Virtuelle Software-Städte annotieren und umgestalten

Bachelorarbeit

Hannes Masuch

Matrikelnummer: 4484296

16.12.2020

Fachbereich Mathematik / Informatik
Studiengang Wirtschaftsinformatik

1. Gutachter: Prof. Dr. Rainer Koschke
2. Gutachter: Dr. Rene Weller

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 16.12.2020

.....

(Hannes Masuch)

Danksagung

Ich möchte mich bei allen Menschen bedanken, die mich beim Erstellen dieser Bachelorarbeit unterstützt haben. Als Erstes möchte ich mich bei Prof. Dr. Rainer Koschke dafür bedanken, dass ich bei ihm meine Bachelorarbeit schreiben konnte und er mir immer alle meine Fragen beantwortet hat.
Ein besonderer Dank gilt meinen Freunden und meiner Familie, die mir stets zur Seite standen und mich moralisch unterstützt haben. Zum Schluss möchte ich allen anonymen Probanden danken, die sich auch unter, durch Covid-19 erschwerten Bedingungen, die Zeit genommen haben an meiner Evaluation teilzunehmen.

Ich beziehe mich in dieser Arbeit ausdrücklich auf beide Geschlechter, jedoch werde ich aufgrund der Einfachheit das generische Maskulin verwenden.

Zusammenfassung

Im Zuge dieser Arbeit ist eine Erweiterung zur Anwendung SEE enstanden, welche es ermöglicht Objekte einer Softwarestadt zu annotieren und die Softwarestadt umzugestalten. Diese Erweiterung lässt sich hierbei sowohl mit einer Desktop Umgebung, als auch in Virtual Reality verwenden.
Hierzu stellt diese Arbeit verschiedene Möglichkeiten der Umsetztung eines Annotationssystems vor. Diese umfasst verschiedene Arten der Eingabe von Annotationen, sowie verschiedene Möglichkeiten Interaktionen mit Annotationen umzusetzten. Außerdem wird durch die Erweiterung ermöglicht, ein zur Laufzeit von SEE erstelltes Layout einer Softwarestadt zu persistieren und so eine Umgestaltung von Softwarestädten zu ermöglichen. Hierzu werden verschiedene Varianten der Serialisierung vorgestellt und miteinander verglichen.
Im Kontext dieser Arbeit wurde eine Nutzerstudie zur Benutzerfreundlichkeit und Gebrauchstauglichkeit der Erweiterung unter Verwendung einer Virtual Reality Umgebung durchgeführt. Innerhalb dieser Studie wird ermittelt wie sich die Gebrauchstauglichkeit und Benutzerfreundlichkeit von SEE durch die hinzugefügte Erweiterung verändert. Hierzu stellt diese Arbeit verschiedene Fragebögen zur Ermittlung der Benutzerfreundlichkeit und Gebrauchstauglichkeit vor und vergleicht diese. Abschließend wird ein Fragebogen ausgewählt der die Kriterien für die Nutzerstudie am besten erfüllt. Im Zuge der Nutzerstudie konnte mit einer Konfidenz von 90% gezeigt werden, dass sich die Gebrauchstauglichkeit und Benutzerfreundlichkeit durch die hinzugefügte Erweiterung erhöht.
Abschließend werden verschiedene Themen und mögliche Forschungsfragen vorgestellt, die sich im Verlauf dieser Arbeit ergeben haben oder aus Anmerkungen von Probanden enstanden sind. Ein interessantes Thema ist beispielsweise die Gebrauchstauglichkeit und Benutzerfreundlichkeit von verschiedenen virtuellen Tastaturen in Virtual Reality, welche aufbauende auf diese Arbeit ermittelt und verglichen werden könnten.

INHALTSVERZEICHNIS

KAPITEL 1

Einleitung

In der Software Analyse existieren verschiedene Methoden und Modelle zur Analyse von Software, wie beispielsweise UML oder die strukturierte Analyse. Eine weitere Methode zur Analyse stellt die Softwarevisualisierung dar. Die Softwarevisualisierung beschäftigt sich mit der visuellen Darstellung von verschiedenen Eigenschaften von Software und deren Teilobjekten sowie ihrer Eigenschaften. Ziel der Softwarevisualisierung ist es komplexe Software durch eine möglichst einfache und abstrakte aber trotzdem umfassende Darstellung zu visualisieren und auf diese Weise verständlich zu machen.[1] Im Zuge dessen wurde 2007 von Wettel und Lanza die Stadtmethaper erstmalig vorgestellt.[2] In der Softwarevisualisierung gewinnt die Technologie, Virtual Realtiy in den letzten Jahren zunehmend an Bedeutung.
Das Projekt Software Engineering Experience, kurz SEE verbindet genau diese beiden Themengebiete. SEE stellt eine auf Basis von verschiedenen, wählbaren Metriken erzeugte virtuelle Stadt zur Verfügung, die sowohl mit einer klassischen Desktop Umgebung als auch unter Verwendung von Virtual Reality erkundbar ist.
Im Zuge dieser Arbeit werden nun Funktionen zu SEE hinzugefügt werden. Des Weiteren wird gezeigt werden, wie sich die Benutzerfreundlichkeit und Gebrachstauglichkeit von SEE durch die hinzugefügten Funktionen verändert. Die hinzuzufügenden Funktionen sind einerseits eine Annotationsfunktion. Diese soll es ermöglichen den Objekten einer Softwarestadt eine Annotation und somit zusätzliche explizite Informationen hinzuzufügen. Andererseits soll es möglich sein, die Position der Objekte zur Laufzeit zu speichern und der Softwarestadt auf diese Weise implizite Informationen hinzuzufügen.

1.1 Aufbau

Im folgenden Kapitel werden zunächst einige Grundlagen erklärt. Dies umfasst das Gebiet der Virtual Reality unter Verwendung der HTC Vive. Des Weiteren werden die Themen Unity und bereits vorhandene Funktionen von SEE, auf die im Verlauf dieser Arbeit aufgebaut wird

[1] C. Knight, "Visualisation for program comprehension: Information and issues," Jul. 1999. [Online]. Available: `https://www.researchgate.net/publication/2464933_Visualisation_for_Program_Comprehension_Information_and_Issues`, S.2.

[2] R. Wettel and M. Lanza, "Visualizing software systems as cities," Jun. 2007. DOI: `10.1109/vissof.2007.4290706`. [Online]. Available: `https://doi.org/10.1109/vissof.2007.4290706`.

betrachtet. Im dritten Kapitel wird der Entwurf der zu erstellenden Funktionen erläutert. In Kapitel Vier werden die Implementierungsdetails zur Umsetztung der benötigten Funktionen betrachtet. In Kapitel Fünf wird die an die Implementierung anschließende Evaluation erläutert. Dies umfasst den Aufbau sowie die Durchführung der Nutzerstudie. Als auch die Auswertung und die daraus resultierenden Ergebnisse. Das letzte Kapitel beinhaltet einen Ausblick über weitere Themen und Fragestellungen die im Verlauf dieser Arbeit aufgekommen sind.

KAPITEL 2

Grundlagen

2.1 Virtual Reality

Unter Virtual Reality oder virtueller Realität wird die Darstellung einer simulierten Umgebung verstanden, die von einer Person wahrgenommen werden kann[1]. Diese Umgebung wird im Normalfall auf einem Head-Mounted-Display dargestellt. Die Anwendungsgebiete von Virtual Reality sind hierbei vielfältig. Virtual Reality findet beispielsweise Anwendung in verschiedenen Forschungsbereichen der Medizin oder der Logistik.[2][3]

2.2 HTC Vive

HTC Vive ist ein Hardwaresystem, welches von HTC in Kooperation mit Valve entwickelt und 2015 erstmalig in Form eines Prototypen vorgestellt wurde. Das Hardwaresystem umfasst hierbei mehrere Ausführungen eines Head-Mounted-Displays. Zusätzlich zu den Head-Mounted-Displays existieren verschiedene Varianten von Controllern um mit der virtuellen Realität zu interagieren. In der Evaluation dieser Arbeit wird ein Head-Mounted-Display der Vive Serie in Kombination mit zwei Controllern der Serie verwendet werden.

[1] J. Steuer, "Defining virtual reality: Dimensions determining telepresence," *Journal of Communication*, vol. 42, no. 4, pp. 73–93, Dec. 1992. DOI: 10.1111/j.1460-2466.1992.tb00812.x. [Online]. Available: https://doi.org/10.1111%2Fj.1460-2466.1992.tb00812.x, S.7.

[2] F. Shu, W. Mi, and Z. Xu, "The information sharing platform for port container terminal logistics using virtual reality," in *2007 IEEE International Conference on Automation and Logistics*, 2007, pp. 2570–2575. DOI: 10.1109/ICAL.2007.4339013.

[3] T. Kesztyüs, M. Mehlitz, E. Schilken, *et al.*, "Preclinical evaluation of a virtual reality neuropsychological test system: Occurrence of side effects," *CyberPsychology & Behavior*, vol. 3, Jun. 2000. DOI: 10.1089/10949310050078788.

2.2.1 Head-Mounted-Display

Das Head-Mounted-Display verfügt über verschiedene Funktionen, wie beispielsweise die Einbindung der realen in die virtuelle Umgebung durch eine Kamera, die in der Front verbaut ist.[4] Im Zuge dieser Arbeit werden jedoch nicht alle Funktionen genutzt. Die genutzten Funktionen sind die Darstellung der virtuellen Realität auf den verbauten Bildschirmen. Zusätzlich wird der Bewegungssensor des Head-Mouindet-Displays verwendet um die Kopfbewegungen des Anwenders in der virtuellen Umgebung abzubilden und die angezeigte Umgebung dem entsprechend anzupassen. Außerdem wird das integrierte Mikrofon des Head-Mounted-Displays verwendet.

2.2.2 Controller

Die Controller der HTC Vive Serie verfügen über mehrere Möglichkeiten eine Eingabe zu tätigen. Zum einen verfügen die Controller über mehrere Knöpfe, sowie einem runden Touchpad. Des Weiteren werden die Bewegungen der Controller mithilfe einer Basisstation aufgenommen und in Virtual Reality übertragen. In Abbildung 2.1 ist die grundlegende Form, sowie die Tasten eines Controllers abgebildet. In Tabelle 2.1 sind die die zugehörigen Bezeichnungen der Tasten zu sehen.

Tabelle 2.1: Tastenbezeichnungen

1	Menü Taste
2	Trackpad
3	System Taste
4	Statuslampe
5	Micro-USB-Anschluss
6	Verfolgungssensor
7	Trigger
8	Griff Taste

[4] *Über das vive headset*, Zugriff: 09.10.2020. [Online]. Available: `https://www.vive.com/de/support/vive/category_howto/about-the-headset.html`.

