93.75	25
p4	68.75	93.75	25
p5	93.75	87.5	-6.25
p6	37.5	87.5	40.0
p7	87.5	81.25	-6.25
p8	75	100	25
p9	100	100	0
p10	68.75	93.75	25
p11	81.25	87.5	6.25
p12	56.25	100	43.75
p13	75	62.5	-12.5
p14	75	87.5	12.5
p15	68.75	100	31.25
p16	75	81.25	6.25
Mittelwert	75.781	88.36	13.828

Tabelle 5.6: Vergleich der Eigene-Fragen-Scores

Aus diesen Daten wurde wie zuvor für die SUS-Scores ein Violin-Plot erstellt, der die Streuung der Ergebnisse visualisiert (siehe Abbildung 5.11).

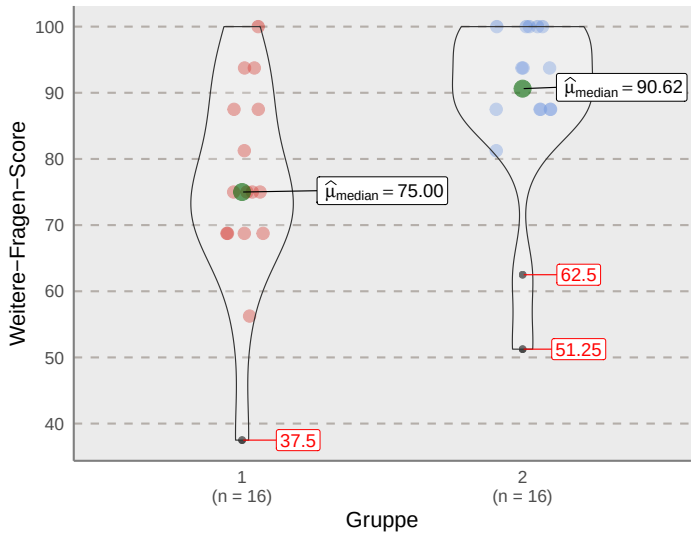


Abbildung 5.11: Eigene-Fragen-Scores

Streuungsparameter An dieser Stelle wird auf eine erneute Beschreibung der Berechnung von Spannweite, Varianz und Standardabweichung verzichtet, da das Vorgehen bereits bei den SUS-Scores beschrieben wurde. Stattdessen handelt es sich hierbei um eine bloße Auflistung der Streuungsmaße.

- $R_{Userdoc} = 62,5$; $R_{HelpSystem} = 37,5$.
- $\sigma^2_{Userdoc} = 224$; $\sigma^2_{HelpSystem} = 92,62$.
- $\sigma_{Userdoc} = 14,97$; $\sigma_{HelpSystem} = 9,62$

Auch diese Daten lassen sich nun auf folgende Ergebnisse reduzieren:

- Von den 16 Probanden haben drei Probanden den Umgang mit der Benutzerdokumentation als besser bzw. benutzbarer eingeschätzt.
- Von den 16 Probanden haben 11 Probanden den Umgang mit dem Hilfesystem als besser bzw. benutzbarer eingeschätzt.
- Zwei Probanden haben keines der Systeme effektiver eingeschätzt.
- Der Mittelwert der Einschätzung der Benutzerdokumentation liegt bei 75,781 - der Mittelwert des Hilfesystems hingegen 13,828 Punkte höher bei 88,36.
- Der Median der Einschätzung der Benutzerdokumentation unterscheidet sich nicht wesentlich von dem des Mittelwertes, bei der Benutzerdokumentation liegt er bei 75,00, bei dem Hilfesystem bei 90,62. Das bedeutet eine bessere Bewertung des Hilfesystems um 16,12 Punkte.
- Die Spannweite der Daten mit der Userdokumentation liegt mit 62,5 deutlich über der des Hilfesystems (37,5). Das bedeutet, die Extremwerte der Bewertung des Hilfesystems liegen weniger weit auseinander.
- Die Varianz der Daten bestätigt diesen Trend – die Varianz der Daten der Userdocumentation (224) ist mehr als doppelt so groß wie die des HelpSystems (92,62).
- Am deutlichesten wird dieser Trend anhand der Standardabweichung: Ein Studienteilnehmer der Benutzerdokumentation weicht durchschnittlich um 14,97 Punkte vom Mittelwert ab, bei dem Hilfesystem hingegen nur um 9,62 Punkte. Somit herrschte bei der Bewertung des Hilfesystems mehr Einigkeit als bei der Bewertung der Benutzerdokumentation.

Damit sind die Statistiken dieser Fragen zwar ein wenig positiver (also näher am Maximum 100) als die des SUS, die Relationen zum SUS zwischen Hilfesystem, Benutzerdokumentation und dessen Einschätzung sind allerdings sehr ähnlich. Dasselbe gilt für die Tendenzen, welche die Streuungsmaße aufzeigen:

So stehen Varianzen von 303,062 (bzw. eigene Fragen: 224) bei der Benutzerdokumentation Varianzen von 223,8 (bzw. eigene Fragen: 92,62) beim Hilfesystem und Standardabweichungen von 17,408 (bzw. 14,97) bei der Benutzerdokumentation und 14,96 (bzw. 9,62) beim Hilfesystem gegenüber. Auch wenn die Werte sich an einigen Stellen unterscheiden, unterstützen sie alle dieselbe These, welche im Folgenden statistisch überprüft werden soll.

Abschließend bedürfen auch diese Daten einer statistischen Bewertung. Genau wie in Kapitel 5.5.3 bedienen wir uns hierfür den Mann-Whitney-U-Test mit einem Signifikanzniveau von $\alpha = 0,05$ und übernehmen ebenfalls die Nullhypothese H_0 und die Alternativhypothese H_1.
Die Auswertung der Daten ergab in diesem Fall die Werte $U = 53,5$ und $p = 0,00237$.
Dieses Ergebnis unterstützt die zuvor ermitellten Resultate und bestätigt dieses auch:

Es existiert ein signifikanter Unterschied zwischen der Usability von SEE mit dem HelpSystem im Vergleich zur Benutzerdokumentation zu Seiten des HelpSystems. Die Nullhypothese H_0 kann auch hier verworfen werden und die Alternativhypothese H_1 wird angenommen.

