

Bearbeitungsbeginn: [01.09.2018]

Vorgelegt am: [26.02.2019]

Thesis

zur Erlangung des Grades

Master of Science

im Studiengang Medieninformatik

an der Fakultät Digitale Medien

Felix Gähr

Matrikelnummer: 256344

**Konzeption und Umsetzung eines Toolkits für
wissenschaftliche Experimente in VR-Umgebungen**

Erstbetreuer: Prof. Christoph Müller

Zweitbetreuer: Prof. Dr.-Ing. Matthias Wölfel

Abstract

In der vorliegenden Masterarbeit wird ein wissenschaftliches Toolkit konzipiert, welches Wissenschaftler dabei unterstützt, Experimente in Virtual Reality-Umgebungen durchzuführen. Es handelt sich bei der Software um ein Plugin für die Unity-Entwicklungsumgebung. Das Toolkit erweitert die Funktionen Unitys um Features wie Steuerungselemente und ein Eventsystem. Mithilfe des Toolkits kann ein Anwender ein Experiment in Unity strukturieren und später dessen Ablauf in Echtzeit über ein automatisch generiertes Interface steuern. Das Toolkit enthält außerdem ein Eventsystem, welches Ereignisse während eines Experimentes aufzeichnet. Die aufgezeichneten Daten werden ins JSON-Format konvertiert und in einer Textdatei gesichert. Diese Daten können von Statistiksoftware eingelesen und verarbeitet werden. Des Weiteren können die JSON-Daten vom Toolkit in Unity wieder eingelesen werden, um den Ablauf eines Experimentes wiederherzustellen. Das Toolkit ermöglicht außerdem den Anschluss von externen Geräten und die Einbindung von Daten, die von diesen Geräten übertragen werden.

Im Anschluss an das Konzept wird dessen praktische Umsetzung und der technische Aufbau beschrieben. Das Toolkit wurde in Kooperation mit einer anderen Masterarbeit in einem wissenschaftlichen Probandentest, sowie in einem Pseudoexperiment eingesetzt, um die verschiedenen Funktionen des Toolkits auf die Probe zu stellen. Hierbei konnte das Toolkit erfolgreich dazu eingesetzt werden, den Ablauf des Experimentes zu steuern und relevante Daten zu sichern. Die aus den Experimenten gewonnenen Erkenntnisse wurden zur Evaluation der verschiedenen Aspekte des Toolkits eingesetzt.

Inhaltsverzeichnis

Abstract	I
Inhaltsverzeichnis	II
Abkürzungsverzeichnis	V
1 Einleitung	1
2 Grundlagen	3
2.1 Virtual Reality	3
2.1.1 Immersion und Präsenz	3
2.1.2 Ausgabe- und Eingabemethoden	4
2.2 Datenerhebung und Verarbeitung	7
2.2.1 Daten in wissenschaftlichen Experimenten	7
2.2.2 Datenvorbereitung	8
2.2.3 Data Mining	9
2.2.4 Datenspeicherung	10
2.2.5 Datenvisualisierung	10
3 Betrachtung bereits vorhandener Toolkits	12
3.1 ViSTA Virtual Reality Toolkit	12
3.2 Unity Analytics	13
3.3 Schlussfolgerungen	14
4 Konzeption des Toolkits	15
4.1 Zielsetzung	15
4.1.1 Kriterien	15
4.2 Konzept	17
4.2.1 Module	17
4.2.1.1 Testaufbau	17

4.2.1.2 Telemetrie	19
4.2.1.3 VR-Integration	22
4.2.1.4 Visualisierung	23
4.2.1.5 Wiedergabe	25
4.2.1.6 Interface	25
4.3 Bedienungshilfen	28
4.2.1 Beispielszene	28
4.2.2 Quickstart-Anleitung	29
4.4 Technischer Aufbau	30
4.4.1 Klassen	30
4.4.1.1 Klassen ohne Modulzugehörigkeit	30
4.4.1.2 Testaufbaumodul	32
4.4.1.3 Telemetriemodul	34
4.4.1.4 VR-Integrationsmodul	37
4.4.1.5 Wiedergabemodul	38
4.4.1.6 Interfacemodul	40
4.4.2 Aufbau des Visualisierungsmoduls	44
4.5 Aufbau des JSON-Formats	45
5 Umsetzung des Toolkits	47
5.1 ReceiveSerial Klasse	47
5.2 Feststellen welche Objekte betrachtet werden	48
5.3 Performance beim Erstellen der JSON-Strings	49
5.4 Aufzeichnung der Temperaturdaten	50
5.5 Codedokumentation	51
6 Evaluation des Toolkits	52
6.1 Methodik	52

6.2 Einsatz des Toolkits in einem Probandentest	52
6.2.1 Einbau des Toolkits in die Testsoftware	53
6.2.2 Durchführung des Probandentests.....	55
6.2.3 Befragung nach Durchführung des Probandentests.....	56
6.2.4 Auswertung der Testdaten.....	61
6.3 Einsatz des Toolkits in einem Pseudoexperiment.....	63
6.3.1 Einbau des Toolkits in die Testszene	63
6.3.2 Befragung nach Durchführung des Pseudoexperiments.....	64
6.4 Erfüllung der Kriterien.....	67
7 Fazit	69
Literaturverzeichnis	- 1 -
Abbildungsverzeichnis	- 5 -
Eidesstattliche Erklärung.....	- 7 -
Eingesetzte Software.....	- 8 -
Inhalt des Datenträgers	- 8 -
Fragenkatalog.....	- 9 -
Quickstart-Anleitung für das VR Scientific Toolkit.....	- 11 -