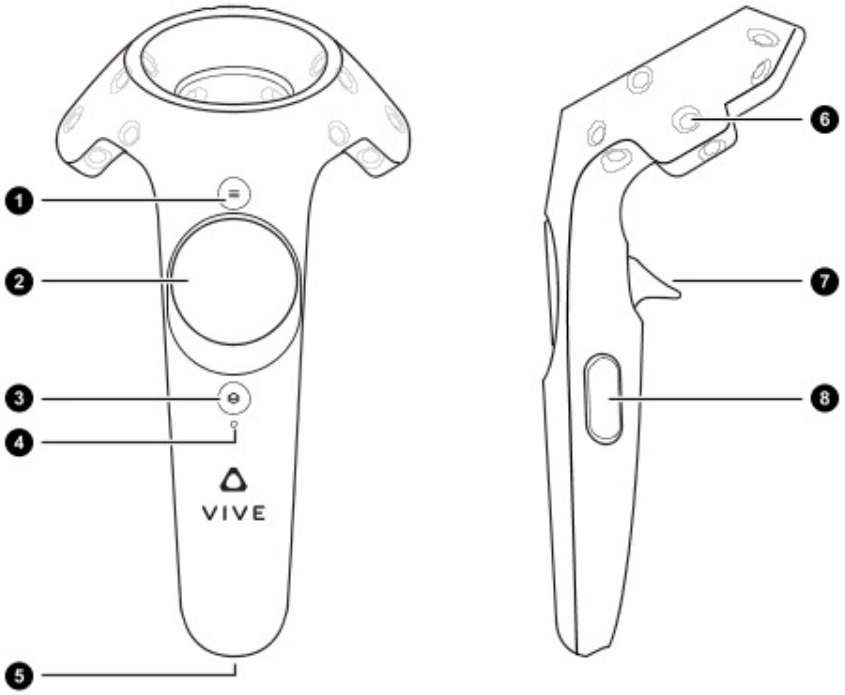

Abbildung 2.1: HTC Vive Controller
Vive controller, Zugriff: 10.10.2020. [Online]. Available:
`https://www.vive.com/media/filer_public/support_zip_img/de/www/vive/guid-2d5454b7-1225-449c-b5e5-50a5ea4184d6-web.png`

2.3 Unity

Unity oder auch Unity Engine ist eine erstmalig 2005 angekündigte Game Engine, welche auf C++ und C# basiert und mitlerweile mit nahezu allen Plattformen kompatibel ist.[5] Eine Game Engine ist ein Rahmenwerk, welches der Entwicklung von graphischen Anwendungen, hauptsächlich Videospielen dient. Die grundlegende Aufgabe einer Game Engine ist hierbei die Darstellung von graphischen Objekten. Die Unity Engine bietet jedoch noch einige weitere Funktionen. Eine solche Funktion ist beispielsweise die Kollisionserkennung zwischen Objekten[6].
Neben diesen integrierten Funktionen kann die Funktionalität der Unity Engine mithilfe von C#-Skripten erweitert werden. Dazu werden die zusätzlichen Skripte an Objekte angefügt. Auf diese Weise werden zusätzliche Funktionen der Objekte und zusätzliche Interaktionsmöglichkeiten mit den Objekten umgesetzt.[7]

2.4 SEE

SEE oder Software Engineering Experience ist ein auf der Unity Engine basierendes Programm, welches auf Basis von Metriken einer Software eine virtuelle Stadt erzeugt. Diese Stadt besteht aus Objekten, welche die Komponenten der Software abbilden. Die dargestellten Komponenten können hierbei sowohl Module, als auch Klassen der Software sein. Die Eigenschaften, wie beispielsweise Höhe oder Farbe der einzelnen Objekte dieser Stadt sind hierbei von den Metriken, wie “lines of code“ oder “number of tokens“, der zugehörigen Softwarekomponenten abhängig. Die Objekte einer Stadt können in verschieden Layouts angeordnet werden. Die Stadt kann einerseits mit einer klassischen Desktopumgebung betrachtet werden. Andererseits ist es auch möglich unter Verwendung eines Head-Mounted-Displays und der zugehörigen Controllern die Stadt zu betrachten. Im folgenden wird für die Interaktion in Virtual Reality eine Kombination von Head-Mounted-Display und Controllern aus der HTC Vive Serie verwendet.

2.4.1 Grundlegende Funktionen

SEE verfügt über einige Funktionen, die grundlegend für die folgende Arbeit sind. Die erste grundlegende Funktion ist die ShowInformation() Funktion, welche bei Betrachtung von Objekten aufgerufen wird. Beim Betrachten von Objekten wird der Name des Objekts oberhalb von diesem angezeigt. Diese Funktion ist Grundlage der Annotation. Annotationen sollen

[5] *Multiplatform*, Zugriff: 11.10.2020. [Online]. Available: `https://unity.com/de/features/multiplatform`.
[6] *Unity physics*, Zugriff: 11.10.2020. [Online]. Available: `https://docs.unity3d.com/Manual/PhysicsSection.html`.
[7] *Scripting*, Zugriff: 11.10.2020. [Online]. Available: `https://docs.unity3d.com/Manual/ScriptingSection.html`.

ebenfalls bei der Betrachtung eines Objektes angezeigt werden, sofern dieses Objekt Annotationen besitzt.
Objekte, die betrachtet werden, können zudem aufgehoben werden. Ist ein Objekt aufgehoben worden, kann der Nutzer dieses frei bewegen und an einer neuen Position ablegen oder die Bewegung abbrechen, wodurch das Objekt an seine ursprüngliche Position zurückkehrt. Diese Funktion ist insofern grundlegend, da sie die Grundlage für eine implizite Informationsdarstellung bietet. Um diese Form der Informationsdarstellung vollständig umzusetzten, bedarf es einer Möglichkeit die neuen Positionen zu speichern und somit die Informationen zu persistieren.
Eine weitere potenziell grundlegende Funktion von SEE ist, dass die Objekte einer Stadt eine Id haben. Diese Id könnte grundlegend für das Speichern der Anordnungen der Objekte sein, da sie eine zuverlässige Basis für das Zuordnen der Objekte zu ihren gespeicherten Eigenschaften darstellt.

KAPITEL 3

Entwurf

Die im Verlauf dieser Arbeit zu erstellende Erweiterung von SEE hat zwei klare Funktionen. Die erste Funktion ist ein Annotationssystem für die Objekte der Stadt zur Verfügung zu stellen. Dies umfasst mehrere Funktionalitäten, wie Hinzufügen, Löschen und Bearbeiten von Annotationen. Die Anforderungen an das Annotationssystem werden in den folgenden Abschnitten genauer erläutert.
Die zweite Funktion der Erweiterung ist das persistende Speichern der Position der Objekte der Stadt. Dabei muss eine eindeutige Zuordnung von Eigenschaften zu den Objekten möglich sein, um ein späteres Laden der Objekteigenschaften zu ermöglichen. Diese Objekteigenschaften sind einerseits die Position des Objekts, wodurch die Möglichkeit zur Speicherung von impliziten Informationen geboten wird. Andererseits müssen etwaige Annotationen, die zu Objekten hinzugefügt wurden, gespeichert werden. Auf diese Weise können implizite Informationen in Form vom benutzerdefinierten Objektanordnungen und explizite Informationen in Form der Annotationen gespeichert werden.

3.1 Annotationen

Im Folgenden werden die Anforderungen an die Annotationen dargelegt, dies umfasst einerseits die Darstellung der Annotationen sowie die verschiedenen Interaktionen. Wobei sowohl die Interaktionen mit Maus und Tastatur, als auch die Interaktionen unter Verwendung der HTC Vive und Virtual Reality beachtet werden müssen. Andererseits müssen die bestehenden Objekte erweitert werden, sodass sie annotierbar sind. Zusätzlich dazu existieren Anforderungen im Kontext der Speicherung und damit einhergehender Wiederherstellung von persistierten Annotationen.

3.1.1 Annotierbare Objekte

Die Erweiterung der Objekte, sodass diese annotierbar sind muss mehreren Anforderungen genügen. Einerseits müssen die annotierbaren Objekte die Annotation verwalten. Dies umfasst die Umsetzung der in den folgenden Abschnitten dargestellten Interaktionen. Zudem

müssen die annotierbaren Objekte das Einblenden bzw. Ausblenden der Annotationen bei Betrachtung des Objekts verwalten. Des Weiteren müssen annotierbare Objekte, welche bereits eine Annotation besitzen markiert sein, sodass direkt erkenntlich ist, dass diese annotiert sind.
Andererseits darf die Erweiterung nicht in bestehende Funktionalitäten eingreifen oder ungewollte Wechselwirkungen mit ihnen verursachen.

3.1.2 Darstellung

Die Darstellung der Annotationen muss mehreren Anforderungen genügen. Die Annotationen dürfen nur bei Betrachtung ihres Objekts sichtbar sein. Objekte die annotiert sind aber nicht betrachtet werden sind graphisch markiert, sodass erkennbar ist, dass sie annotiert sind. Bei Betrachtung eines annotierten Objekts müssen die Annotationen des Objekts so gestaltet sein, dass ihr Text in Virtual Reality als auch in einer Desktop Umgebung lesbar ist. Daraus resultierend dürfen die Annotationen eine feste Breite nicht überschreiten um lesbar zu bleiben. Zudem muss beachtet werden, dass ein Objekt mehrere Annotationen haben kann. Daraus resultierend sollten die Annotationen eines Objekts geordnet angezeigt werden und nicht verstreut sein oder sich möglicherweise überlappen.
Mit der Anzeige der Namen der Objekte existiert bereits eine textuelle Darstellung oberhalb der Objekte, welche beim Betrachten des Objekts sichtbar ist. Diese muss bei der Darstellung der Annotationen berücksichtigt werden um Konflikte und mögliche Überlagerungen zwischen den Texten auszuschließen.

3.1.3 Interaktion

Die Interaktionen mit einem zu annotierenden Objekt lassen sich grundsätzlich in zwei verschiedenen Formen umsetzten. Einerseits wäre es möglich jeder Interaktion, also dem Hinzufügen, Löschen und Bearbeiten von Annotationen eine eigene Taste zuzuweisen. Diese Tasten müssten dann bei Betrachtung eines Objekts gedrückt werden um die jeweilige Interaktion hervorzurufen.
Die zweite Möglichkeit besteht darin, dass ein Annotationmenü entworfen wird. Dieses kann bei Betrachtung eines Objekts durch einen Tastendruck einer vorher festgelegten Taste geöffnet werden. Das Annotationmenü bietet hierbei verschiedene Auswahlmöglichkeiten um die unterschiedlichen Interaktionen anzustoßen.
Aufgrund der begrenzten Tastenanzahl der HTC Vive Controller ist die erste Möglichkeit im Kontext von Virtual Reality nicht gut umsetzbar. Daraus resultierend wird im restlichen Verlauf dieser Arbeit nur die Interaktion mit einem Annotationsmenü betrachtet.

3.1.3.1 Annotationsmenü

Um das Annotationsmenü zu öffnen müssen die Objekte, die möglicherweise annotiert werden sollen eine Interaktionsmöglichkeit bieten. Aus diesem Annotationsmenü heraus müssen Interaktionsmöglichkeiten zum Hinzufügen, Löschen und Bearbeiten der Annotationen des Objekts auswählbar sein. Die Auswahl der Interaktionsmöglichkeit muss hierbei für eine Desktop sowie eine Virtual Reality Umgebung möglich sein.
Bei der Umsetzung der Interaktionen gibt es erneut verschiedene Möglichkeiten. Diese werden in den folgenden Abschnitten erläutert und die spätere Umsetzung wird festgelegt.