5.6 Interpretation und Diskussion

In diesem Abschnitt sollen Schlussfolgerungen aus den Auswertungen gezogen, Daten kritisch hinterfragt und Annahmen für die Grundgesamtheit der Softwareentwickler getroffen werden. Besonders auffällig bei der Auswertung der Daten waren die großen Streuungsmaße der Bewertungen auch innerhalb der Testgruppen. Schaut man sich die demographischen Daten der Teilnehmer an fällt jedoch auf, dass hier keine offensichtliche Begründung für diese unterschiedlichen Bewertungen ersichtlich sind. Es ist also keine Korrelation zwischen der Bewertung der Usability der Hilfen und der Demographie zu erkennen, woraus sich schließen lässt, dass andere Faktoren Einfluss auf diese Bewertung nehmen, welche nicht evaluiert wurden.
Darüber hinaus wurde allerdings festgestellt, dass eine interaktive Benutzerhilfe insgesamt besser bewertet wurde als die Benutzerdokumentation.
Um nun Rückschlüsse auf die Grundgesamtheit der Softwareentwickler zu ziehen und zu sehen, ob diese Daten auch auf die Gesellschaft bzw. Grundgesamtheit der Softwareentwickler anwendbar ist, müssen zunächst Daten zu Softwarentwicklern recherchiert werden. Softwareentwickler sind im Durchschnitt 40,2 Jahre alt bzw. 38,5 Jahre alt, wenn sie lediglich im Bereich der Informatik arbeiten[10]. Das bedeutet, dass es hier eine Diskrepanz zwischen den erhobenen Daten (Durchschnittsalter zwischen 20-29 Jahren) und der realen Welt gibt. Auch bezüglich des Anteils an Frauen bestehen Unterschiede, so sind im Rahmen dieser Benutzerstudie lediglich 6,25% der Probanden weiblich gewesen, in der realen Welt beträgt ihr Anteil in IT-Berufen jedoch circa 17%[11]. Das lässt darauf schließen, dass die vorangegangenen Un-

[10] siehe `https://www.destatis.de/DE/Presse/Pressemitteilungen/2018/11/PD18_448_122.html`[zuletzt abgerufen am 04.01.2022]
[11] siehe: `https://www.bitkom-research.de/de/pressemitteilung/it-fachkraefte-nur-jeder-siebte-bewerber-ist-weiblich`[zuletzt abgerufen am 04.01.2022]

tersuchungen für junge, männliche Softwareentwickler validiert angenommen werden können, für andere Nutzergruppen müssen allerdings weitere Studien durchgeführt werden, die im Rahmen dieser Arbeit nicht umsetzbar gewesen sind.

5.7 Threats to Validity

Dieser Abschnitt soll Aspekte und Rahmenbedingungen thematisieren, welche einen Einfluss auf die Resultate der Studie haben könnten und somit verfälschend wirken würden. Es handelt sich hierbei lediglich um Spekulationen und Gedankenanstöße. Darüber hinaus werden nicht alle aus der Literatur bekannten Aspekte [Zho+16] aufgelistet, sondern lediglich diese, bei denen ein möglicher Einfluss angenommen wird. Auch auf eine explizite Unterscheidung zwischen internen und externen Threats wird an dieser Stelle verzichtet.

Testing/Lerneffekte Zur Vermeidung von Lerneffekten wurde jeder Proband in der Studie nur für einen Durchlauf mit dem Hilfesystem und einem Durchlauf mit der Benutzerdokumentation verwendet. Um einen überhöhten Einfluss der Reihenfolge der Durchführung auszuschließen, erhielt die Hälfte aller Probanden zunächst das Hilfesystem, die andere Hälfte zunächst die Benutzerdokumentation. Trotz dessen ist es nicht auszuschließen, dass eine der beiden Hilfen einen größeren Lerneffekt auslöst und somit den zweiten Durchlauf mit der anderen Hilfe stärker beeinflusst.

Experimentatoreinfluss Da es sich bei SEE um ein sehr spezifisches Fachgebiet handelt und viele Probanden sich untereinander kennen und auch den Durchführer der Studie, ist ein Einfluss dahingehend möglich, dass Probanden die Studie bzw. das Hilfesystem bewusst oder unbewusst besser bewerten als es sie es normalerweise tun würden. Der mögliche Einfluss dieses Threats ließ sich nicht völlig entfernen, wurde jedoch durch verschiedene Versuche, wie z.B. einige Probanden, zu denen kein direkter Kontakt herrscht, reduziert.

Repräsentanz/Demographie der Probanden Bei der Demographie der Teilnehmer fällt schnell auf, dass sowohl die Gruppe der weiblichen Probanden, als auch ältere Probanden unterrepräsentiert sind und somit nicht stellvertretend für die Gesellschaft bzw. die Gesamtheit an möglichen Nutzern von See sind. Es lässt sich lediglich festhalten, dass der Bereich junger Softwareentwickler o.Ä. insgesamt ein Hilfesystem einer Benutzerdokumentation vorzieht. Eine weitere interessante Frage, welche jedoch im Rahmen dieser Arbeit nicht beantwortet wird, ist, ob ältere oder erfahrenere Softwareentwickler, die immer mit Benutzerdokumentationen gearbeitet haben, sich dieser Meinung anschließen würden, oder ob es andere Resultate in anderen Alters- bzw. Erfahrungsgruppen gäbe.

Auswahl der Aufgaben Da es sich um eine geführte Benutzerstudie mit gestellten Aufgaben handelte, kann es sein, dass einzelne Aufgaben den Nutzer vor größere Probleme stellten

als andere, unabhängig von der vorliegenden Benutzerhilfe. Darüber hinaus ist es möglich, dass die Nützlichkeit der Hilfe von der Komplexität der gestellten Aufgabe abhängt. So kann z.B. ein Video komplexe Anwendungsfälle besser darstellen und erklären als eine Benutzerdokumentation, oder ein einfacherer Anwendungsfall wird durch die Benutzerdokumentation prägnanter und kürzer beschrieben als durch das Hilfesystem. Diese Aspekte wurden bei der Auswahl der Aufgaben allerdings berücksichtigt, sodass für beide Aufgaben jeweils komplexe und nicht so komplexe Anwendungsfälle genutzt wurden und den Benutzer somit in einem breiten Gebiet testen ließen. Ein Einfluss hierdurch lässt sich jedoch trotzdem nicht völlig ausschließen, da die Komplexität einzelner Anwendungsfälle sehr subjektiv wahrgenommen werden kann.