Abkürzungsverzeichnis

HMD – Head-Mounted Display

JSON – JavaScript Object Notation

VR – Virtual Reality

1 Einleitung

Die Erschaffung einer virtuellen Welt, die von der echten kaum noch zu unterscheiden ist, ist eine Idee, die schon viele Menschen fasziniert hat. Das zeigt sich in Science-Fiction Filmen wie „The Matrix“ oder in tatsächlichen Versuchen, über Technologien wie Head-Mounted Displays, eine solche Realität zu erschaffen. Zunehmend nähert sich dieser Traum nun der Wirklichkeit an. VR-Systeme stehen nicht mehr nur in Forschungslaboren, sondern finden sich immer mehr auch in Privathaushalten und in der Industrie (Jerald, 2016, S. 12). Beispielsweise wuchs auf der Spieleplattform Steam der Anteil von Nutzern mit VR-Systemen im Januar 2019 um ca. 12% im Vergleich zum Vormonat, was dazu führte, dass mit knapp 1% der Nutzer erstmals mehr VR-Nutzer auf Steam aktiv waren als Nutzer von Linux-Betriebssystemen (Valve Corporation, 2019). Im Jahr 2018 haben laut einer repräsentativen Bitkom-Umfrage zwar erst rund 16% der Deutschen VR schon einmal ausprobiert - eine leichte Erhöhung im Vergleich zu den 13% des Vorjahres - aber weitere 17% der Befragten können sich vorstellen, in Zukunft Virtual Reality zu nutzen. Dies weist auf ein hohes Wachstumspotential des VR-Marktes hin (Bitkom e.V., 2019).

Mit dem Aufkommen der VR-Systeme für Konsumenten werden in rasanter Geschwindigkeit neue Technologien für VR entwickelt (Jerald, 2016, S. 483). Beispielweise erscheint mit der Vive Pro Eye in Kürze ein kommerzielles VR System, welches die Augenbewegungen des Nutzers aufzeichnen kann (HTC, 2019). Solche Produkte sind nicht nur für Konsumenten interessant. Für Wissenschaftler schaffen die raschen Entwicklungen der VR-Technologie und des Marktes immer wieder neue Möglichkeiten, neue Technologien und deren Wirkung auf den Menschen zu untersuchen. Außerdem bietet VR für Experimente die einzigartige Möglichkeit, Probanden in Umgebungen und Situationen zu versetzen, die in der realen Welt gar nicht existieren.

In dieser Arbeit werden die Konzeption und Erstellung einer Software beschrieben, welche die Umsetzung solcher VR-Experimente vereinfacht. Das hier entstandene Toolkit soll es Wissenschaftlern ermöglichen, ein Experiment

in der Multiplattform-Entwicklungsumgebung Unity ohne großen Aufwand zu strukturieren und relevante Forschungsdaten aufzuzeichnen. Hierzu werden zunächst relevante theoretische Grundlagen zu Virtual Reality und dem Umgang mit Daten in der Wissenschaft aufgeführt. Anschließend wird bereits vorhandene Software betrachtet, welche bereits Funktionen besitzt, die für ein wissenschaftliches Toolkit für VR nützlich wären. Hier soll die Frage beantwortet werden, ob es sinnvoll wäre, ein Toolkit auf einer bereits bestehenden Software aufzubauen bzw. diese zu ergänzen. Im nächsten Kapitel wird die Konzeption des Toolkits beschrieben. Es werden Kriterien definiert, welche Ziele repräsentieren, die das Toolkit erreichen soll. Im Konzept wird der Aufbau des Toolkits und seine Funktionen, wie ein Eventsystem, eine Teststeuerung und die Wiedergabe von gespeicherten Experimenten, erläutert. Es wird außerdem die technische Struktur des Toolkits dokumentiert. Im Kapitel „Umsetzung“ wird auf Besonderheiten und Schwierigkeiten eingegangen, die bei der praktischen Umsetzung des Toolkits entstanden sind.

Im fünften Kapitel wird das Toolkit schließlich evaluiert. Hierzu werden erneut die in Kapitel 4 definierten Kriterien betrachtet und es wird erläutert, inwiefern das Toolkit diese erfüllen konnte. Außerdem wird beschrieben, wie das Toolkit eingesetzt wurde, um ein reales Experiment aufzusetzen und durchzuführen. Dieses Experiment wurde in Zusammenarbeit mit Bianca Schneider im Rahmen ihrer Masterarbeit zur Temperaturwahrnehmung in VR durchgeführt. Auf Basis dieses praktischen Einsatzes des Toolkits werden Schlüsse auf die Stärken und Schwächen sowie Erweiterungsmöglichkeiten des Toolkits getroffen. Über einen weiteren Test des Toolkits an der Hochschule Karlsruhe konnte ebenfalls Feedback über die Nützlichkeit des Toolkits und die Erfüllung der Kriterien gesammelt werden. Abschließend wird im Fazit die Entstehung des Toolkits und dessen Evaluation resümiert.

2 Grundlagen

In diesem Kapitel werden die theoretischen Grundlagen erläutert, die für die Konzeption des wissenschaftlichen Toolkits erforderlich waren. Zunächst wird der Begriff der Virtual Reality definiert und deren Eigenschaften und Besonderheiten erläutert. Anschließend werden Methoden der Datenerhebung und -verarbeitung sowie der Visualisierung vorgestellt.