3.1.3.2 Hinzufügen

An das Hinzufügen der Annotationen stellt sich als Hauptanforderung die Eingabe des Textes, der annotiert werden soll. In einer Desktop Umgebung ist dies kein Problem, da der Text einfach mit der Tastatur eingegeben werden kann. Unter Verwendung der HTC Vive und Virtual Reality ist dies jedoch nicht möglich. Als Alternativen bieten sich hierbei zwei Möglichkeiten.
Die erste Möglichkeit ist die Verwendung einer virtuellen Tastatur. Im Verlauf der Recherchen zu virtuellen Tastaturen zeigte sich, dass bereits eine Vielzahl von unterschiedlichen Konzepten für virtuelle Tastaturen existieren. Diese virtuellen Tastaturen verfügten teilweise über komplexe Funktionen, wie eine automatische Vervollständigung. Es zeigt sich jedoch schnell als Problem, dass den nachweislich gut bedienbaren Tastaturen eine komplexere Struktur, wie bespielweise eine Kollisionserkennung oder Gestenerkennung zugrunde lag.[1][2] Im folgenden werden zwei einfachere Varianten vorgestellt, die potentiell im Kontext dieser Arbeit erstellt werden könnten.
Die erste Variante der Bildschirmtastatur wird nur mit dem Head-Mounted-Display bedient. Hierbei wird der Bewegungssensor genutzt um die Bewegungen des Kopfes abzubilden und so das Zentrum des Sichtfelds zu erfassen. Auf diese Weise können Buchstaben auf einer statischen Tastatur ausgewählt werden, wenn sie sich für eine festgelegte Zeit im Zentrum des Sichtfeldes befinden.
Die zweite Variante verwendet die Controller der HTC Vive um mit ihnen eine Auswahl auf der Bildschirmtastatur zu treffen. Hierzu wird ein Strahl vom Ende des Controllers geworfen. Wenn sich dieser Strahl auf einen der Buchstaben zeigt und eine vorher festgelegt Taste gedrückt wird, wird der Buchstabe eingegeben.
Die zweite Möglichkeit ist das in das Head-Mounted-Display integrierte Mikrofon zu verwenden. Mithilfe dieser audioellen Eingabemöglichkeit und einem Diktieralgorithmus kann

[1] C. Boletsis and S. Kongsvik, "Text input in virtual reality: A preliminary evaluation of the drum-like VR keyboard," *Technologies*, vol. 7, no. 2, Apr. 2019. DOI: 10.3390/technologies7020031. [Online]. Available: https://doi.org/10.3390/technologies7020031.

[2] C. Yu, Y. Gu, Z. Yang, *et al.*, "Tap, dwell or gesture?" In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, ACM, May 2017. DOI: 10.1145/3025453.3025964. [Online]. Available: https://doi.org/10.1145/3025453.3025964.

ebenfalls eine Texteingabe realisiert werden. Bei der Recherche zu einem Diktieralgorithmus zeigte sich, dass Unity die Möglichkeit bietet den Diktieralgorithmus von Windows 10 zu nutzen. Zur Verwendung von diesem Algorithmus existiert eine ausführliche Dokumentation sowie ein Beispiel auf der Unity Website.[3]
Diese Arbeit wird sich auf die audioelle Eingabe in Virtual Reality beschränken. Die Gründe hierfür sind einerseits, dass die vorgestellten virtuellen Tastaturen im Vergleich deutlich langsamer zu bedienen sind. Des Weiteren existieren bessere Varianten einer virtuellen Tastatur, welche jedoch nicht im Rahmen dieser Arbeit umsetztbar wären. Andererseits existiert mit dem gut dokumentierten Diktieralgorithmus eine schnellere und zuverlässige Alternative.

3.1.3.3 Löschen

An das Löschen von Annotationen stellt sich als Hauptanforderung wie die zu löschende Annotation ausgewählt wird. Hierbei lässt sich zwischen zwei Varianten unterscheiden.
Einerseits wäre es möglich, dass immer die zuletzt hinzugefügte Annotation entfernt wird. Dies wäre jedoch nicht besonders nutzerfreundlich wenn beachtet wird, dass möglicherweise mehrere Nutzer einem Objekt verschiedene Annotationen hinzugefügt haben.
Andererseits wäre es möglich, die zu löschende Annotation direkt auszuwählen. In einer Desktop Umgebung wäre dies einfach mit der Maus umsetzbar. Unter Verwendung der HTC Vive und Virtual Reality wäre dies mithilfe eines Strahls, der von einem Controller aus geworfen wird umsetzbar.
Im Verlauf dieser Arbeit wird die zweite Variante angewand, da sie deutlich nutzerfreundlicher ist. Zudem wird ein ähnlicher Auswahlmechanismus für das Bearbeiten der Annotationen sowie das Annotationsmenü benötigt und stellt somit keinen Mehraufwand dar.

3.1.3.4 Bearbeiten

An das Bearbeiten von Annotationen stellen sich zwei Hauptanforderungen. Diese Anforderungen sind einerseits die Auswahl der zu bearbeitenden Annotation und andererseits die Eingabe des neuen Textes dieser Annotation. Beide Anforderungen ähneln den zuvor aufgeführten Anforderungen des Hinzufügens und Löschens oder sind identisch mit ihnen. Daher benötigt das Bearbeiten von Annotationen keine grundsätzlich neuen Funktionen.

3.1.4 Speichern und Laden

Die Hauptanforderung an die Annotationen im Kontext von Speichern und Laden ist, dass die Annotationen speicherbar und ladebar sind. Dies lässt sich gewährleisten, indem die Anno-

[3] *Dictationrecognizer*, Zugriff: 17.10.2020. [Online]. Available: `https://docs.unity3d.com/ScriptReference/Windows.Speech.DictationRecognizer.html`.

tationen Teil des annotierten Objekts sind. Auf diese Weise können sie vom in Abschnitt 3.2 beschriebenen Speicher- und Ladesystem neben der Position der Objekte berücksichtigt und gespeichert werden.

3.2 Speicher- und Ladesystem

Die Hauptanforderung an das Speichern und Laden ist, dass die gespeicherten Eigenschaften ihren Objekten eindeutig zuordenbar sind. Dies bedeutet konkret, dass die Eigenschaften eines Objekts eindeutig serialisiert werden und bei Bedarf deserialisiert werden können. Eine Möglichkeit dies zu gewährleisten wäre die Zuordnung über die Namen der Objekte. Ein offensichtlicher Nachteil hierbei wäre, dass bei einer Namensänderung von Objekten die zugeordneten Annotationen, sowie die Position verloren gehen würden.
Alternativ hierzu kann die Zuordnung über die bereits vergebenen Ids der Objekte gewährleistet werden. Unter der Annahme, dass Objekten unabhängig ihrer Eigenschaften immer die selbe Id zugeordnet wird, würden sich hierbei keine Nachteile ergeben.
Im Verlauf dieser Arbeit wird mit der Zuordnung über die Objekt-Ids gearbeitet, da diese im Vergleich keine Nachteile besitzt. Außerdem muss festgelegt werden wie das Speichern und Laden angestoßen wird. Hierbei bieten sich zwei verschiede Möglichkeiten. Einerseits wäre es möglich dem Speichern eine eigene Tastenbelegung zuzuweisen. Andererseits wäre es möglich in festgelegten zeitlichen Abständen automatisch, sowie bei Beendigung des Programms zu speichern. Da die Anzahl an Tasten der HTC Vive Controller, wie bereits in Abschnitt 3.1.3 erwähnt, begrenzt ist, wird in dieser Arbeit die Variante der automatischen Speicherung umgesetzt. Zusätzlich dazu kann das Speichern mithilfe eines Knopfes im Inspector Menü von Unity angestoßen werden. Das Laden kann ebenfalls durch einen solchen Knopf angestoßen werden.
Neben der Hauptanforderung muss ausgewählt werden, in welchem Datenformat gespeichert werden soll. Hierbei ist zwischen binären und nicht binären Datenformaten zu unterscheiden. Im folgenden Abschnitt werden die beiden Möglichkeiten samt ihrer Vorteile und Nachteile aufgezählt und gegeneinander abgewogen. Anschließend wird festgelegt, welche Variante verwendet wird.

3.2.1 Vergleich Dateiformate

Wie bereits angedeutet ist es möglich zwei verschiedene Arten von Dateiformaten zur Serialisierung und Deserialisierung zu verwenden. Einerseits wäre es möglich die Daten im Binärformat zu speichern. Das .NET Framework stellt hierzu eine Dokumentation sowie ein einfaches Beispiel bereit.[4] Die Speicherung im Binärformat bringt jedoch einige Nachteile sowie Vorteile

[4] *Binary serialization*, Zugriff: 24.10.2020, Jan. 2018. [Online]. Available: `https://docs.microsoft.com/de-de/dotnet/standard/serialization/binary-serialization`.

mit sich. Ein Merkmal der binären Serialisierung ist, dass eine Binärdatei nicht für Menschen lesbar ist. Dies bringt wiederum einen Nachteil mit sich, vor welchem in der Dokumentation gewarnt wird.[5] Die binäre Serialisierung ist demnach nicht sicher, da vor der Deserialisierung nicht der Inhalt geprüft werden kann. Dies führt dazu, dass beim Deserialisieren von nicht vertrauenswürdigen Dateien Schaden am System des Nutzers entstehen kann. Dies ist in diesem Kontext jedoch zu vernachlässigen, da die zu deserialisierenden Datein selbst erstellt und somit schadsoftwarefrei sind. Ein Vorteil der binären Serialisierung ist, dass sie sehr einfach umzusetzten ist, wie in Listing 3.1 und Listing 3.2 zu sehen ist. Ein weiterer Vorteil ist, dass die binäre Serialisierung "typtreu" ist. Dies hat den Vorteil, dass der Typ des serialisierten Objekts mitgespeichert wird. Der subjektiv größte Vorteil der binären Serialisierung ist, dass ich bereits im Zuge eines privaten Projekts Erfahrungen mit einer binären Serialisierung gemacht habe.

```
1  [Serializable]
2  public class MyObject {
3    public int n1 = 0;
4    public int n2 = 0;
5    public String str = null;
6  }
```

Listing 3.1: Beispielklasse einfache Serialisierung
6

```
1  MyObject obj = new MyObject();
2  obj.n1 = 1;
3  obj.n2 = 24;
4  obj.str = "Some String";
5  IFormatter formatter = new BinaryFormatter();
6  Stream stream = new FileStream("MyFile.bin", FileMode.Create,
7  FileAccess.Write, FileShare.None);
8  formatter.Serialize(stream, obj);
9  stream.Close();
```

Listing 3.2: Beispiel einfache Serialisierung
7

Andererseits wäre es möglich die Daten in einer JSON Datei zu speichern. Hierzu wird ebenfalls eine Dokumentation, sowie Beispiel durch das .NET Framework bereitgestellt.[8] Die Verwendung von JSON zur Serialisierung und Deserialisierung bringt ebenfalls Vorteile als auch

[5] *Binaryformatter security guide*, Zugriff: 24.10.2020, Jul. 2020. [Online]. Available: `https://docs.microsoft.com/de-de/dotnet/standard/serialization/binaryformatter-security-guide`.

[8] *Json serialization and deserialization (marshalling and unmarshalling) in .net - overview*, Zugriff: 24.10.2020, Jan. 2020. [Online]. Available: `https://docs.microsoft.com/de-de/dotnet/standard/serialization/system-text-json-overview`.

Nachteile mit sich. Ein Vorteil ist, dass eine JSON Datei für Menschen lesbar ist. Im Gegensatz zu einer binären Serialisierung ist eine JSON basierte Serialisierung nicht “typtreu“. Hieraus könnten möglicherweise Fehler bei der Deserialisierung entstehen.
Im Zuge dieser Arbeit wird im Folgenden eine binäre Serialisierung umgesetzt. Der Hauptgrund hierfür ist die Tatsache, dass bereits Erfahrung in der Umsetzung und Handhabung einer binären Serialisierung vorhanden ist. Zudem existieren keine gravierenden Gründe die gegen eine binäre Serialisierung und für eine JSON basierte Serialisierung sprechen.