5.8 Zusammenfassung

Die Durchführung der Nutzerstudie ergab somit, dass sowohl die Resultate des SUS als auch die der eigenen Fragen den Umgang mit dem Hilfesystem signifikant besser bewertet wurden als die Resultate mit der Benutzerdokumentation. Hierbei gilt allerdings zu beachten, dass dies nicht generell für die Gesellschaft bzw. Gesamtheit der Softwareentwickler gelten muss, da die Daten nicht für die Gesamtheit repräsentativ sein müssen. Bei den erhobenen Daten wurden zusätzlich hohe Streuungsmaße berechnet, was auf eine große Uneinigkeit der Probanden in der Bewertung schließen lässt. Diese Uneinigkeit war sowohl bei den SUS-Scores, als auch bei den Eigene-Fragen-Scores in der Bewertung der Benutzerdokumentation höher als in der Bewertung des Hilfesystems. Darüber hinaus gilt es noch zu erwähnen, dass einzelne Probanden die Benutzerdokumentation im relevanten Maße als besser als das Hilfesystem bewertet haben, was für ein Gesamtfazit ebenfalls berücksichtigt werden sollte.

KAPITEL 6

Fazit

Das Ziel dieser Arbeit war es, zu überprüfen, ob es Alternativen zum Einsatz einer Benutzerdokumentation bei einer Software wie SEE geben kann. Dafür wurde als Vergleichsobjekt ein Hilfesystem konzipiert und implementiert. Das Design des Hilfesystems genau wie das der Benutzerdokumentation orientiert sich hierbei an gängigen Praxen und Ständen der Forschung in Bezug auf Dokumentationsdesign und Lernverhalten von Menschen. Nachdem diese Implementierung mit der eigens erstellten Benutzerdokumentation in einer vergleichenden Benutzerstudie verglichen wurde, konnten hieraus mithilfe des System Usability Scale und nach demselben Konzept eigens entworfener Fragen Daten erhoben werden. Diese Daten wurden daraufhin ausgewertet und statistisch anhand des Mann-Whitney-U-Tests evaluiert. Das Resultat dieser Evaluation war, dass das Hilfesystem sowohl anhand des SUS, als auch anhand der eigenen Fragen signifikant besser bewertet wurde. Hieraus lässt sich schließen, dass Nutzer von SEE den Umgang mit der Software lieber und besser anhand des Hilfesystems erlernen würden als anhand einer Benutzerdokumentation. Jedoch gilt es auch zu berücksichtigen, dass einzelne Probanden sich explizit gegen dieses Resultat aussprachen und gegenteilig bewertet haben. Darüber hinaus wurde festgestellt, dass eine Übertragung der Ergebnisse auf die gesamte Gesellschaft nicht zwangsläufig möglich ist, da die Testgruppe für diese nicht vollständig repräsentativ ist.

Der Übersicht halber werden die zu Beginn der Arbeit aufgestellten Leitfragen an dieser Stelle erneut wiederholt, bevor diese beantwortet werden:

- 1) Wie wirkt sich die Integration eines Hilfe-Systems in Bezug auf die Usability von SEE im Vergleich zu einem herkömmlichen Benutzerhandbuch aus?
- 2) Wie effizient ist die Einarbeitung in die Applikation mithilfe des Hilfesystems im Vergleich zu einem Benutzerhandbuch?
- 3) Kann ein Hilfesystem eine Alternative zu einem Benutzerhandbuch bieten bzw. dieses vollumfänglich ersetzen?

Abschließend können diese Leitfragen wie folgt beantwortet werden:

- 1) Die Integration des Hilfesystems wirkt sich im Vergleich zur Nutzung einer Benutzerdokumentation positiv auf die Benutzbarkeit der Anwendung SEE aus.

- 2) Über die Effizienz der Einarbeitung lässt sich anhand der ermittelten Daten nur eine eingeschränkte Antwort geben, jedoch scheint sich diese durch zusätzlich gegebenes Feedback und die erfolgreiche Lösung der Evaluationsaufgaben bis auf wenige Ausnahmen nicht wesentlich zu unterscheiden.
- 3) Auf Grundlage aller ermittelten Daten kann ein Hilfesystem für SEE eine sinnvolle Alternative zu einer Benutzerdokumentation sein und dieses vollständig ersetzen. Allerdings sollte dies zunächst durch weitere Feldversuche mit breiteren Nutzergruppen validiert werden, um die Repräsentativität der in dieser Arbeit ermittelten Daten für die Gesamtheit der Nutzer zu überprüfen. Ein paralleler Einsatz beider Systeme in Kombination mit einem steigen Wandel mag also durchaus sinnvoll sein.

Somit schließt die Arbeit mit einem Zitat von dem deutschen CSU-Politiker Roman Herzog(1934-2017), welches mein persönliches Fazit dieser Arbeit sehr gut widerspiegelt:

> „Wir brauchen nicht alles Bewährte über Bord zu werfen. Aber Erneuerung tut Not, schon um das Bewährte für die Zukunft zu sichern."

KAPITEL 7

Ausblick

Um einen kurzen Einblick in künftige Möglichkeiten mit diesem System zu bieten, wird in diesem letzten Kapitel kurz auf verschiedene Aspekte eingegangen, die im Entwicklungsprozess dieser Arbeit präsent wurden.

7.1 Integration

Zunächst müsste das System in die Hauptanwendung integriert werden. Hierzu muss zunächst das Menü erweitert werden, woraufhin die einzelnen Hilfesystemeinträge inklusive textueller Beschreibung und Videos ergänzt werden müssen, die bisher nur exemplarisch umgesetzt sind. Dies ist zwar durch die erstellte Struktur nicht sonderlich komplex, jedoch mit einigem Aufwand verbunden. Für diesen Integrationsprozess müssen also entsprechende Ressourcen an Arbeitskraft freigeräumt werden. Darüber hinaus ist das Hilfesystem mit Wartungsprozessen verbunden, d.h. jeder Anwendungsfall, der Änderungen im Interaktionsablauf unterzogen wird, muss im Hilfesystem aktualisiert werden. Für diesen Prozess wäre eine entsprechende Prozessorganisation sinnvoll, wie z.B. eine direkte Anpassung des Systems vor der Integration des Features, damit die Hilfe nicht obsolet bzw. veraltet wird. Sollte dieser Prozess allerdings sinnvoll integriert werden, kann das Hilfesystem wie bereits zuvor evaluiert eine sinnvolle Ergänzung für die Usability und das Erlernen von SEE bieten.