2.1 Virtual Reality

Virtuelle Realität, bzw. „Virtual Reality“ (VR) ist ein sehr breit gefächelter Begriff, der aus unterschiedlichen Blickwinkeln betrachtet werden kann. Eine mögliche Definition von Jason Jerald ist die einer computergenerierten Simulation, mit welcher Nutzer interagieren können, als wäre sie real (Jerald, 2016, S. 9). Im Unterschied zu einer herkömmlichen Computersimulation, die auf der Interaktion mit einem grafischen Nutzerinterface auf einem Bildschirm basiert, interagiert der Benutzer einer virtuellen Realität mit dieser von innen heraus (Dörner, Broll, Grimm, Jung, & Göbel, 2013, S. 16 f.). Schon im Jahre 1965 prognostizierte der amerikanische Informatiker Ivan E. Sutherland ein Display, welches die komplette Materie eines Raums kontrollieren kann und so eine perfekte Simulation erschafft, die von einem Nutzer nicht von der Wirklichkeit zu unterscheiden wäre (Sutherland, 1965). Ein ähnliches Konzept findet sich im „Holodeck“ aus der Fernsehserie Star Trek (Dörner et al., S. 13). Eine solche perfekte virtuelle Realität, in der alle Sinneseindrücke eines Menschen simuliert werden, ist bis heute allerdings technisch nicht umsetzbar. Dennoch führen neueste technologische Entwicklungen dazu, dass VR-Erfahrungen immer realistischer werden (Dörner et al., 2013, S. 17 f.).

2.1.1 Immersion und Präsenz

Als zentrale Merkmale, die eine virtuelle Realität ausmachen, werden oft die Begriffe Immersion und Präsenz genannt. Immersion beschreibt laut Slater und Wilbur, inwieweit eine computerbasierte Darstellung es schafft, eine Illusion der Realität zu erzeugen. Verschiedene Faktoren können Immersion erzeugen und

verbessern, zum Beispiel die Abschottung von der Außenwelt, die Auflösung des Bildes oder die Stimulation verschiedener Sinne (Slater & Wilbur, 1997, S. 3).

Als Präsenz beschreiben Slater und Wilbur den Geisteszustand von Nutzern einer virtuellen Umgebung. Ein Gefühl der Präsenz bedeutet, dass der Nutzer sich komplett als Teil einer virtuellen Umgebung fühlt und die virtuelle Welt als realer als die echte Welt betrachtet. Im Gegensatz zur Immersion, die nur von den angewandten Technologien abhängt, kann der Präsenzgrad auch in der gleichen Anwendung von Nutzer zu Nutzer verschieden sein. Verschiedene Studien zeigen aber, dass höhere Immersion auch oft in höherer Präsenz resultiert (Slater & Wilbur, 1997, S. 4 ff.). Jason Jerald beschreibt die Präsenz als „function of both the user and immersion“ (Jerald, 2016, S. 46), also als eine Variable, die sowohl von der objektiv messbaren Immersion als auch vom individuellen Nutzer abhängt.

2.1.2 Ausgabe- und Eingabemethoden

Virtual Reality-Systeme können unterschiedliche Ausgabetechnologien nutzen, um eine virtuelle Realität darzustellen. Ziel eines solchen Systems ist es immer, eine überzeugende virtuelle Welt zu schaffen, welche möglichst so wahrgenommen wird, wie die reale Welt. Zu den Ausgabetechnologien zählen sowohl visuelle als auch akustische und haptische Ausgabemethoden (Grimm, Herold, Reiners, & Cruz-Neira, 2013, S. 127-129).

Ein Beispiel einer visuellen VR-Ausgabetechnologie ist ein sogenanntes CAVE-System, bei welchem es sich um einen Raum handelt, welcher an Wänden, Decke und Boden mit Bildschirmen bedeckt ist (Grimm et al., 2013, S. 132 f.).



Abbildung 1 Ein CAVE Ausgabesystem (Grimm, Herold, Reiners, & Cruz-Neira, 2013, S. 134)

Ein weiteres visuelles Ausgabesystem, welches auch das in dieser Arbeit verwendete Ausgabesystem darstellt, ist das Head-Mounted-Display (HMD). Ein HMD ist ein Display, welches über eine brillenähnliche Vorrichtung am Kopf des Nutzers angebracht wird. Der Nutzer betrachtet die virtuelle Welt, die auf einem integrierten Bildschirm dargestellt wird, durch ein optisches Linsensystem. HMDs lassen sich in verschiedene Kategorien einteilen, wie Durchsicht-HMDs, Video-HMDs und Direktsicht-HMDs (Grimmet al., 2013, S. 142 f.). Ein wichtiger Aspekt eines HMDs ist das Tracking von Position und Rotation. Der im HMD dargestellte Bildschirm wird anhand der vom Tracking erkannten Position so angepasst, dass der Nutzer sich im virtuellen Raum umsehen kann. Ohne Tracking würde keine überzeugende virtuelle Realität entstehen, da für den Nutzer der Eindruck entstehen würde, dass sich alle Objekte mit seiner Kopfbewegung mitdrehen (Jerald, 2016, S. 32).

Bei dem HMD des in dieser Arbeit verwendeten VR-Systems, der HTC Vive, handelt es sich um ein Direktsicht-HMD. Bei dieser Art HMD blickt der Nutzer durch ein Linsensystem auf ein undurchsichtiges Display. Der Nutzer ist beim Tragen des HMDs optisch komplett von der Außenwelt abgeschottet, nimmt also nur noch die virtuelle Welt wahr (Grimm et al., S. 147 f.).