KAPITEL 4

Implementierung

4.1 Annotationen

Im Folgenden wird auf die Implementierung der Annotationen sowie dem zugehörigen Annotationsmenü eingegangen. Hierzu wird zuerst auf die Implementierung der annotierbaren Objekte eingegangen. Anschließend werden die verschiedenen Seiten des Annotationsmenüs und die integrierte Diktierfunktion vorgestellt.

4.1.1 Annotierbare Objekte

Wie bereits in Abschnitt Abschnitt 3.1.1 dargelegt sollen die annotierbaren Objekte eine Erweiterung zu den bestehenden Objekten sein. Daraus resultierend ist es sinnvoll, dass die annotierbaren Objekte eine abgeleitete Klasse der bestehenden Objekte sind. Zudem wird hierdurch die Nutzung der in Abschnitt 2.4.1 erwähnten Funktion, ShowInformation(), zur Darstellung der Namen oberhalb des Objekt vereinfacht, da diese Bestandteil der bestehenden Objekt Klasse ist. Dies vereinfacht die Umsetztung der Annotationen deutlich, da die Namen der Objekte mithilfe eines PreFabs, also einer Vorlage erzeugt werden. Somit können Annotationen, unter Verwendung des PreFabs sowie einer angepassten ShowInformation() Funktion dargestellt werden. Zudem existiert mit der HideInformation() Funktion die Möglichkeit die Annotation nicht mehr anzuzeigen, wobei diese bei einem erneuten Aufruf der ShowInformation() Funktion wieder angezeigt werden. Somit können durch eine Erweiterung der bestehenden Objektklasse, sowie ihrer Funktionen, Annotationen dargestellt und verwaltet werden.

4.1.2 Annotationsmenü

Wie bereits in Abschnitt 3.1.3.1 angedeutet muss das Annotationsmenü drei Interaktionsmöglichkeiten bieten. Zusätzlich dazu sollte das Annotationmenü eine Interaktion zum Schließen des Selbigen anbieten. Die Darstellung eines solchen Menüs lässt sich in Unity mithilfe der GUI Elemente erzeugen. Die Verwendung von GUI Elementen bedarf immer einen Canvas auf dem

die Elemente angeordnet werden. Dies kann dann als ein Unity PreFab, also eine Vorlage gespeichert werden. Ein Beispiel hierfür ist in Abbildung 4.1 zu sehen.

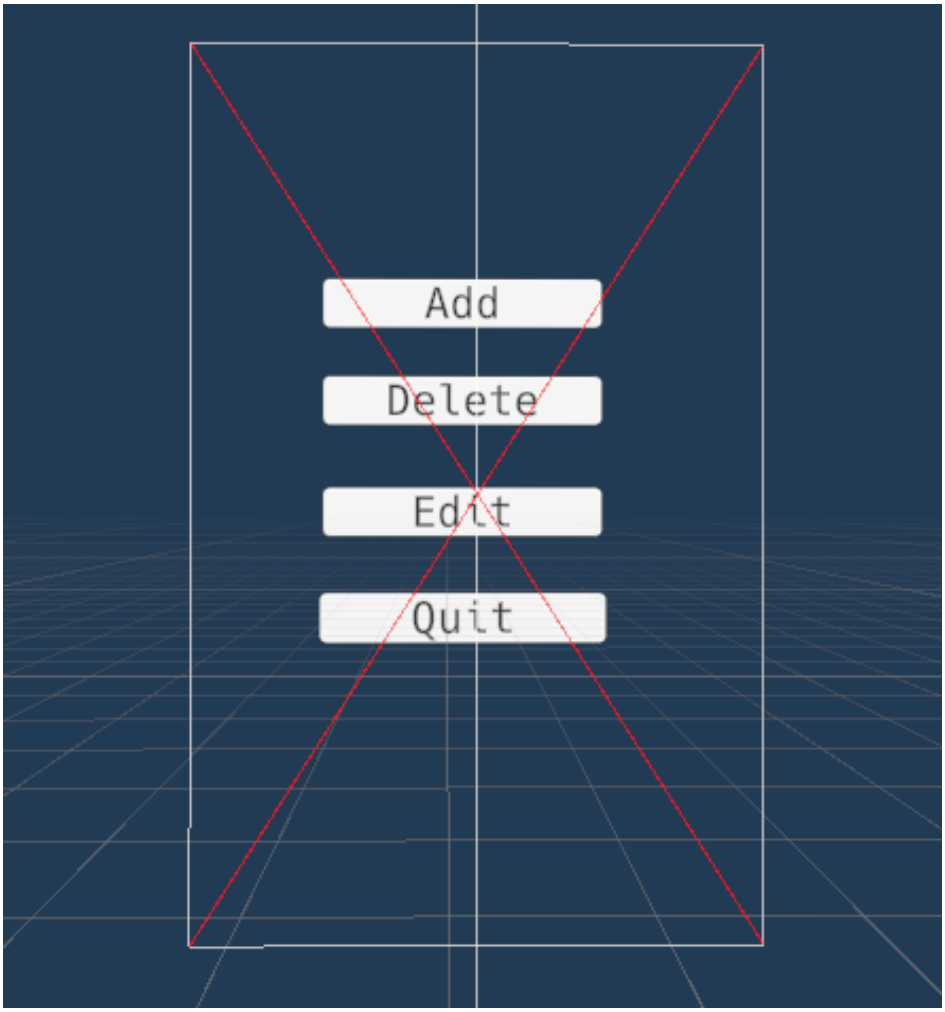

Abbildung 4.1: Annotationmenü PreFab

Um dieses Menü nun zu verwenden kann es bei Bedarf einfach aus dem PreFab heraus erzeugt werden. Hierbei muss jedoch beachtet werden, wie der Render Mode des Canvas gesetzt ist. Unter Verwendung des Render Modes “Screen Space Overlay“ wird die GUI über den Bildschirm gelegt und ist somit nicht Teil der 3D Umgebung. Dies führt dazu, dass die Interaktionen unter Verwendung von Virtual Reality nicht umsetztbar sind. Dies ist dadurch begründet, dass die Controller der HTC Vive Teil der 3D Umgebung sind und somit ebenfalls von der GUI überlagert werden. Selbiges gilt für den Render Mode “Screen Space Camera“. Die Lösung hierfür ist der Render Mode “World Space“, bei welchem der Canvas als ein 3D Objekt erzeugt wird, wodurch Interaktionen mit den HTC Vive Controllern möglich werden. Hierbei kann der Canvas wie jedes andere Objekt frei in der 3D Umgebung platziert werden. Die Interaktion zwischen dem Menü und den Controllern kann nun mithilfe eines Strahls der von einem Controller aus geworfen wird gewährleistet werden. Ein Beispiel hierfür ist in Abbildung 4.2 zu sehen.

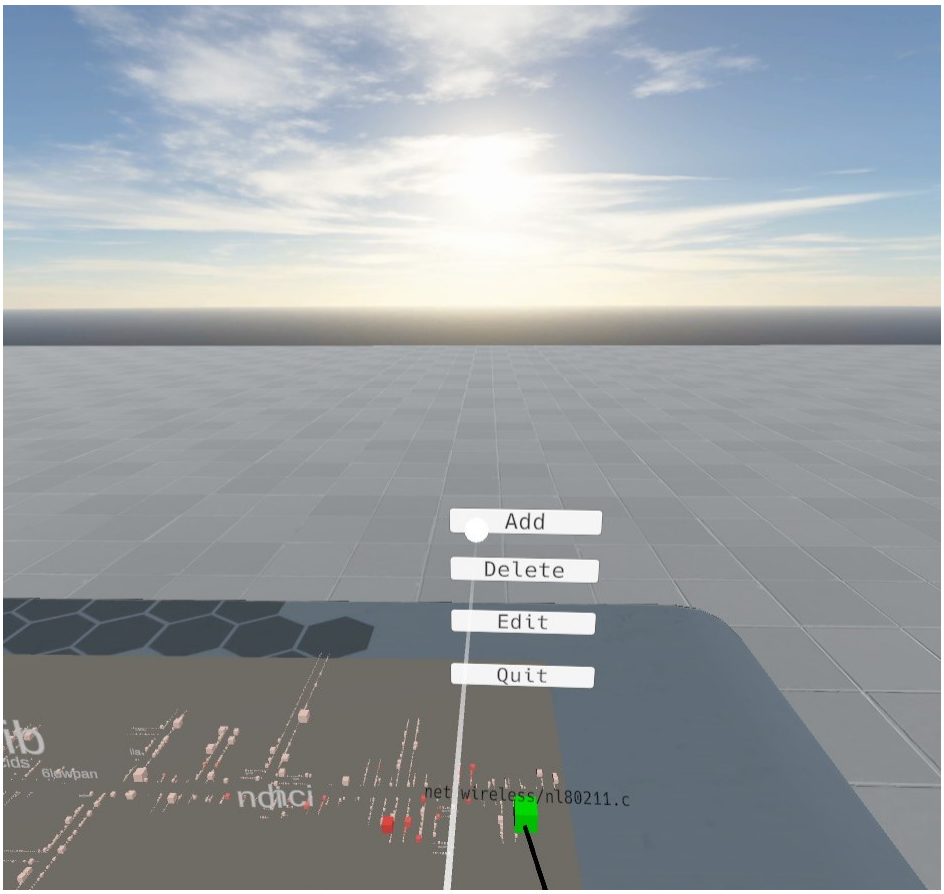

Abbildung 4.2: Annotationmenü Virtual Reality

4.1.2.1 Annotation hinzufügen

Das Hinzufügen von Annotationen, insbesondere die Eingabe des Texts benötigt ebenfalls ein Menü. Dieses Menü kann hierbei auf dem selben Canvas, wie das Hauptmenü erzeugt werden. Neben einer Interaktionsmöglichkeit zum Starten und Stoppen des Diktiervorgangs sowie dem finalen Hinzufügen der Annotation zum betrachteten Objekt, muss das Menü ein Textfeld enthalten. Dies wird einerseits zur Überprüfung des Diktierten benötigt, andererseits kann hier unter Verwendung einer Desktop Umgebung eine Eingabe mit der Tastatur gemacht werden. Zusätzlich dazu wird noch eine Interaktionsmöglichkeit zum Zurückkehren zum Hauptmenü geboten. Die Umsetztung dieses Menüs ist in Abbildung 4.3 zu sehen.

Wie bereits in Abschnitt 3.1.2 erwähnt sollen alle Objekte die annotiert sind eine Markierung haben, sodass dies sofort erkenntlich ist. Die Umsetzung der Markierung ist in Abbildung 4.4 zu sehen.