7.2 Anmerkungen

Im Zuge der Evaluation wurden einige Verbesserungsvorschläge gemacht, welche zumindest vor dem Integrationsprozess bedacht und bewertet werden sollten. Diese werden im Folgenden kurz vorgestellt.

Menü-Öffnung Es wurde angemerkt, dass die Bindung des Hilfesystems an ein Objekt innerhalb der Anwendung nur begrenzt sinnvoll sein kann. Ein Beispiel hierfür wäre, dass ein Nutzer gerade fokussiert innerhalb einer Stadt auf der Suche nach bestimmten Knoten ist und hierfür eine Unterstützung wünscht. Dann muss er den Fokus wechseln, eventuell wieder herauszoomen, um die Person anzuklicken, über welche sich das System öffnen lässt,

er wird also von seinem laufenden Prozess abgelenkt. Eine alternative Möglichkeit zu der Bindung an ein Objekt wäre es, das Hilfesystem entweder an eine Taste oder einen Shortcut auf der Tastatur zu binden, oder es auf der Benutzeroberfläche des Nutzers über ein Symbol permanent anzeigen zu lassen, wie z.B. ein Fragezeichen am rechten oberen Bildschirmrand. Dieser Vorschlag mag die Nutzbarkeit des Systems noch erhöhen, sollte allerdings in der Praxis getestet werden.

Menü-Struktur Eine weitere Anmerkung am System war es, dass die Menüführung für den ungeübten Benutzer nicht zwangsläufig intuitiv war. Die Gruppierung/Strukturierung der Elemente sollte also noch einmal überdacht werden und eventuell neu angeordnet werden, um auch neuen Nutzern den Einstieg zu erleichtern. Auch hierauf ist das Hilfesystem aber bereits ausgelegt und sollte keine gravierenden Probleme verursachen. Ein Alternative wäre auch, einen größeren Fokus auf das Suchfeld zu legen und es eventuell durch Umrahmung o.Ä. mehr hervorzuheben, da dieses für einige Probanden in der Benutzerstudie unterging.

Erweiterung der Keyword-Suche Das zuvor erwähnte Suchfeld könnte eventuell über eine Liste an Keywords erweitert werden, anstatt lediglich nach dem Titel des Eintrages zu suchen. Ein Beispiel wäre eine Liste an Worten wie „Add Node, New Node, Create Node“ für den Add-Node-Prozess, um ein breiteres Suchfeld zu ermöglichen. Auch diese Liste könnte dann über eine Fuzzy-Search gesucht werden und auf den Add-Node-Eintrag verweisen. Dieser Vorschlag ist allerdings bisher nicht in der Architektur berücksichtigt und würde einigen Integrationsaufwand vorrausetzen, dafür aber auch die Usablity erhöhen. Dennoch ist es eventuell eher ein Thema für einen späteren Verbesserungsprozess, sollte sich dieses Problem im laufenden Betrieb als substantiell herausstellen.

7.3 Weitere Entwicklungsmöglichkeiten

Sollte das Grundkonzept eines Hilfesystems sich in der Praxis als geeignet herausstellen, die Integration der Videos aber Probleme bzw. zu viel Aufwand verursachen, da sich Anwendungsfälle zu oft ändern, wäre eine Umstellung auf eine ablaufende Beispielszene wie im Kapitel 3.2.1 kurz beschrieben, eine mögliche Alternative. Dies würde den Aktualisierungsprozess verkürzen, jedoch müsste hierfür die Architektur des Systems umgebaut und eine Beispielszene erstellt werden, sodass nur noch das Grundkonzept und Design erhalten bliebe. Auch diese Option sollte zunächst im praktischen Einsatz überprüft werden und kann im weiteren Entwicklungsprozess ersetzt optimiert bzw. ersetzt werden, sollte man dies als sinnvoll erachten.

ABBILDUNGSVERZEICHNIS

Literaturverzeichnis

[21] *Software Engineering Experience*. [Online; zuletzt aufgerufen: 08-May-2021]. 2021. URL: https://see.uni-bremen.de/.

[Art93] W. Brian Arthur. *On the Evolution of Complexity*. Working Papers 93-11-070. Santa Fe Institute, Nov. 1993. URL: https://ideas.repec.org/p/wop/safiwp/93-11-070.html.

[Bah20] Ines Bahr. *Softwareinvestitionen kritisch für das Überleben des Mittelstands nach der Coronakrise*. Ed. by ap-verlag. [Online; zuletzt aufgerufen: 08-May-2021]. June 2020. URL: https://ap-verlag.de/softwareinvestitionen-kritisch-fuer-das-ueberleben-des-mittelstands-nach-der-coronakrise/61002/.

[Ble+21] Moritz Blecker et al. *Projektbericht SEE*. 2021.

[Bro96] John Brooke. "Sus: a "quick and dirty'usability". In: *Usability evaluation in industry* 189 (1996).

[BRW11] Alexandra Bergedick, Dirk Rohr, and Anja Wegener. *Bilden mit Bildern: Visualisieren in der Weiterbildung*. wbv Publikation, 2011.

[Gee21] Richard Geerligs. *Wie viele Softwareentwickler gibt es in Deutschland, den USA und auf der Welt?* [Online; zuletzt aufgerufen: 21-November-2021]. Sept. 2021. URL: https://www.daxx.com/de/blog/entwicklungstrends/anzahl-an-softwareentwicklern-deutschland-weltweit-usa.

[Hai18] Johannes Hain. "Einfache statistische Testverfahren". In: *Lehrstuhl für Mathematik VIII-Statistik* (2018).

[HN98] Jerry L Hintze and Ray D Nelson. "Violin plots: a box plot-density trace synergism". In: *The American Statistician* 52.2 (1998), pp. 181–184.

[JDP20] Florian Jung, Veronika Dashuber, and Michael Philippsen. "Towards Collaborative and Dynamic Software Visualization in VR". In: *Proceedings of the International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 3: IVAPP*. INSTICC. SciTePress, 2020, pp. 149–156. ISBN: 978-989-758-402-2. DOI: 10.5220/0008945201490156.