Abbildung 2 Die HTC Vive (HTC, 2018)

Für eine wirklich immersive VR-Erfahrung spielen auch die Eingabesysteme eine große Rolle (Jerald, 2016, S. 43). Sie dienen der Erfassung von Nutzeraktionen, damit der Nutzer mit der virtuellen Welt interagieren kann (Grimm, Herold, Hummel, & Broll, 2013, S. 97). Dabei können beispielsweise bewusste Eingaben des Nutzers (z. B. ein Knopfdruck) oder Daten, wie die Kopfposition, verarbeitet werden. Für letzteres wird eine Methode namens Tracking verwendet. Tracking bedeutet, dass die Position und Rotation eines Gegenstandes im Raum, in diesem Fall der Kopf, kontinuierlich bestimmt und dem Computer übergeben wird (Grimm et al., 2013, S. 98).

Das in dieser Arbeit eingesetzte HTC Vive VR-System besteht neben dem HMD auch aus zwei Bewegungscontrollern und zwei Basisstationen. Die Basisstationen verwenden Valves Lighthouse-Trackingtechnologie und werden vom HMD und den Controllern genutzt, um ihre Position innerhalb eines vorher festgelegten Bereiches festzustellen. Die Controller ermöglichen es, neben grundlegender Tasteneingabe, in der virtuellen Umgebung dreidimensional mit Gegenständen zu interagieren, diese zu werfen usw. Als Software-API wird das, ebenfalls von Valve entwickelte, SteamVR verwendet (HTC, 2018).

2.2 Datenerhebung und Verarbeitung

Ein Großteil des in dieser Arbeit konzipierten Toolkits befasst sich mit der Aufzeichnung und Verarbeitung von Daten. Dabei handelt es sich um automatisch aufgezeichnete Telemetriedaten sowie Daten, die manuell eingegeben werden. Dieses Kapitel enthält eine Einführung in die Bedeutung von Daten für wissenschaftliche Experimente sowie eine Zusammenfassung von für das Toolkit relevanten Methoden zur Erfassung und Analyse von Daten.

2.2.1 Daten in wissenschaftlichen Experimenten

Der Umgang mit Daten ist ein erheblicher Teil eines wissenschaftlichen Experimentes. Ein Experiment ist eine systematische Form der empirischen Untersuchung. Ein Wissenschaftler erzeugt hierbei eine bestimmte Situation, in der er Beobachtungen aufstellen kann. Experimente haben immer das Ziel, Daten zu erheben und damit neues Wissen zu erlangen (Rack & Christophersen, 2009, S. 18). In der Regel wird zunächst eine Hypothese oder Fragestellung definiert, welche anschließend durch ein Experiment überprüft wird. (Kaya, 2009, S. 57 f.).

Als Daten werden Informationsobjekte bezeichnet, die zum Zweck der Datenweitergabe in der Form von Bildern, Texten, Graphen usw. gespeichert werden (Wiegand, 1998, S. 163 f.). Reine Daten besitzen keine Relation zu anderen Daten und stellen auch noch kein Wissen dar, da sie noch nicht interpretiert und in einen Kontext gesetzt wurden (Keller & Tergan, 2005, S. 3).

Daten werden verwendet, um Informationen zu generieren. Informationen, sind Daten, die interpretiert und in einen Kontext gesetzt wurden. Aus Informationen entsteht letztendlich Wissen (Keller & Tergan, 2005, S. 3). In wissenschaftlichen Forschungsprozessen spielen Daten und Informationen in den Schritten der Datenerhebung und Datenanalyse eine große Rolle (Kaya, 2009, S. 49).

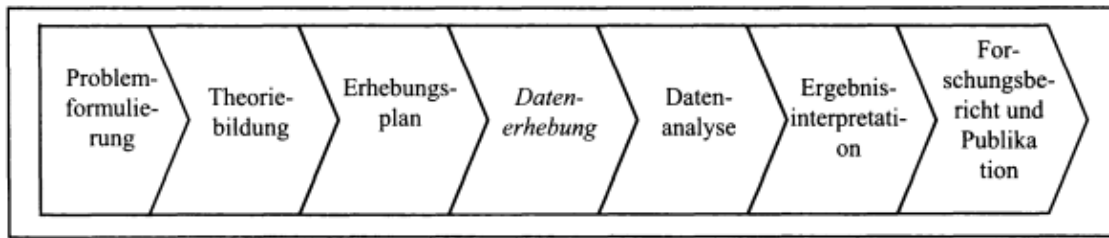


Abbildung 3 Ablauf eines empirischen Forschungsprozesses (Kaya, 2009, S. 49)

Bei Forschungsdaten kann allgemein zwischen Primärdaten und Sekundärdaten unterschieden werden. Während Primärdaten Daten sind, welche im Verlauf des Experimentes erhoben werden, sind Sekundärdaten bereits vorhandene Datensätze, welche, ohne dass eine eigene Erhebung erforderlich ist, aufbereitet und analysiert werden können (Kaya, 2009, S. 49). Für das in dieser Arbeit konzipierte Toolkit sind hauptsächlich die Primärdaten relevant.

Primärdaten können zum Beispiel über Befragungen und Beobachtungen erhoben werden. Bei einer Befragung werden Probanden oder Auskunftspersonen zu einem Erhebungsgegenstand befragt und deren Antworten gesammelt (Kaya, 2009, S. 51). Als Beobachtung bezeichnet man dagegen die systematische Erfassung von Daten über visuelle oder akustische Wahrnehmung des Geschehens (Kaya, 2009, S. 56). Die Beobachtung kann entweder über einen menschlichen Beobachter oder über technische Geräte, wie z. B. eine Kamera oder ein Mikrophon, stattfinden (Kaya, 2009, S. 57).