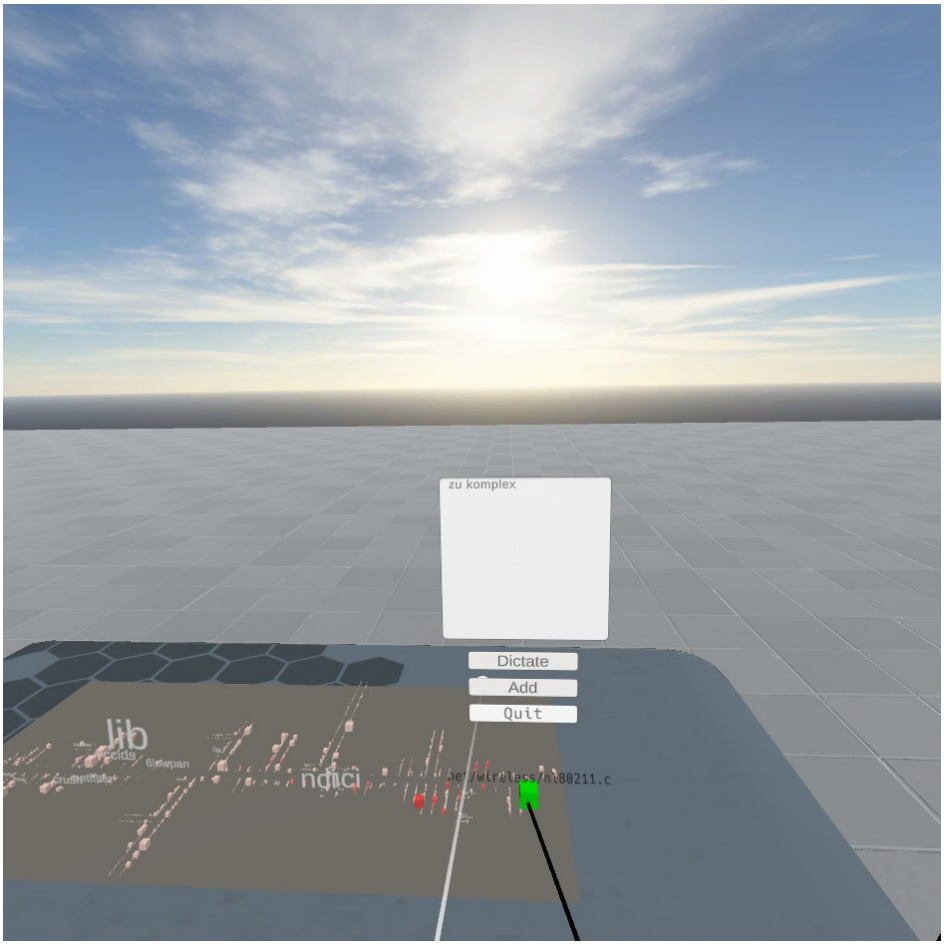

Abbildung 4.3: Annotationmenü Hinzufügen

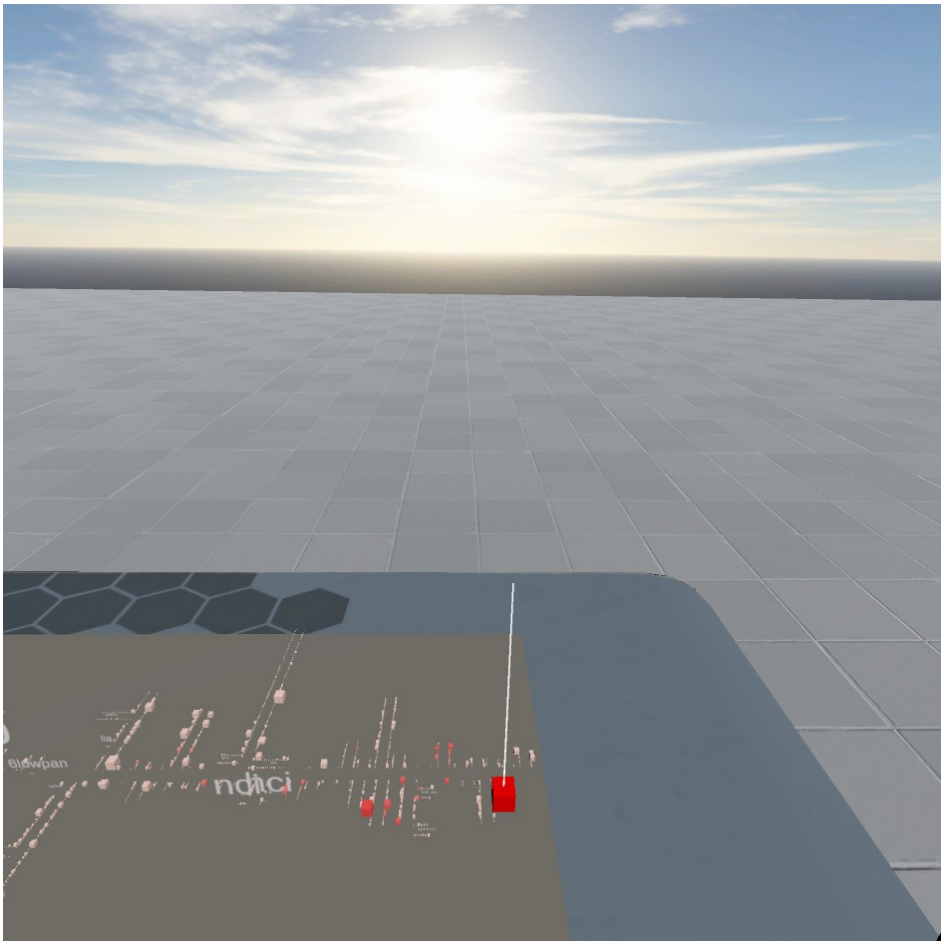

Abbildung 4.4: Annoatiationsmarkierung

4.1.2.2 Diktieren

Wie bereits in Abschnitt 3.1.3.2 festgelegt soll die Texteingabe für die Annotationen in Virtual Reality über eine Diktierfunktion getätigt werden. Hierzu bietet Unity eine eigene Klasse, welche die Diktierfunktion von Windows 10 verwendet. Um diese erfolgreich zu verwenden muss zuvor die Diktierfunktion in Windows aktiviert werden.

4.1.2.3 Annotation löschen

Das Löschen von Annotationen bedarf kein weiteres Menü. Stattdessen wird bei Auswahl der Löschen Interaktion im Hauptmenü der Canvas mit seinen Elementen ausgeblendet. Nun kann mithilfe des Strahls, welcher sonst zur Bedienung der Menüs verwendet wird, eine Annotation ausgewählt werden. Diese wird daraufhin entfernt. Sollte das Objekt noch weitere Annotationen haben werden diese neu angeordnet um den entstandenen Raum korrekt aufzufüllen. Die Auswahl einer Annotation ist in Abbildung 4.5 zu sehen.

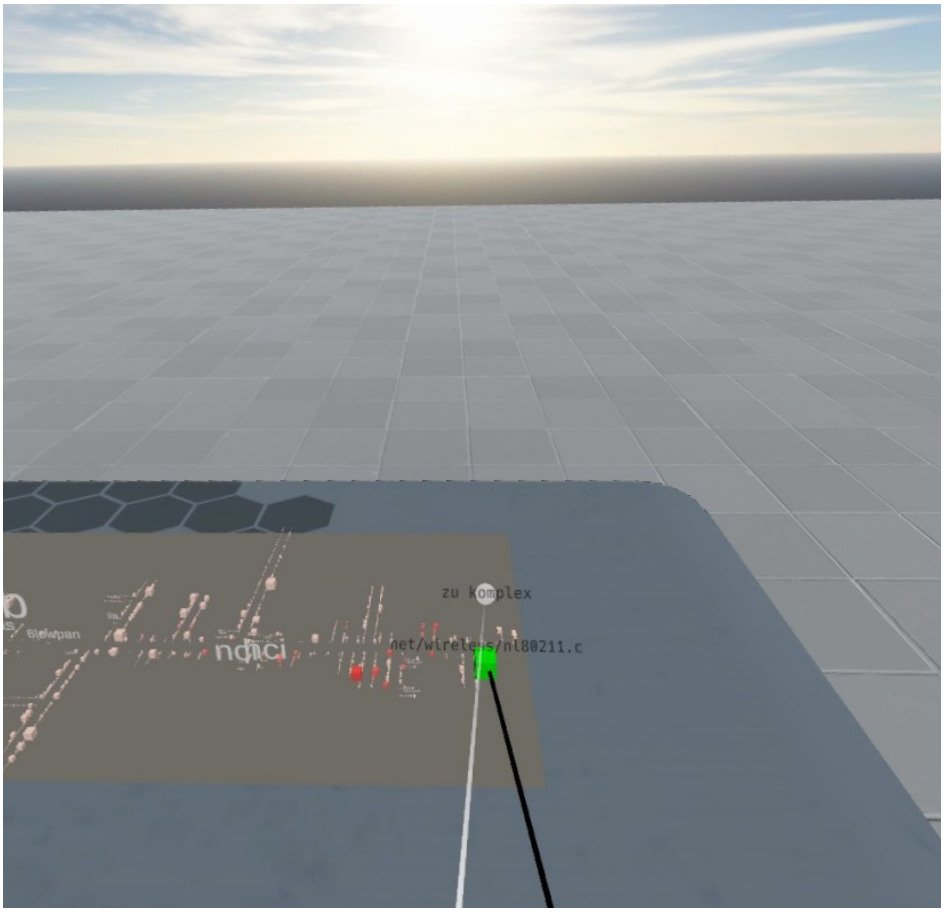

Abbildung 4.5: Annotation zum Löschen wählen

4.1.2.4 Annotation editieren

Das Editieren von Annotationen benötigt ebenfalls ein eigenes Menü. Dieses ist vom Aufbau identisch mit dem Menü um Annotiationen hinzuzufügen. Anders als beim Hinzufügen wird dieses nicht beim Auswählen der Interaktion im Hauptmenü geöffnet. Stattdessen wird analog zum Löschen von Annotationen der Canvas ausgeblendet. Nun kann eine Annotation des Objekts ausgewählt werden. Wurde eine Annotation ausgewählt, öffnet sich das Menü und der Text der ausgewählten Annotation befindet sich im Textfeld. Dieser kann nun unter Verwendung einer Desktop Umgebung angepasst werden. Unter Verwendung einer Virtual Reality Umgebung kann mithilfe der Diktierfunktion eine neue Annotation diktiert werden, welche die zuvor ausgewählte Annotation ersetzt.

4.2 Speichern- und Laden

Im Folgenden wird auf die Implementierung des Speichern und Ladens des zur Laufzeit erstellten Layouts der Objekte, sowie der Annotationen der Objekte eingegangen. Hierzu wird zunächst festgelegt welche konkreten Attribute der Objekte gespeichert werden. Darauf aufbauend wird eine Daten Objekt Klasse erstellt, welche nur die ausgewählten Attribute enthält. Diese zusätzliche Klasse wird benötigt, da nicht alle, insbesondere die von Unity hinzugefügten Datentypen serialisert werden können.[1] Abschließend wird auf die Implementierung der Klasse zur Serialisierung und Deserialisierung der Daten Objekte eingegangen.

4.2.1 Datenobjekt

Wie bereits in Abschnitt 4.2 angedeutet können nicht alle Datentypen serialisiert werden. Daraus resultiert das annotierbare Objekte nicht direkt serialisiert werden können. Daher wird eine Klasse AnnotatableObjektData für diese erstellt. Das Datenobjekt wandelt alle nicht serialisierbaren Datentypen in serialisierbare Datentypen um.
Das Daten Objekt muss einerseits ein Attribut Id haben. Dieses dient der eindeutigen Zuordnung von Daten Objekt zu annotierbaren Objekt. Andererseits muss die Position gespeichert werden. Hierbei muss beachtet werden, dass die Position in Unity als Vector3 angegeben wird. Dieses Datenformat kann nicht direkt serialisiert werden. Daher hat das Daten Objekt kein Vector3 Attribut sondern ein Array vom Datentyp float. Selbiges ist bei der Speicherung des Attributs Scale zu beachten. Zudem muss das Daten Objekt die Annoationen des annotierten Objekts abbilden. Diese werden hierbei in ein Array vom Datentyp String umgewandelt, da die Annotationen an sich ein GameObjekt sind und somit nicht direkt serialisiert werden können.