[Kos20] Rainer Koschke. "Auge in Auge mit Ihrer Softwarearchitektur". In: (2020), pp. 5–10.

[Lew18] James R Lewis. "The system usability scale: past, present, and future". In: *International Journal of Human–Computer Interaction* 34.7 (2018), pp. 577–590.

[Lew91] James R. Lewis. *Psychometric evaluation of an after-scenario questionnaire for computer usability studies: the ASQ?* [Online; zuletzt aufgerufen: 21-November-2021]. Jan. 1991. URL: https://dl.acm.org/doi/abs/10.1145/122672.122692.

[Mic21] Michsky. *Modern UI Pack.* [Online; zuletzt aufgerufen: 21-November-2021]. 2021. URL: https://www.michsky.com/project/modern-ui-pack/.

[MMP] Sam Mclellan, Andrew Muddimer, and S. Camille Peres. "2012): The Effect of Experience on System Usability Scale Ratings". In: *Journal of Usability Studies* (), pp. 56–67.

[Mra04] Karoline Mrazek. *Erstellung von Software-Benutzerdokumentation.* [Online; zuletzt aufgerufen: 08-May-2021]. Feb. 2004. URL: https://www.infrasoft.at/images/downloads/Erstellung_von_Software-Benutzerdokumentation.pdf.

[Prü] Jochen Prümper. *Fragebogen ISONORM 9241/110-S Beurteilung von Software auf Grundlage der Internationalen Ergonomie-Norm DIN EN ISO 9241-110.* [Online; zuletzt aufgerufen: 04-December-2021]. URL: http://www.uselab.tu-berlin.de/wiki/images/6/62/ISONorm_Kurzversion.pdf.

[Rai02] Bas Raijmakers. "Usability ist ein Mittel, kein Ziel". In: *Usability.* Springer, 2002, pp. 129–157.

[Sei15] Carsten Seifert. *Spiele entwickeln mit Unity 5: 2D-und 3D-Games mit Unity und C# für Desktop, Web & Mobile.* Carl Hanser Verlag GmbH Co KG, 2015.

[SL16] Jeff Sauro and James R Lewis. *Quantifying the user experience: Practical statistics for user research.* Morgan Kaufmann, 2016.

[Tea21] Datatab Team. *Mann-Whitney U Test.* [Online; zuletzt aufgerufen: 30.12.2021]. 2021. URL: https://datatab.de/tutorial/mann-whitney-u-test.

[Uni21a] Unity. *Curved UI - VR Ready Solution To Bend / Warp Your Canvas!* [Online; zuletzt aufgerufen: 21-November-2021]. 2021. URL: https://assetstore.unity.com/packages/tools/gui/curved-ui-vr-ready-solution-to-bend-warp-your-canvas-53258.

[Uni21b] Unity. *RT-Voice PRO.* [Online; zuletzt aufgerufen: 21-November-2021]. 2021. URL: https://assetstore.unity.com/packages/tools/audio/rt-voice-pro-41068.

[VV14] Hans Van der Meij and Jan Van Der Meij. "A comparison of paper-based and video tutorials for software learning". In: *Computers & education* 78 (2014), pp. 150–159.

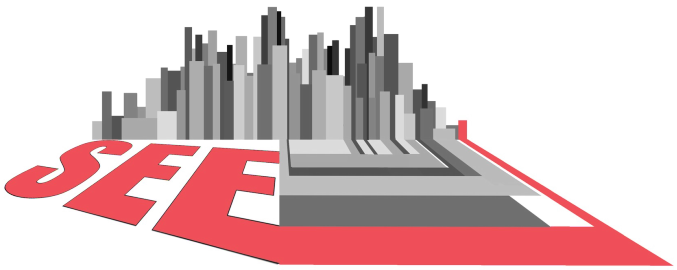
[Wil92] Frank Wilcoxon. "Individual comparisons by ranking methods". In: *Breakthroughs in statistics.* Springer, 1992, pp. 196–202.

[Zho+16] Xin Zhou et al. "A map of threats to validity of systematic literature reviews in software engineering". In: *2016 23rd Asia-Pacific Software Engineering Conference (APSEC).* IEEE. 2016, pp. 153–160.

ANHANG A

Benutzerdokumentation

SEE



Software Engineering Experience

A user guide

7th November 2021

Contents

1 Introduction

In this document, it will get explained all use cases and possibilities of interaction with the application **Software Engineering Experience** or rather **SEE**. It is structured by the use cases **architecture**, **quality**, **debugging**, **evolution** and the **navigation** for the cities and player. Look at the table of contents for a structured search or read this document for details depending of your aim. For further information look at the homepage of SEE:

https://see.uni-bremen.de/

2 Architecture

The goal of this part is to explain and describe all use cases in relation to the architecture. This includes all interactions inside of the player menu and moreover further interactions such as searching for a node etc..

2.1 Player Menu

To open the player menu press the key **Space**. It will display all of the possibilities to interact with the code-cities. Leftclick on the specific button to enter the entry. You can see an indicator on the right bottom which shows your current state.