2.2.2 Datenvorbereitung

Ein Datensatz ist nach dessen Sammlung und Speicherung noch nicht unbedingt in einem idealen Zustand, um ihn zu analysieren. Bei der Sammlung von Daten kann es oft passieren, dass es innerhalb des Datensatzes fehlende, ungenaue oder falsche Daten gibt. Auch kann es sein, dass der Datensatz mehr Daten enthält, als für die weitere Analyse sinnvoll wäre (Han, Kamber, & Pei, 2012, S. 83). Daten können außerdem in einer unstrukturierten Form vorliegen, zum Beispiel über ein langes Textdokument verteilt (Skiena, 2017, S. 14). Deshalb ist der erste Schritt nach der eigentlichen Sammlung von Daten, die Vorverarbeitung und Bereinigung, genannt Datenvorbereitung oder Data Preprocessing. Hierbei

werden verschiedene Techniken, wie Datenreduktion, Datenintegration und Datenbereinigung angewandt, um einen Datensatz zu erhalten, der sich gut zur weiteren Analyse eignet (Han et al., 2012, S. 83).

Ist beispielsweise der Datensatz zu groß, um ihn effizient weiterverwerten zu können, kommt Datenreduktion zum Einsatz. Eine Möglichkeit zur Reduktion, bei der wenig Information verloren geht, ist die Wavelet Transformation. Durch Transformation eines Datenvektors auf einen Koeffizientenvektor können die Daten einfacher komprimiert werden (Han et al., 2012, S. 99-102).

Bei einzelnen fehlenden oder erkennbar falschen Daten sollte das entsprechende Attribut ersetzt werden. Hierzu kann z. B. der Mittelwert der umliegenden Attribute verwendet werden oder der Wert wird durch eine globale Konstante ersetzt. Je nachdem wie wichtig es ist, dass jeder Wert akkurat ist, gibt es auch die Möglichkeit, das komplette Objekt zu löschen bzw. zu ignorieren, wobei es sich dabei um eine Lösung handelt, die nur im Notfall eingesetzt werden sollte (Han et al., 2012, S. 88).

2.2.3 Data Mining

Data Mining ist der Prozess des Auffindens von Mustern innerhalb großer Datensätze durch Einsatz von analytischen Methoden (Han, Kamber, & Pei, 2012, S. 8). Techniken des Data Mining werden heutzutage in der Industrie, Wissenschaft und von Regierungen zur Kriminalitätsbekämpfung angewandt. (Clifton, 2018) Die primären Ziele des Data Mining sind die Beschreibung von Mustern in einer für Menschen lesbaren Form sowie die Vorhersage von möglichen weiteren Entwicklungen (Fayyad, Gregory, & Smyth, 1996, S. 44).

Ein Datensatz, welcher im Data Mining verwendet wird, besteht aus Objekten und Attributen. Ein Objekt bezeichnet dabei eine Instanz, wie z. B. eine Testperson oder einen Versuchsablauf. Objekte besitzen Attribute, welche man auch als Variablen der jeweiligen Objekte bezeichnen kann. Hierbei kann es sich z. B. um numerische Werte oder Zeichenfolgen handeln. Ein aus diesen Elementen bestehender Datensatz kann strukturiert in einer Datenbank gesichert werden (Han et al., 2012, S. 40-43).

2.2.4 Datenspeicherung

Bei vielen Datenerhebungen entstehen heutzutage Datensätze, die ein sehr großes Volumen an Daten enthalten. Datenspeicher bieten jedoch keinen unbegrenzten Speicherplatz. Aus diesem Grund muss darauf geachtet werden, dass Daten auf eine effiziente Art und Weise gespeichert werden, die es aber auch ermöglicht, problemlos einzelne Daten oder Teildatensätze zu extrahieren (Chen, Mao, & Zhang, 2014, S. 33-35). Hierfür muss ein Format festgelegt werden, in dem die Daten strukturiert sind. Für ein optimal verwertbares Datenformat definiert Skiena folgende weitere Kriterien:

- Es kann problemlos von Computern verarbeitet werden.
- Es kann problemlos von Menschen gelesen werden.
- Es wird von möglichst vielen Programmen und Systemen unterstützt.

(Skiena, 2017, S. 61 f.)

Zwischen diesen Kriterien und dem zuvor genannten Kriterium der Platzeffizienz besteht jedoch unter Umständen ein Widerspruch, da Daten, welche auf minimalen Speicher komprimiert wurden, in der Regel nicht mehr gut für Menschen lesbar sind und auch keine gute Kompatibilität zu Programmen besitzen. Skiena argumentiert, dass wegen der sinkenden Kosten von Speicherplatz die Lesbarkeit letztendlich wichtiger ist (Skiena, 2017, S. 62).

2.2.5 Datenvisualisierung

Die Datenvisualisierung ist eine Methode, aus Daten ausgewertete Informationen und Wissen zu verdeutlichen und in einer für Menschen übersichtlich gestalteten Form darzustellen (Keller & Tergan, 2005, S. 1 f.). Oft entsteht bei Experimenten, Datenerhebungen usw. ein so großer Datensatz, dass es so gut wie unmöglich ist, darüber einen Überblick zu behalten. Deshalb wird die Visualisierung als Werkzeug verwendet, das dabei helfen soll, Informationen zu verstehen und in einen Kontext einordnen zu können. Sie nutzen die kognitive Fähigkeit des Menschen, visuelle Informationen sehr schnell zu verarbeiten (Keller & Tergan, 2005, S. 5). Außerdem können Visualisierungen

dazu verwendet werden, den Fokus auf bestimmte Aspekte und Zusammenhänge von Informationen zu legen (Keller & Tergan, 2005, S. 6). Auch in der Erkennung von fehlerhaften Daten können Visualisierungen nützlich sein, da sie beispielsweise Messwerte, die extrem von den restlichen Werten abweichen, sofort erkennbar machen (Skiena, 2017, S. 155).