[1] *Binary serialization*, Zugriff: 24.10.2020, Jan. 2018. [Online]. Available: `https://docs.microsoft.com/de-de/dotnet/standard/serialization/binary-serialization`.

Der Konstruktor eines Datenobjekts benötigt nun ein annotierbares Objekt. Innerhalb des Konstruktors werden die zuvor genannten Attribute des annotierten Objekts wie beschrieben konvertiert, sodass ein serialisierbares Datenobjekt ensteht, welches alle relevanten Attribute des annotierten Objekts abbildet.

4.2.2 Serialisierung und Deserialisierung

Wie bererits in Abschnitt 4.2 erwähnten Bedarf das Serialisieren und Deserialisieren eine eigene Klasse. Diese bietet jeweils eine Funktion zum Serialisieren und Deserialisieren an. Beide Funktionen benötigen als Parameter einen Pfad, in dem die Datei gespeichert oder aus dem die Datei ausgelesen wird. Dieser Pfad wird analog zum initialen Erstellen einer Stadt, im Inspectormenü des Stadtobjekts festgelegt. Desweiteren benötigt die Serialisierungsfunktion eine Liste von zu serialsierenden Objekten. Da sowohl das automatische Speichern, als auch das Speichern per Knopfdruck innerhalb des Stadtobjekts angestoßen wird, kann die Funktion AllNoteDescendants() genutzt werden um eine Liste aller Objekte der Stadt zu erlangen. Unter Nutzung dieser Liste kann die Serialisierungsfunktion eine Liste von Datenobjekten erstellen, welche alle relevanten Eigenschaften der Objekte der Stadt abbildet. Diese wird anschließend mit Hilfe des Binaryformater des .NET Frameworks serialsiert und unter dem zuvor angegebenen Pfad gespeichert.
Die Deserialisierung benötigt keine weiteren Parameter außer dem Pfad der Datei. Die Deserialisierungsfunktion verwendet ebenfalls den Binaryformater und generiert so die zuvor gespeicherte Liste von Datenobjekten. Da die Deserialisierungsfunktion ebenfalls aus dem Stadtobjekt heraus aufgerufen wird, wird die Liste der Datenobjekte an das Stadtobjekt zurückgegeben. Mithilfe der Ids der Datenobjekte können die weiteren Attribute der Datenobjekte ihren entsprechenden annotierten Objekten zugeordnet werden, woraufhin die Attribute der Stadtobjekte angepasst werden.

KAPITEL 5

Evaluation

In dieser Arbeit soll evaluiert werden, ob sich die Benutzerfreundlichkeit und Gebrauchstauglichkeit von SEE durch die Möglichkeit der Angabe von zusätzlichen expliziten und impliziten Informationen verändert.

5.1 Planung

Die Veränderung der Benutzerfreundlichkeit und Gebrauchstauglichkeit soll anhand einer Nutzerstudie ermittelt werden. Die Nutzerstudie gliedert sich hierbei in zwei Durchläufe auf, wobei die Reihenfolge der Durchläufe alterniert. Innerhalb eines Durchlaufs werden die Probanden eine Softwarecity in Virtual Reality erkunden. Die Durchgänge unterscheiden sich hierbei darin, dass in einem Durchlauf die im Zuge dieser Arbeit erstellten Funktionen verfügbar sind, während sie im anderen Durchgang nicht zur Verfügung stehen. Die beiden Durchgänge basieren grundlegend auf der selben Softwarestadt, jedoch ist die Anordnung der Objekte verschieden. Softwarestadt-A verfügt hierbei über die in dieser Arbeit vorgestellten Funktionen zur expliziten und impliziten Informationsdarstellung, während Softwarestadt-B nicht über diese Funktionen verfügt. Anschließend an jeden Durchlauf werden die Probanden die Benutzerfreundlichkeit und Gebrauchstauglichkeit der Anwendung anhand eines Fragebogens bewerten.

5.1.1 Fragebogen

Der für die Nutzerstudie verwendete Fragebogen muss einigen Anforderungen genügen. Die Hauptanforderungen sind hierbei, dass der Fragebogen auch bei einer geringen Anzahl an Probanden belastbare Ergebnisse liefert. Zudem darf der Fragebogen nicht zu umfangreich sein. Des Weiteren muss der Fragebogen einigen Nebenanforderungen genügen. Der Fragebogen sollte kostenlos und frei zugänglich sein. Außerdem sollte eine validierte Version des Fragebogens in deutscher Sprache verfügbar sein. Der Grund hierfür ist, dass verschiedene Personen dem selben Wort einer Fremdsprache, wie beispielsweise Englisch, eine unterschiedliche Semantik zuordenen könnten. Dies kann zu einem grundsätzlich unterschiedlichem Verständnis

einer Frage führen, wodurch die Ergbnisse der Nutzerstudie verfälscht werden könnten.
Im Folgenden werden die Vorteile und Nachteile der im Verlauf dieser Arbeit betrachteten Fragebögen dargelegt. Auf Basis dieser Vor- und Nachteile wird anschließend ein Fragebogen ausgewählt.

5.1.1.1 System Usability Scale

Die System Usability Scale wurde 1996 von John Brooke entwickelt und ist ein zehn Fragen umfassender Fragebogen.[1] Die Antworten der Fragen basieren hierbei auf einer fünfstufigen Likert Skala. Die Verlässlichkeit der System Usability Scale wurde seit ihrer Entwicklung mehrfach überprüft und es wurde festgestellt, dass sie belastbare und vergleichbare Ergebnisse liefert. Desweiteren wurde festgestellt, dass die System Usability Scale auch bei einer geringen Anzahl von Probanden zuverlässig ist.[2][3]
Die System Usability Scale wurde ursprünglich in englischer Sprache erstellt. Eine deutsche Übersetzung wurde 2013 in einem Crowdsourcing-Projekt von Dr. Wolfgang Reinhardt erarbeitet und in Zusammenarbeit mit SAP veröffentlicht.[4] Um die Validität der deutschen Übersetzung sicherzustellen wurde sie von mehreren englischen Muttersprachlern zurück übersetzt.[5]

5.1.1.2 Usability Metric for User Experience

Der Fragebogen Usabality Metric for User Experience oder UMUX wurde 2010 von Kraig Finstad vorgestellt.[6] UMUX basiert grundlegend auf der System Usability Scale und hat als Ziel einen kürzeren aber doch validen Fragebogen darzustellen. UMUX besteht aus Vier Fragen welche anhand einer siebenstufigen Likert Skala beantwortet werden. Die Validität wurde anhand der Korrelation zur System Usability Scale ermittelt, wobei eine Korrelation von über 0,80 erreicht wurde, somit scheint UMUX valide zu sein.[7] Zur Anwendung bei einer

[1] J. Brooke, "SUS: A 'quick and dirty' usability scale," in *Usability Evaluation In Industry*, CRC Press, Jun. 1996, pp. 207–212. DOI: 10.1201/9781498710411-35. [Online]. Available: https://doi.org/10.1201/9781498710411-35.

[2] T. S. Tullis and J. N. Stetson, "Upa 2004 presentation - a comparison of questionnaires for assessing website usability," 2004, S.6.

[3] J. Brooke, "Sus: A retrospective," *Journal of Usability Studies*, vol. 8, pp. 29–40, Jan. 2013, S.33.

[4] M. Thielsch and M. Salaschek, "Toolbox zur kontinuierlichen website-evaluation und qualitätssicherung," de, 2018. DOI: 10.17623/BZGA:224-2.1. [Online]. Available: https://www.bzga.de/fileadmin/user_upload/forschung/cahpot/bzga_toolbox_website-evaluation_21--c44487b3ba3050661c45e04fb1648ffe.pdf, S.36.

[5] B. Rummel, *System usability scale – jetzt auch auf deutsch*, Zugriff: 20.10.2020, Jan. 2015. [Online]. Available: https://experience.sap.com/skillup/system-usability-scale-jetzt-auch-auf-deutsch/.

[6] K. Finstad, "The usability metric for user experience," *Interacting with Computers*, vol. 22, no. 5, pp. 323–327, Sep. 2010. DOI: 10.1016/j.intcom.2010.04.004. [Online]. Available: https://doi.org/10.1016/j.intcom.2010.04.004.

[7] K. Finstad, "The usability metric for user experience," *Interacting with Computers*, vol. 22, no. 5, pp. 323–327, Sep. 2010. DOI: 10.1016/j.intcom.2010.04.004. [Online]. Available: https://doi.org/10.1016/

geringen Probandenzahl sind keine Studien auffindbar. Desweiteren existiert keine validierte deutsche Übersetzung.

5.1.1.3 User Experience Questionnaire

Die ursprüngliche Version des User Experience Questionnaire wurde 2005 in deutscher Sprache entwickelt.[8] Der Fragebogen hat 26 verschiedene Fragen, die in Sechs Unterkategorien geordnet sind. Die Fragen werden anhand einer siebenstufigen Likert Skale beantwortet. Die Validität wurde für die deutsche Version anhand von zwei Studien bestimmt.[9] Zur Anwendung des User Experience Questionnaire bei kleineren Probandenzahlen sind keine weiteren Studien auffindbar.

5.1.1.4 Auswahl Fragebogen

Auf Basis der zuvor erläuterten Merkmale der vorgestellten Fragebögen wird im restlichen Verlauf dieser Arbeit mit der System Usabilty Scale gearbeitet. Dies ist dadurch begründet, dass die System Usability Scale alle Hauptanforderungen an den Fragebogen erfüllt. Zudem ist die System Usabality Scale in deutscher Sprache frei zugänglich und validiert. Somit erfüllt die System Usability Scale alle zuvor aufgestellten Anforderungen und hat im Vergleich einen mittleren Fragenumfang.
Der verwendete Fragebogen ist in Abbildung 5.1 zu sehen.

5.2 Durchführung

Die Nutzerstudie wurde mit Acht Probanden durchgeführt. Von diesen Acht Probanden sind Sieben Studenten im Fachbereich Informatik. Die Praxiserfahrung der Probanden in der Softwareentwicklung ist unterschiedlich. Einige Probanden haben bereits teilweise mehrjährige Erfahrung in der Industrie gesammelt, während andere keine weitere Erfahrung neben ihrem universitären Wandel haben. Der achte Proband hat eine abgeschlossene Ausbildung als Fachinformatiker und ist seit mehreren Jahren in der Softwareentwicklung tätig. Die Erfahrung der Probanden mit Virtual Reality unterscheidet sich ebenfalls. Einerseits hatten einige Probanden zuvor keine Erfahrungen mit Virtual Reality. Andererseits hatte ein Proband ei-

j.intcom.2010.04.004, S.326.

[8] B. Laugwitz, M. Schrepp, and T. Held, "Konstruktion eines fragebogens zur messung der user experience von softwareprodukten," in *Mensch und Computer 2006*, Oldenbourg Wissenschaftsverlag, Sep. 2006. DOI: 10.1524/9783486841749.125. [Online]. Available: https://doi.org/10.1524/9783486841749.125.

[9] B. Laugwitz, M. Schrepp, and T. Held, "Konstruktion eines fragebogens zur messung der user experience von softwareprodukten," in *Mensch und Computer 2006*, Oldenbourg Wissenschaftsverlag, Sep. 2006. DOI: 10.1524/9783486841749.125. [Online]. Available: https://doi.org/10.1524/9783486841749.125, S.132f.