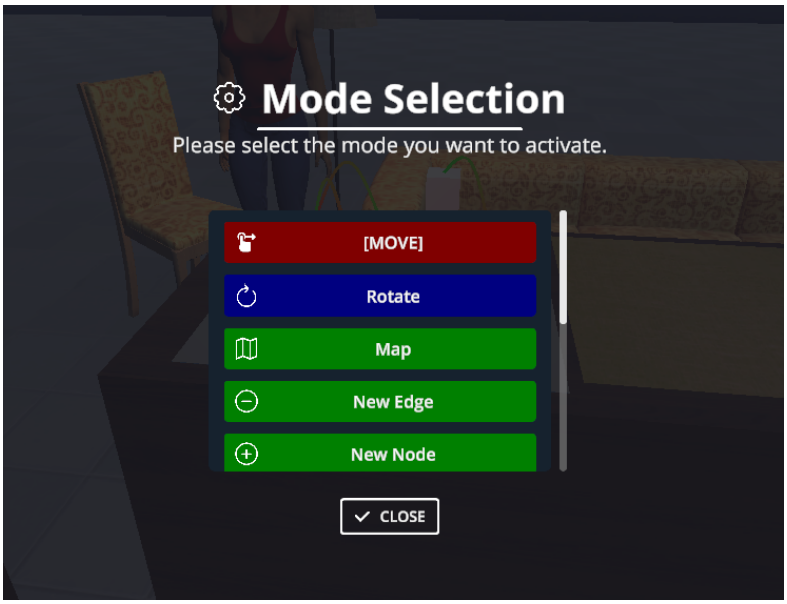


Figure 1: The player menu

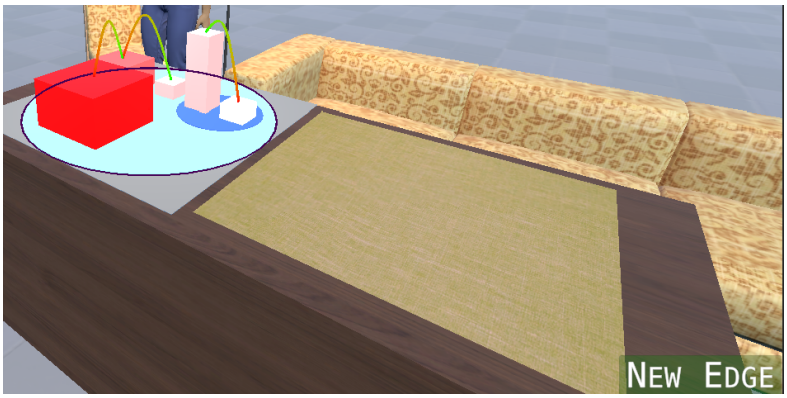


Figure 2: The state indicator which shows currently **new edge**

2.1.1 Add Edge

For adding a new edge to a city, press **Space** to open the player menu. Next, select the entry **add edge**. Select a start node for the new edge by leftclick on the specific node.
If selected a wrong start node, you can undo this selection by pushing key **F11**. If selected the right start node, select the target node by leftclick on the target node, too. The new edge will be created automatically.

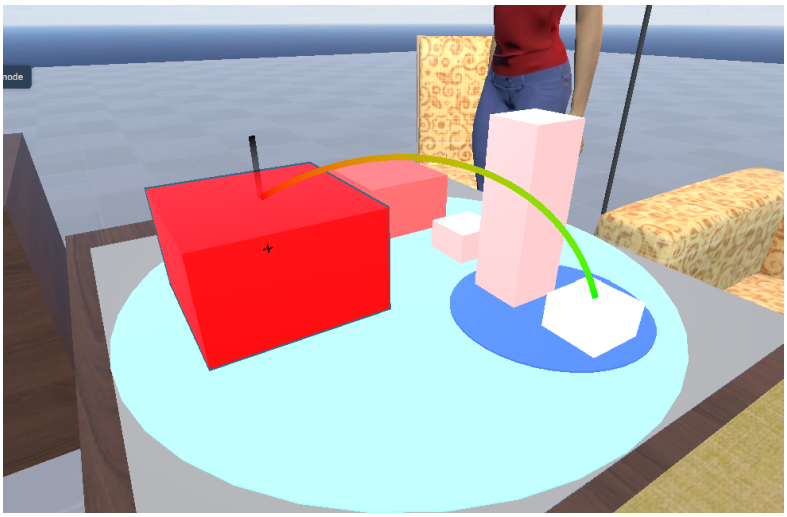


Figure 3: A new edge between two nodes

2.1.2 Edit Node

For editing an existing node of a city, press **Space** to open the player menu. Next, select the entry **edit node**. Select the node to edit by a leftclick. A dialog-window will appear and you can change all metric values of a node inside of the input fields. Confirm your editing with the button **OK** or delete it with

the button **CANCEL**.
Hint: The attribute **Node name** appears while hovering over the node. You can see the new value there, if it was added.

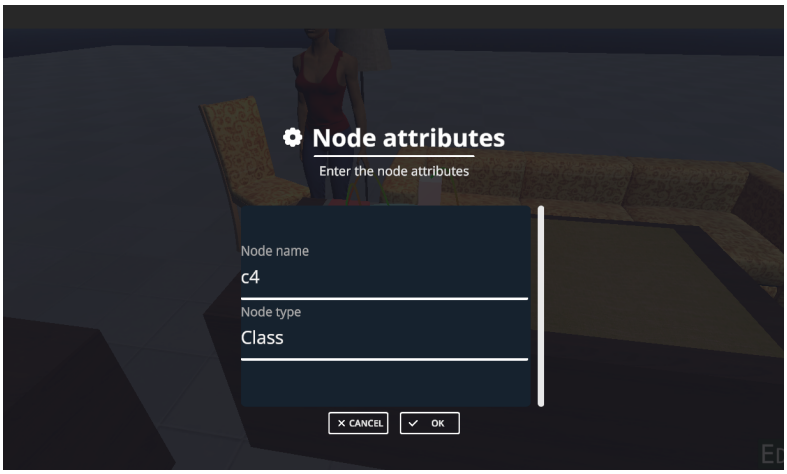


Figure 4: All metrics of the selected node

2.2 Hide Node

2.2.1 Hide all

Coming soon...

2.2.2 Hide incoming and outgoing

Coming soon...

2.2.3 Hide forward transitive closure/backward transitive closure

Coming soon...

2.2.4 Hide selected/Hide unselected

For hiding selected or unselected nodes or edges, press **Space** to open the player menu. Next, select the entry **Hide node**. Another menu will appear which shows multiple entrs. For this use case. select either **hide selected** or **hide unselected**.

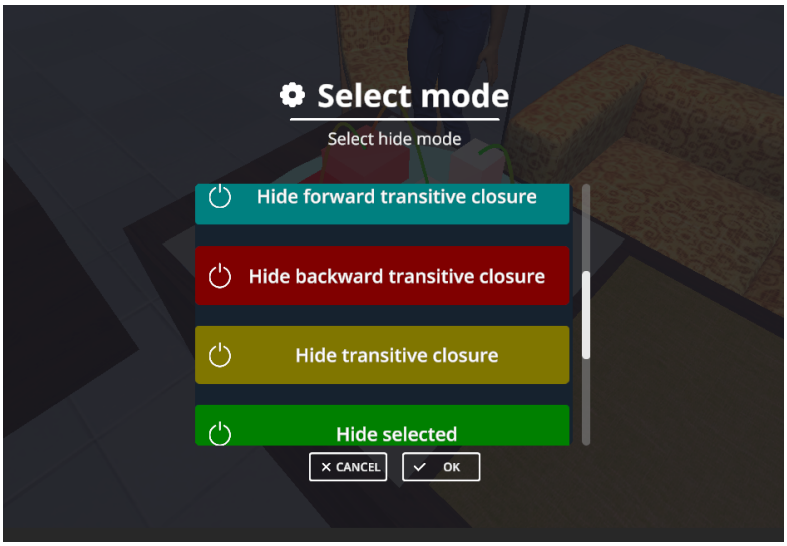


Figure 5: The hiding menu

Now you can select some objects of the city. If you want to select multiple objects, press key **STRG** (**CTRL**) while clicking on the nodes or edges. After finishing, push the button **Done** or push **Back** for stashing your changes.