Um eine Visualisierung zu erstellen, müssen Rohdaten zunächst, wie im Kapitel 2.2.2 Datenvorbereitung beschrieben, transformiert und bereinigt werden. Für die aufbereiteten Daten lassen sich visuelle Strukturen erstellen. (Jaeschke, Leissler, & Hemmje, 2005, S. 121). Die verarbeiteten Daten werden schließlich auf die visuelle Struktur angewendet, um eine oder mehrere fertige Ansichten anzuzeigen (Jaeschke et al., 2005, S. 126)

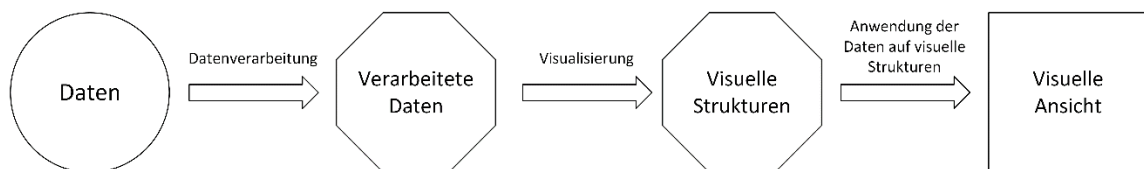


Abbildung 4 Erstellungsprozess einer Visualisierung (Eigene Erstellung, in Anlehnung an Jaeschke, Leissler, & Hemmje, 2005, S. 121)

Bei der Erstellung visueller Strukturen müssen Entscheidungen getroffen werden, wie Daten visuell angeordnet werden und welche Eigenschaften sie besitzen. Es muss beispielsweise entschieden werden, wie viel Platz die Visualisierung benötigt, welche Farben die Elemente haben, in welcher Reihenfolge sie gezeigt werden und wie sie zueinander angeordnet werden sollen (Mazza, 2009, S. 20 f.).

3 Betrachtung bereits vorhandener Toolkits

Im nachfolgenden Kapitel soll Software vorgestellt werden, die bereits Teile der Funktionalität eines wissenschaftlichen Toolkits für VR abbildet. Für die Konzeption des in dieser Arbeit erstellten Toolkits war es wichtig, auch vorhandene Software zu untersuchen, um festzustellen, ob es sinnvoller wäre, auf einem bereits vorhandenen Toolkit aufzubauen oder eine komplett neue Software zu erstellen.

3.1 ViSTA Virtual Reality Toolkit

Das ViSTA Virtual Reality Toolkit ist ein C++-basiertes Open-Source-Toolkit, welches das Ziel hat, “wissenschaftliche Anwendungen mit Hilfe von Methoden und Techniken der Virtual Reality und umfassender Visualisierung zu verbessern“ (RWTH Aachen, 2018). Es besteht aus mehreren C++ Bibliotheken, welche plattformunabhängig sind.

Grundsätzlich ist ViSTA auf das Visualisieren von wissenschaftlichen Daten und nicht auf das Sammeln von Daten innerhalb von Virtual Reality-Umgebungen ausgelegt. Eine Funktionsüberschneidung gibt es jedoch bei der Möglichkeit, Messgeräte anzubinden. Eine Erweiterung des ViSTA Toolkits wäre zwar theoretisch denkbar, ist aber im Rahmen der Thesis nicht praktikabel, da es keine ausführliche öffentliche Dokumentation zu den Skripten des Toolkits gibt. Des Weiteren kann durch die Verwendung von plattformunabhängigen C++ Bibliotheken nicht auf Unity-spezifische Funktionen, wie den Inspektor, zugegriffen werden.

Für weitere Informationen wurde eine Kopie des Conference Papers angefragt, in welchem das Toolkit vorgestellt wird. Diese Anfrage wurde jedoch bis zur Fertigstellung dieses Kapitels nicht beantwortet, weshalb die Funktionalität des ViSTA Toolkits im Rahmen dieser Arbeit nicht umfassend untersucht werden konnte.

3.2 Unity Analytics

Unity Analytics ist Unitys integriertes Toolkit zur Erfassung und Verarbeitung von Nutzerdaten (Unity Technologies, 2018). Es richtet sich primär an Spieleentwickler, die das Verhalten ihrer Spieler untersuchen möchten. Hierzu gibt es sowohl Standardereignisse als auch die Möglichkeit Events zu definieren, die z. B. ausgelöst werden, wenn der Spieler ein bestimmtes Level betritt. Alle gesammelten Analytics Daten werden an Unitys Analytics-Server gesendet. Auf Unitys Webseite gibt es dann die Möglichkeit, sich verschiedene Visualisierungen zu den Daten anzusehen (Abbildung 5).