Fragebogen zur System-Gebrauchstauglichkeit

1. Ich denke, dass ich das System gerne häufig benutzen würde.

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

2. Ich fand das System unnötig komplex.

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

3. Ich fand das System einfach zu benutzen.

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

4. Ich glaube, ich würde die Hilfe einer technisch versierten Person benötigen, um das System benutzen zu können.

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

5. Ich fand, die verschiedenen Funktionen in diesem System waren gut integriert.

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

6. Ich denke, das System enthielt zu viele Inkonsistenzen.

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

7. Ich kann mir vorstellen, dass die meisten Menschen den Umgang mit diesem System sehr schnell lernen.

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

8. Ich fand das System sehr umständlich zu nutzen.

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

9. Ich fühlte mich bei der Benutzung des Systems sehr sicher.

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

10. Ich musste eine Menge lernen, bevor ich anfangen konnte das System zu verwenden.

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

Abbildung 5.1: Fragebogen der Nutzerstudie
10

nige Erfahrungen mit Virtual Reality basierten Videospielen und ein Proband hatte bereits Erfahrungen mit SEE in Virtual Reality im Kontext einer Vorstellung des Projekts.
Die Nutzerstudie wurde aufgrund der im Zusammenhang mit der Corona Pandemie verhängten Zugangsbeschränkungen der Universität, in einem Raum meiner Wohnung durchgeführt. Die Virtual Reality Ausstattung, die für die Studie nötig war wurde zuvor von der Universität ausgeliehen. Die Probanden nahmen einzeln an der Studie teil. Alle Probanden hatten zuvor keinen Einblick in die im Kontext dieser Arbeit erstellten Funktionen, um eine Verfälschung der Ergebnisses zu verhindern.
Vor Beginn der tatsächlichen Nutzerstudie erhielten die Probanden eine kurze einführende Erklärung. Diese Erklärung umfasste zum einen das grundlegende Konzept von Softwarestädten. Desweiteren wurde den Probanden erklärt, welche Attribute der Objekte der Softwarestadt von welchen Softwaremetriken abhängig sind. Um Missverständnisse zu vermeiden wurden die Probanden anschließend gefragt, ob sie den Erklärungen folgen konnten und etwaige Rückfragen wurden beantwortet.
Nachdem die grundlegende Erklärung und Beantwortung von Nachfragen abgeschlossen ist, wurden die Probanden in den Raum mit dem Virtual Reality Equipment geführt. Anschließend wurde den Probanden anhand der Controller der HTC Vive, sowie einer Abbildung der Tastenbelegung, die Steuerung für den jeweiligen Durchlauf erklärt. Nach Abschluss der Erklärungen wurden die Probanden erneut gefragt, ob sie alles verstanden haben und weitere Rückfragen wurden beantwortet.
Wenn alle Rückfragen geklärt waren setzte der Proband das Head-Mounted-Display auf und der erste Durchgang begann. Wie zuvor erwähnt begann eine Hälfte der Probanden mit Softwarestadt-A, während die andere Hälfte mit Softwarestadt-B begann. Während des Durchgangs konnte der Proband alle Funktionen die zur Verfügung standen nutzen und so eine Softwarestadt frei erkunden. Nachdem der Proband fertig war füllte er den zuvor vorgestellten Fragebogen zur Ermittlung der Benutzerfreundlichkeit und Gebrauchstauglichkeit aus. Anschließend wurde dem Probanden die Steuerung für seinen zweiten Durchlauf, analog zum ersten Durchgang, erklärt. Darauf folgend begann der zweite Durchgang. Nachdem auch der zweite Durchgang beendet war füllte der Proband den Fragebogen ein zweites Mal aus.

5.3 Ergebnisse

Im folgenden werden zunächst die Fragebögen der Probanden ausgewertet. Anschließend wird ermittelt, ob die Veränderung der Benutzerfreundlichkeit und Gebrauchstauglichkeit statistisch signifikant ist. Hierzu wird der Wilcoxon Vorzeichen-Rang-Test verwendet.

5.3.1 Auswertung Fragebögen

In den folgenden Tabellen werden zunächst die unverarbeiteten Daten aus den Fragebögen dargestellt.

Tabelle 5.1: Softwarestadt-A Fragebögen

	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10
P1	4	1	4	1	4	1	5	1	5	1
P2	3	4	3	3	4	2	4	4	3	2
P3	2	4	3	1	4	3	3	3	4	1
P4	3	2	2	2	3	3	2	3	2	4
P5	4	2	4	2	4	3	3	3	4	4
P6	5	2	4	3	4	3	4	2	4	3
P7	4	2	4	4	3	2	5	4	3	2
P8	4	2	4	3	4	2	4	2	4	3

Tabelle 5.2: Softwarestadt-B Fragebögen

	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10
P1	3	2	4	1	2	1	5	1	4	1
P2	4	3	3	3	3	2	3	2	4	3
P3	2	4	3	1	3	3	3	3	3	1
P4	2	2	3	3	2	2	2	2	3	3
P5	3	3	2	4	2	3	2	3	4	4
P6	4	3	2	2	4	3	4	3	4	3
P7	2	2	4	5	2	3	5	3	2	1
P8	3	2	4	2	2	3	3	4	3	3

Die Auswertung der Rohdaten wird genau von der System Usability Scale vorgegeben.[11] Zunächst müssen die Punkte aller geraden Fragen invertiert werden. Dies hat den Grund, dass alle geraden Fragen negativ gestellt sind. Um die Punktzahlen zu invertieren müssen diese von Fünf subtrahiert werden. Desweiteren muss von allen Punktzahlen der ungeraden Fragen Eins subtrahiert werden. Daraus folgend haben alle Fragen eine Punktzahl zwischen Null und Vier. Im Anschluss werden diese Punktzahlen summiert, somit erhält man eine Gesamtpunktzahl zwischen Null und 40. Diese Gesamtpunktzahl wird darauf folgend mit 2,5 multipliziert. Das Ergebnis daraus liegt nun zwischen Null und 100. Dies ist nun die abschließende Punktzahl nach der System Usability Scale. In Tabelle 5.3 sind die Ergebnisse für die einzelnen Probanden dargestellt. Desweiteren enthält die letzte Zeile der Tabelle 5.3 das arithmetische Mittel für die beiden Städte.

[11] J. Brooke, "SUS: A 'quick and dirty' usability scale," in *Usability Evaluation In Industry*, CRC Press, Jun. 1996, pp. 207–212. DOI: 10.1201/9781498710411-35. [Online]. Available: https://doi.org/10.1201/9781498710411-35, S.211.

Tabelle 5.3: Ergebnisse Nutzerstudie

Proband	Softwarestadt-A	Softwarestadt-B
P1	92,5	80
P2	55	60
P3	60	55
P4	45	50
P5	62,5	40
P6	70	60
P7	62,5	52,5
P8	70	52,5
	64,7	**56,3**

Wie in Tabelle 5.3 zu sehen ist, ist die Benutzerfreundlichkeit durch die in dieser Arbeit hinzugefügten Funktionen gestiegen. Um nun nachzuweisen, dass der Anstieg der Benutzerfreundlichkeit statistisch signifikant ist, wird der Wilcoxon-Vorzeichen-Rang-Test angewandt.

5.3.2 Wilcoxon Vorzeichen-Rang-Test

Der Wilcoxon Vorzeichen-Rang-Test wurde 1945 von Frank Wilcoxon vorgestellt und ist ein nicht parametrischer Test, welcher die statistische Signifikanz einer Differenz zwischen Zwei Stichproben prüft.[12] Um den Wilcoxon-Vorzeichen-Rang-Test anzuwenden muss zunächst eine Nullhypothese H_0 formuliert werden. In diesem Fall ist die Nullhypothese H_0: Die Benutzerfreundlich und Gebrauchstauglichkeit ist durch die hinzugefügten Funktionen gesunken oder gleich geblieben. Die Gegenhypothese H_1 ist demnach: Die Benutzerfreundlichkeit und Gebrauchstauglichkeit ist durch die hinzugefügten Funktionen gestiegen. Die gewählte Konfizend beträgt hierbei 90%. Mithilfe der Anzahl an Teilnehmern n und der gewählten Konfidenz von 90% kann nun aus der Tabelle zum Wilcoxon-Vorzeichen-Rang-Test der kritische Wert abgelesen werden. Unter den zu Grunde liegenden Parametern beträgt der kritische Wert Acht. Die nun zu errechnende Teststatistik W muss nun kleiner oder gleich Acht sein, um die Nullhypothese H_0 abzulehnen.
Um mit dem eigentlichen Test zu beginnen werden zunächst die Differenzen zwischen der Softwarestadt-A und Softwarestadt-B pro Proband gebildet. Die errechneten Differenzen sind in Tabelle 5.4 zu sehen.

[12] F. Wilcoxon, "Individual comparisons by ranking methods," *Biometrics Bulletin*, Dec. 1945. DOI: 10.2307/3001968. [Online]. Available: https://doi.org/10.2307/3001968.

Tabelle 5.4: Vorzeichen-Rang-Test Differenzen

Proband	Punktzahl-A	Punktzahl-B	Differenz	Rang
P1	92,5	80	12,5	6
P2	55	60	-5	2
P3	60	55	5	2
P4	45	50	-5	2
P5	62,5	40	22,5	8
P6	70	60	10	4,5
P7	62,5	52,5	10	4,5
P8	70	52,5	17,5	7

Anschließend werden den Differenzen Ränge zugeordnet. Hierbei werden die Vorzeichen nicht beachtet und es gilt, umso niedriger die Differenz, desto niedriger der Rang. Desweiteren muss beachtet werden, dass bei einer gleichen Differenzen das Mittel der Ränge an die Differenzen vergeben wird.
Im nächsten Schritt werden die Rangsummen errechnet. Hierzu werden die Ränge aller negativen Differenzen aufsummiert. Dies ergibt die Rangsumme W^-. Das selbe Vorgehen wird für alle positiven Differenzen angewandt, wodurch man die zweite Rangsumme W^+ erhält. Eine geordnete Darstellung der Ränge, sowie die errechneten Rangsummen sind in Tabelle 5.5 abgebildet.

Tabelle 5.5: Vorzeichen-Rang-Test Rangsummen

Differenz	Rang	W^+	W^-
-5	2		2
-5	2		2
5	2	2	
10	4,5	4,5	
10	4,5	4,5	
12,5	6	6	
17,5	8	7	
22,5	8	8	
		32	**4**

Nachdem sowohl die negative, als auch die positive Rangsumme errechnet wurde kann die Teststatistik W gebildet werden. Um die Teststatistik W zu bilden wird das Minimum der Rangsummen ermittelt.

$$W = min(W^-, W^+) \tag{5.1}$$

Für die Nutzerstudie ergibt sich somit eine Teststatistik von $W = 4$. Daraus resultierend unterschreitet die Teststatistik den kritischen Wert von Acht. Dies führt dazu, dass die Nullhypothese H_0 mit einer Konfidenz von 90% abgelehnt werden kann und die Gegenhypothese H_1 angenommen wird. Somit ist der Anstieg der Benutzerfreundlichkeit und Gebrauchstauglichkeit durch die im Verlauf dieser Arbeit hinzugefügten Funktionen, zum Annotieren und Umgestalten von Softwarestädten, statistisch signifikant.