Figure 6: The node selection - currently the blue and red node are selected

The selected or unselected objects depending of your selection are hidden now, you can close the menu with the button **CLOSE**.

Hint: If you hide a node with incoming or outgoing edges or child nodes, they are hidden now automatically, too. The same applies to all child nodes of child nodes and so on.

2.2.5 Hide edges

Coming soon...

2.2.6 Highlight connecting edges

Coming soon...

2.3 Search for a node

Search for a node by its name, therefore press key **F** to open the search dialog. There, you can insert a node name or a part of a node name. Press the button **OK** for searching or cancel the process with **CANCEL**.
If pressing the button **OK**, you will see the 5 best matching nodes ascending by their matching score.

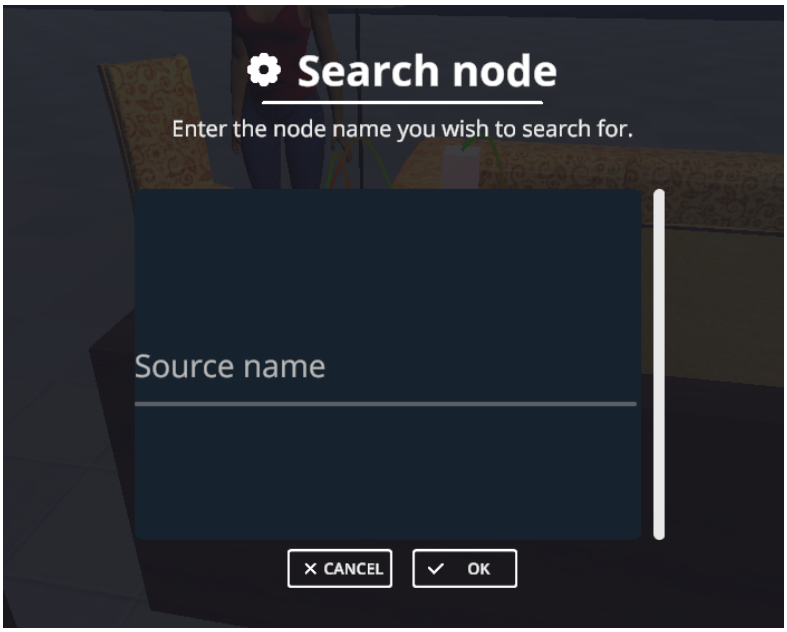


Figure 7: The search menu

Leftclick on the node name of the node you want to search for. The respective node will be highlighted now by blinking and a light beam for 15 seconds.

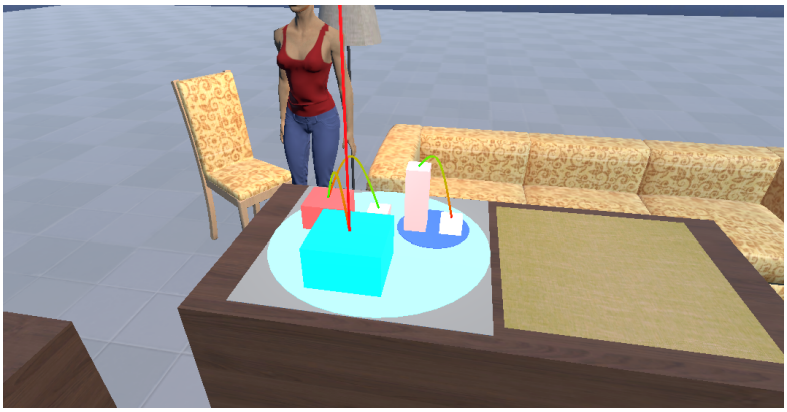


Figure 8: The highlighted Node is blinking, a light beam appears

3 Evolution

The goal of this part is to explain and describe all use cases in relation to the evolution.

3.1 Navigation bar

The navigation bar of the evolution allows a navigation and selection of versions of your software. It runs automatically after starting the evolution. You can select whether you want to skip forwards or backward. After pushing one of these buttons, it turns into a pause button which pauses the evolution again. On the right side of the navigation bar you can see the current version of the software which is displayed and the number of all versions behind.

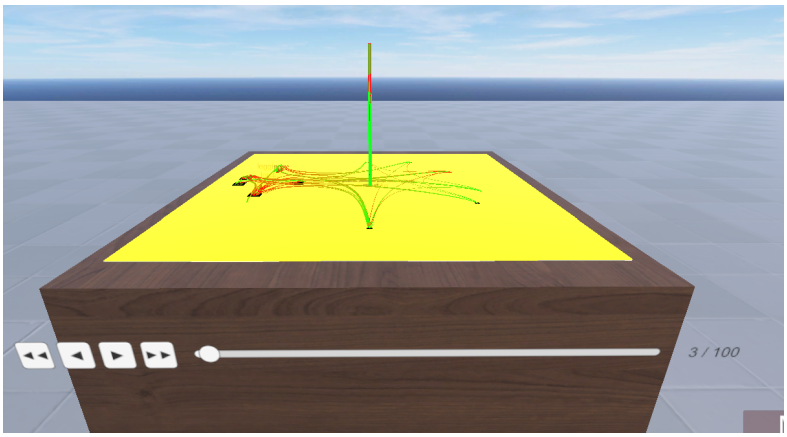


Figure 9: the navigation bar of the evolution

In addition, you can choose the speed of the software evolution. Therefore, you have to push the outer buttons multiple times:

- one time for **2x**
- two times for **4x**
- three times for **normal speed**

Furthermore, left click on the dragger on the progress bar and move them left or right for jumping to another iteration manually.