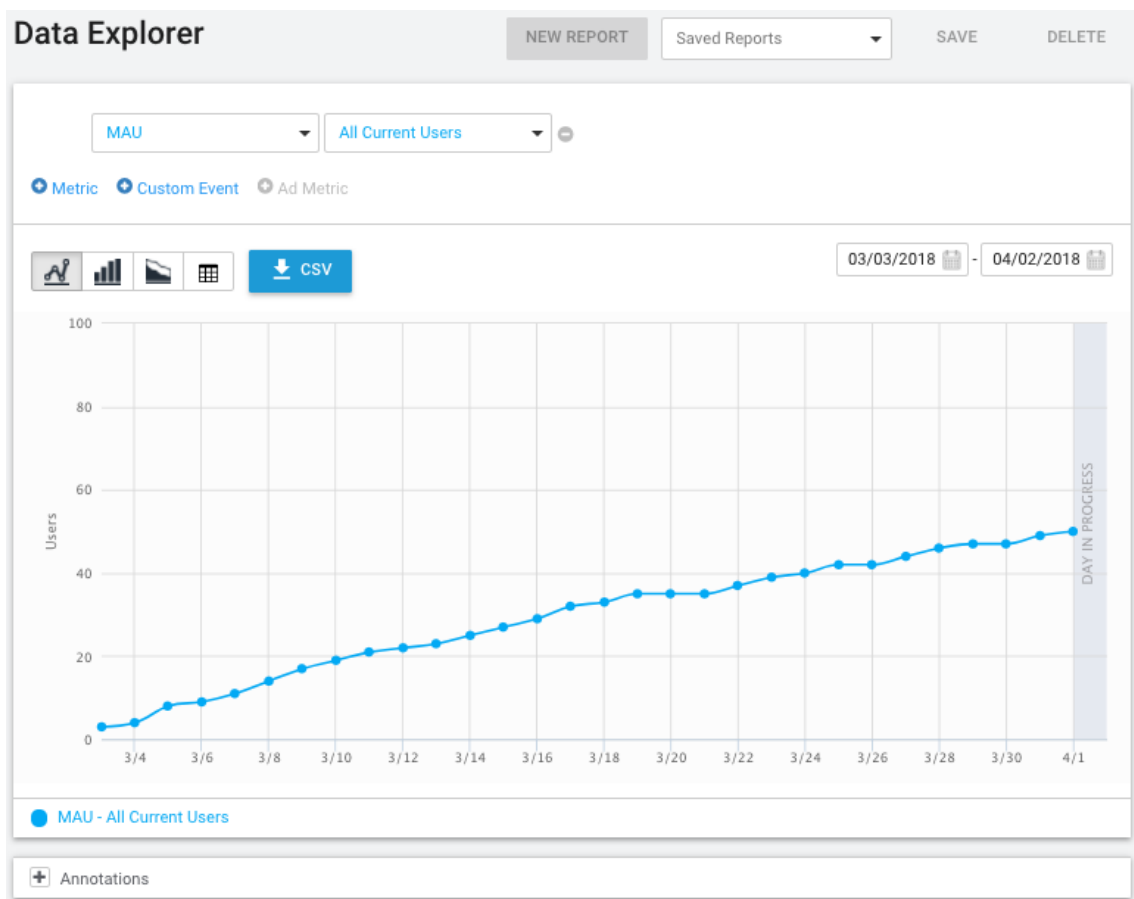


Abbildung 5. Beispiel einer automatisch generierten Visualisierung in Unity Analytics. (Unity Technologies, 2018)

Das Event System von Unity Analytics wäre prinzipiell eine denkbare Basis für ein wissenschaftliches Toolkit. Jedoch wäre dieses auf die Nutzung der Rohdaten, die Analytics ausgibt, beschränkt, da der Quellcode von Analytics nicht einsehbar oder erweiterbar ist. Es ist außerdem nicht ideal, dass die

Aufzeichnung von Daten nur über Unitys Server funktioniert, da in Testumgebungen unter Umständen keine Internetverbindung besteht. Darüber hinaus ist die Ausgabe der Rohdaten nach deren Sammlung nur mit einem Abonnement von Unity Pro möglich (Unity Technologies, 2018). Möchte man mit Unity Analytics regelmäßig die Transformationsdaten eines GameObjects aufzeichnen, ist dies außerdem praktisch unmöglich, da es ein festgelegtes Limit von 100 „Custom Events“ pro Stunde gibt (Unity Technologies, 2018). Muss zum Beispiel nachträglich analysiert werden, wo sich ein Proband in der virtuellen Umgebung am meisten aufgehalten hat, könnte die Position des Probanden nur alle 36 Sekunden abgefragt werden, was für viele Zwecke zu ungenau wäre.

3.3 Schlussfolgerungen

Die untersuchten Toolkits eignen sich nicht als brauchbare Basis, um darauf ein Toolkit für wissenschaftliche Experimente aufzubauen. ViSTA wäre prinzipiell eine interessante Basis, da es bereits auf wissenschaftliche Nutzung ausgelegt ist, einen offenen Quellcode hat und die Möglichkeit bietet, Visualisierungen von Daten in VR zu betrachten. Die unzureichende Dokumentation und die Tatsache, dass auf Anfrage keine weiteren Informationen zur Verfügung gestellt wurden, machten den Ansatz, auf ViSTA aufzubauen, jedoch unpraktikabel. Unity Analytics bietet auf den ersten Blick ein interessantes Event-System, welches aber nicht erweiterbar ist und viele Einschränkungen besitzt. Mit neu erstellten Skripten ließe sich ein deutlich flexibleres und genaueres Eventsystem erschaffen, das sich besser für die wissenschaftliche Nutzung eignen würde.

Aus diesen Gründen wurde die Entscheidung getroffen, ein Toolkit zu entwickeln, welches nicht auf einem bereits vorhandenen Toolkit aufbaut. Auf diese Art und Weise kann eine neue Basis für die Durchführung von Experimenten in VR entwickelt werden, die komplett dokumentiert und erweiterbar ist.

4 Konzeption des Toolkits

In diesem Kapitel werden die aus dem Theorieteil gewonnenen Kenntnisse angewandt, um ein Konzept eines wissenschaftlichen Toolkits für VR-Anwendungen zu erstellen. Es wird zunächst die Zielsetzung definiert, zusammen mit Kriterien, die später der Evaluation dienen sollen. Anschließend wird der konzeptionelle Aufbau der verschiedenen Module beschrieben. Zuletzt wird dargestellt, wie die Module des Toolkits technisch aufgebaut werden sollen, indem notwendige Klassen und deren Relationen zueinander definiert werden.