5.3.3 Fazit

In der Nutzerstudie zu dieser Arbeit konnte gezeigt werden, dass die Benutzerfreundlichkeit und Gebrauchstauglichkeit von SEE durch das Hinzufügen von Funktionen zum Umgestalten und Annotieren von Software-Städten gestiegen ist. Desweiteren konnte gezeigt werden, dass der Anstieg mit einer Konfidenz von 90% statistisch signifikant ist.

KAPITEL 6

Ausblick

Im Verlauf dieser Arbeit sind weitere Funktionen und Fragestellungen aufgekommen, welche zukünfig interessant sein könnten. In den folgenden Abschnitten werden diese Funktionen und Fragestellungen erläutert.

6.1 Menü

Das Erstellen und Konfigurieren einer Stadt wird aktuell über das Inspector Menü von Unity gesteuert. Selbiges gilt für das Laden einer zuvor bearbeiteten Softwarestadt, welches im Zuge dieser Arbeit hinzugefügt wurde. Insbesondere unter der Verwendung eines Head-Mounted-Displays und Virtual Reality kann dies störend sein, da zur Bedienung des Inspector Menüs das Head-Mounted-Displays abgelegt werden muss. Eine mögliche Alternative hierzu wäre, ein Menü für diese Einstellungen zu erstellen, welches zum Start von SEE aufgerufen wird.

6.2 Virtuelle Tastaturen

Eine weitere Funktion die insbesondere für ein Menü, wie in Abschnitt 6.1 vorgeschlagen wichtig wäre, ist eine virtuelle Tastatur. Diese wäre nötig um beispielsweise den Pfad, aus welchem Datein geladen werden müssen anzugeben. Des Weiteren wäre eine virtuelle Tastatur, als eine zusätzliche Eingabemöglichkeit zur Diktierfunktion, für das Erstellen von Annotationen interessant. Hierzu könnte eine vergleichende Nutzerstudie zu verschiedenen Arten von virtuellen Tastaturen wie beispielsweise eine Drumtastatur oder gestengesteuerten Tastatur und ihrer Gebrauchstauglichkeit erstellt werden.

Abbildung 6.1: Drumtastatur
C. Boletsis and S. Kongsvik, "Text input in virtual reality: A preliminary evaluation of the drum-like VR keyboard," *Technologies*, vol. 7, no. 2, Apr. 2019. DOI: 10.3390/technologies7020031. [Online]. Available: https://doi.org/10.3390/technologies7020031

6.3 Legende

Ein Proband der Nutzerstudie hatte die Idee eine Legende für die verschiedenen Metriken der Objekte hinzuzufügen. Insbesondere unter der Verwendung eines Head-Mounted-Displays und Virtual Reality könnte dies sinnvoll sein. Dies ist dadurch begründet, dass im Gegensatz zur Verwendung einer Desktop Umgebung nicht einfach im Inspector Menü von Unity geprüft werden kann welche Metrik, wie abgebildet wird. Eine mögliche Form der Darstellung wären die Achsen eines dreidimensionalen Koordinatensystem, welche mit den entsprechenden Metriken beschriftet sind.

ABBILDUNGSVERZEICHNIS

LISTINGS

[1] *Basic serialization*, Zugriff: 24.10.2020, Mar. 2017. [Online]. Available: `https://docs.microsoft.com/de-de/dotnet/standard/serialization/basic-serialization`.

[2] *Basic serialization*, Zugriff: 24.10.2020, Mar. 2017. [Online]. Available: `https://docs.microsoft.com/de-de/dotnet/standard/serialization/basic-serialization`.

Literaturverzeichnis

[1] C. Knight, "Visualisation for program comprehension: Information and issues," Jul. 1999. [Online]. Available: https://www.researchgate.net/publication/2464933_Visualisation_for_Program_Comprehension_Information_and_Issues.

[2] R. Wettel and M. Lanza, "Visualizing software systems as cities," Jun. 2007. DOI: 10.1109/vissof.2007.4290706. [Online]. Available: https://doi.org/10.1109/vissof.2007.4290706.

[3] J. Steuer, "Defining virtual reality: Dimensions determining telepresence," *Journal of Communication*, vol. 42, no. 4, pp. 73–93, Dec. 1992. DOI: 10.1111/j.1460-2466.1992.tb00812.x. [Online]. Available: https://doi.org/10.1111%2Fj.1460-2466.1992.tb00812.x.

[4] F. Shu, W. Mi, and Z. Xu, "The information sharing platform for port container terminal logistics using virtual reality," in *2007 IEEE International Conference on Automation and Logistics*, 2007, pp. 2570–2575. DOI: 10.1109/ICAL.2007.4339013.

[5] T. Kesztyüs, M. Mehlitz, E. Schilken, G. Weniger, S. Wolf, U. Piccolo, E. Irle, and O. Rienhoff, "Preclinical evaluation of a virtual reality neuropsychological test system: Occurrence of side effects," *CyberPsychology & Behavior*, vol. 3, Jun. 2000. DOI: 10.1089/10949310050078788.

[6] *Über das vive headset*, Zugriff: 09.10.2020. [Online]. Available: https://www.vive.com/de/support/vive/category_howto/about-the-headset.html.

[7] *Vive controller*, Zugriff: 10.10.2020. [Online]. Available: https://www.vive.com/media/filer_public/support_zip_img/de/www/vive/guid-2d5454b7-1225-449c-b5e5-50a5ea4184d6-web.png.

[8] *Multiplatform*, Zugriff: 11.10.2020. [Online]. Available: https://unity.com/de/features/multiplatform.

[9] *Unity physics*, Zugriff: 11.10.2020. [Online]. Available: https://docs.unity3d.com/Manual/PhysicsSection.html.

[10] *Scripting*, Zugriff: 11.10.2020. [Online]. Available: https://docs.unity3d.com/Manual/ScriptingSection.html.

[11] C. Boletsis and S. Kongsvik, "Text input in virtual reality: A preliminary evaluation of the drum-like VR keyboard," *Technologies*, vol. 7, no. 2, Apr. 2019. DOI: 10.3390/technologies7020031. [Online]. Available: https://doi.org/10.3390/technologies7020031.

[12] C. Yu, Y. Gu, Z. Yang, X. Yi, H. Luo, and Y. Shi, "Tap, dwell or gesture?" In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, ACM, May 2017. DOI: 10.1145/3025453.3025964. [Online]. Available: https://doi.org/10.1145/3025453.3025964.

[13] *Dictationrecognizer*, Zugriff: 17.10.2020. [Online]. Available: https://docs.unity3d.com/ScriptReference/Windows.Speech.DictationRecognizer.html.

[14] *Binary serialization*, Zugriff: 24.10.2020, Jan. 2018. [Online]. Available: https://docs.microsoft.com/de-de/dotnet/standard/serialization/binary-serialization.

[15] *Binaryformatter security guide*, Zugriff: 24.10.2020, Jul. 2020. [Online]. Available: https://docs.microsoft.com/de-de/dotnet/standard/serialization/binaryformatter-security-guide.

[16] *Basic serialization*, Zugriff: 24.10.2020, Mar. 2017. [Online]. Available: https://docs.microsoft.com/de-de/dotnet/standard/serialization/basic-serialization.

[17] *Json serialization and deserialization (marshalling and unmarshalling) in .net - overview*, Zugriff: 24.10.2020, Jan. 2020. [Online]. Available: https://docs.microsoft.com/de-de/dotnet/standard/serialization/system-text-json-overview.

[18] J. Brooke, "SUS: A 'quick and dirty' usability scale," in *Usability Evaluation In Industry*, CRC Press, Jun. 1996, pp. 207–212. DOI: 10.1201/9781498710411-35. [Online]. Available: https://doi.org/10.1201/9781498710411-35.

[19] T. S. Tullis and J. N. Stetson, "Upa 2004 presentation - a comparison of questionnaires for assessing website usability," 2004.

[20] J. Brooke, "Sus: A retrospective," *Journal of Usability Studies*, vol. 8, pp. 29–40, Jan. 2013.

[21] M. Thielsch and M. Salaschek, "Toolbox zur kontinuierlichen website-evaluation und qualitätssicherung," de, 2018. DOI: 10.17623/BZGA:224-2.1. [Online]. Available: https://www.bzga.de/fileadmin/user_upload/forschung/cahpot/bzga_toolbox_website-evaluation_21--c44487b3ba3050661c45e04fb1648ffe.pdf.

[22] B. Rummel, *System usability scale – jetzt auch auf deutsch*, Zugriff: 20.10.2020, Jan. 2015. [Online]. Available: https://experience.sap.com/skillup/system-usability-scale-jetzt-auch-auf-deutsch/.

[23] K. Finstad, "The usability metric for user experience," *Interacting with Computers*, vol. 22, no. 5, pp. 323–327, Sep. 2010. DOI: 10.1016/j.intcom.2010.04.004. [Online]. Available: https://doi.org/10.1016/j.intcom.2010.04.004.

[24] B. Laugwitz, M. Schrepp, and T. Held, "Konstruktion eines fragebogens zur messung der user experience von softwareprodukten," in *Mensch und Computer 2006*, Oldenbourg Wissenschaftsverlag, Sep. 2006. DOI: 10.1524/9783486841749.125. [Online]. Available: https://doi.org/10.1524/9783486841749.125.

[25] B. Rummel and E. Ruegenhagen, *Fragenbogen zur system-gebrauchstauglichkeit*, Zugriff: 20.10.2020, 2013. [Online]. Available: https://experience.sap.com/files/System_Usability_Scale_A4_DE.doc.

[26] F. Wilcoxon, "Individual comparisons by ranking methods," *Biometrics Bulletin*, Dec. 1945. DOI: 10.2307/3001968. [Online]. Available: https://doi.org/10.2307/3001968.

[27] F. Sani and J. Todman, Eds., *Experimental Design and Statistics for Psychology*. Blackwell Publishing Ltd, Jan. 2006. DOI: 10.1002/9780470776124. [Online]. Available: https://doi.org/10.1002/9780470776124.

[28] *Über die vive controller*, Zugriff: 09.10.2020. [Online]. Available: https://www.vive.com/de/support/vive/category_howto/about-the-controllers.html.

Zusätzlicher Anhang

	level of significance for a one-tailed test					
	.10	.05	.025	.01	.005	.001
	level of significance for a two-tailed test					
N	.20	.10	.05	.02	.01	.002
4	0					
5	2	0				
6	4	2	0			
7	6	3	2	0		
8	8	5	3	1	0	
9	11	8	5	3	1	
10	14	10	8	5	3	0
11	18	13	10	7	5	1
12	22	17	14	10	7	2
13	26	21	17	12	9	4
14	31	25	21	15	12	6
15	37	30	25	19	16	8
16	42	35	29	23	19	11
17	49	41	35	28	23	14
18	55	47	40	32	27	18
19	62	53	46	37	32	21
20	70	60	52	43	37	23
21	78	67	58	50	44	32
22	86	75	65	55	47	34
23	95	83	73	62	54	40
24	104	91	81	69	61	46
25	115	101	89	76	68	51
26	124	110	98	84	75	58
27	135	119	107	93	83	63
28	146	130	116	101	91	71
29	157	141	126	111	101	80
30	169	152	136	119	106	85
31	181	163	147	129	118	95
32	195	175	159	140	127	102
33	208	187	170	151	137	111
34	221	200	183	163	149	123
35	236	213	195	175	159	130
36	250	227	208	185	171	139
37	266	241	221	198	182	153
38	281	256	234	211	194	163
39	297	271	249	224	209	176
40	314	286	264	238	219	186

Wilcoxon Vorzeichen-Rank-Test kritische Werte
[3]