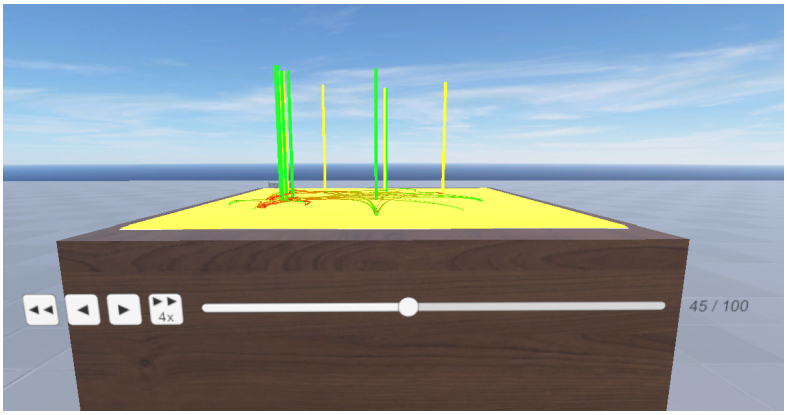


Figure 10: Fast speed forward

4 Debugging

The goal of this part is to explain and describe all use cases in relation to the part of debugging and software at run time.

Coming soon...

5 Quality

The goal of this part is to explain and describe all use cases in relation to the software quality and metrics.

Coming soon...

6 Navigation

The goal of this part is to explain and describe all use cases in relation to the navigation for example player navigation or city navigation.

6.1 Player Navigation

6.1.1 Switch table

You can switch the view of the table to look at another table. By default, your view is locked on the architecture-table. Press key **L** to unlock your view.

Now you can navigate in free space with the default **WASD** navigation. Focusing on another table with a locked view is currently not possible, but you may look at another table with the free navigation.

6.1.2 Simple navigation

Move forward and backward with the keys **W** and **S**. You can move faster by holding the **SHIFT** key while moving forward or backward.
If your view is free and not focused on a table, you can use keys **A** and **D** for moving left or right.
To switch your point of view, hold the right mouse-button and move the mouse. You can see that your position is changing. You can combine these navigation interactions simultaneously.

6.2 City Navigation

To get a better view on a code-city or see more details etc. you can navigate the code-cities. In the following you will get all possibilities of city navigation listed and explained.

6.2.1 Zoom into Code-Cities

If you want to see more details of big code-cities or you just want to see a part of it, you can zoom into these cities.
Therefore, hover over the specific code-city or rather the part of the city you want to zoom in and scroll the mouse wheel up. The city-elements outside of the table will be cutted out so you can see the zoomed elements isolated. You will zoom into the city focused on the cursor of your mouse. If you want to zoom out, scroll the mouse wheel down while the city is focused.

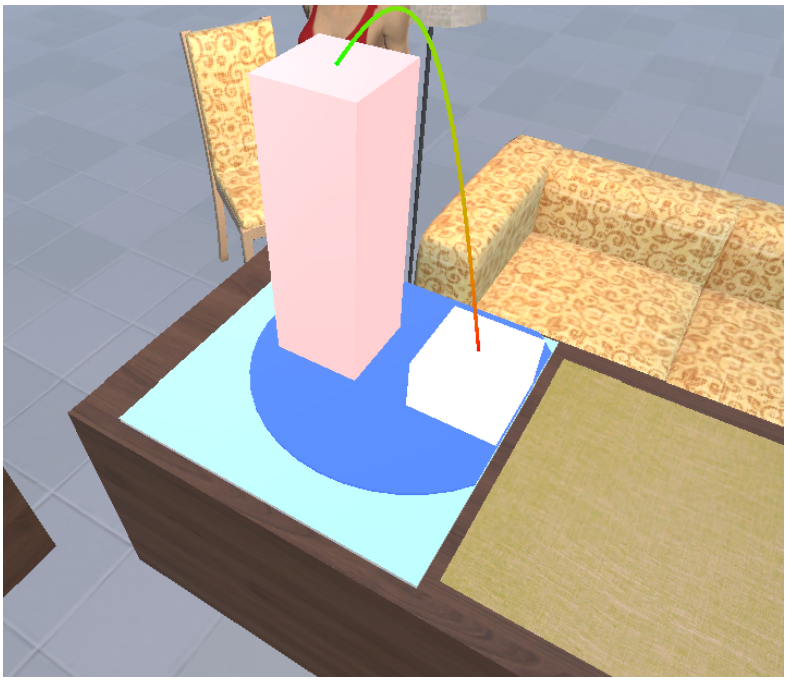


Figure 11: Zooming into a code-city

Hint: It is possible that your code-city is not centered anymore after zooming in or out. Press Key **R** to reset focused position to the middle of the city.

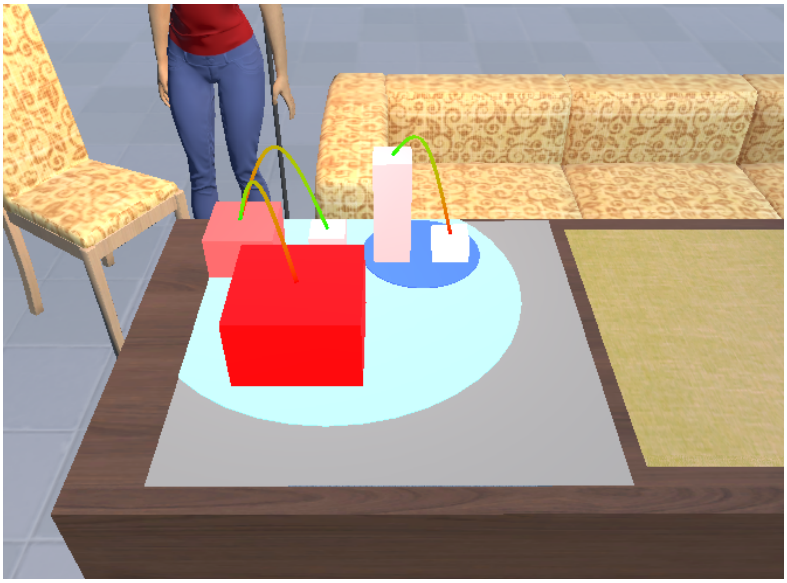


Figure 12: A not centered code-city. Press key **R** to reset it