4.1 Zielsetzung

Im Rahmen der Masterthesis sollen ein Konzept und ein Prototyp eines wissenschaftlichen Toolkits für Experimente in VR-Umgebungen entstehen. Dieses Toolkit soll es Wissenschaftlern vereinfachen, Experimente in VR durchzuführen. Hierzu besitzt es Funktionen, welche den Aufbau und die Steuerung des Experiments sowie die spätere Analyse von automatisch aufgezeichneten Daten ermöglichen. Die Software soll im Rahmen einer Studie zur Temperaturwahrnehmung in VR, sowie einem Pseudoexperiment getestet und evaluiert werden.

4.1.1 Kriterien

Basierend auf den Zielen und Anforderungen werden in diesem Kapitel Kriterien formuliert. Diese sollen vor der Erstellung des Konzepts zur Spezifikation und nach der Durchführung des Tests zur Evaluation der Software verwendet werden (Herczeg, 2018, S. 207). Die Kriterien werden genutzt, um festzustellen, ob die Zielsetzung erreicht wurde. Die hier definierten Kriterien „Erweiterbarkeit“ und „Bedienbarkeit“ basieren auf Herczogs Liste der „Basiskriterien der Gebrauchstauglichkeit“ (Herczeg, 2018, S. 211).

Erweiterbarkeit

Das Toolkit soll so aufgebaut sein, dass es möglichst einfach ist, weitere Funktionen einzubauen und die bestehenden zu erweitern. Hierzu sollen auch die Skripte möglichst sauber geschrieben und dokumentiert sein. Die Skripte sollen, soweit es möglich ist, voneinander unabhängig funktionieren.

Bedienbarkeit

Die Benutzung des Toolkits soll keine große Einarbeitung erfordern. Ein Großteil der Bedienung soll über das Unity Interface stattfinden und die Notwendigkeit von Skripten im Rahmen der Benutzung des Toolkits minimiert werden. Durch einfache Bedienelemente und Erklärungen innerhalb des Programms sollen Schwierigkeiten bei der Bedienung minimiert werden. Alle Eingabemöglichkeiten sollen bestmöglich die Bedienung der Systeme der Software unterstützen (Herczeg, 2018, S. 244 f.).

Performance

Das Toolkit soll sich minimal auf die Performance der Unity Anwendung auswirken. Dazu müssen Funktionsdurchläufe minimalisiert werden, sodass in jedem Frame, den das Toolkit durchläuft, möglichst wenige komplexe Rechenoperationen durchgeführt werden. Sichtbare Performanceprobleme in VR können zu einem Präsenzbruch führen (Jerald, 2016, S. 47). Dies wiederum kann die aus dem Experiment gewonnen Daten verfälschen.

Anbindung externer Hardware

Das Toolkit soll es ermöglichen, externe Sensoren zu verbinden und möglichst ohne Verwendung von zusätzlicher Software Daten von dieser Hardware auszulesen und zu verwenden. Idealerweise könnte der Nutzer ein Gerät anschließen, dieses ohne eigene Skripte in Unity auslesen und dessen Daten aufzeichnen.

4.2 Konzept

Das Scientific Virtual Reality Toolkit (STK) soll die Durchführung von wissenschaftlichen Probandentests in Virtual Reality erleichtern. Es stellt dafür verschiedene Funktionen zur Verfügung, welche es ermöglichen, mit geringem Programmieraufwand eine Testumgebung zu erstellen, in der alle für den Test relevanten Daten gesammelt, geordnet und gespeichert werden. Daten externer Geräte, wie beispielsweise Sensoren, können über eine serielle Schnittstelle eingelesen und in Verbindung mit Testereignissen gebracht werden. Alle gesammelten Daten werden im JSON-Format zur Weiterverarbeitung gespeichert. Zur Analyse und Visualisierung der gesammelten Daten enthält das STK einige Standardskripte in der Programmiersprache R, welche gegebenenfalls angepasst und erweitert werden können. Unter Verwendung der gespeicherten Daten kann der komplette Ablauf eines Probandentests wiederhergestellt und aus unterschiedlichen Blickwinkeln betrachtet werden.

4.2.1 Module

Das Toolkit ist in verschiedene Module aufgeteilt, welche aus zusammenhängenden Skripten bestehen, die jeweils ein zusammengehöriges Set an Funktionen ermöglichen. In diesem Kapitel werden die Funktionen der Module erläutert.

4.2.1.1 Testaufbau

Das Testaufbaumodul kontrolliert den Aufbau und Ablauf des Experiments. Es enthält Skripte, die es dem Nutzer ermöglichen, den Experimentablauf zu definieren und diesen, während der Test durchgeführt wird, zu steuern. Aus dem hier definierten Ablauf ergibt sich außerdem der Aufbau der JSON-Datei (Siehe Kapitel 4.2.1.2 Telemetrie).

Ein Experiment wird vom Toolkit in verschiedene Forschungsabschnitte, genannt „Stages“, aufgeteilt. In einer Stage wird beispielsweise eine bestimmte Messung ausgeführt oder ein bestimmtes Set an Fragen untersucht. Ein

Experiment kann eine beliebige Menge von Stages enthalten, hat jedoch mindestens eine.

Der Test Controller

Der Test Controller ist ein Interface Element, welches dem Nutzer während des Testablaufs am Bildschirm angezeigt wird, dem Probanden in VR jedoch nicht. Über ihn wird die Struktur des Experiments definiert. Der Test Controller enthält außerdem Bedienelemente und Eingabeflächen, die während des Tests verwendet werden können.

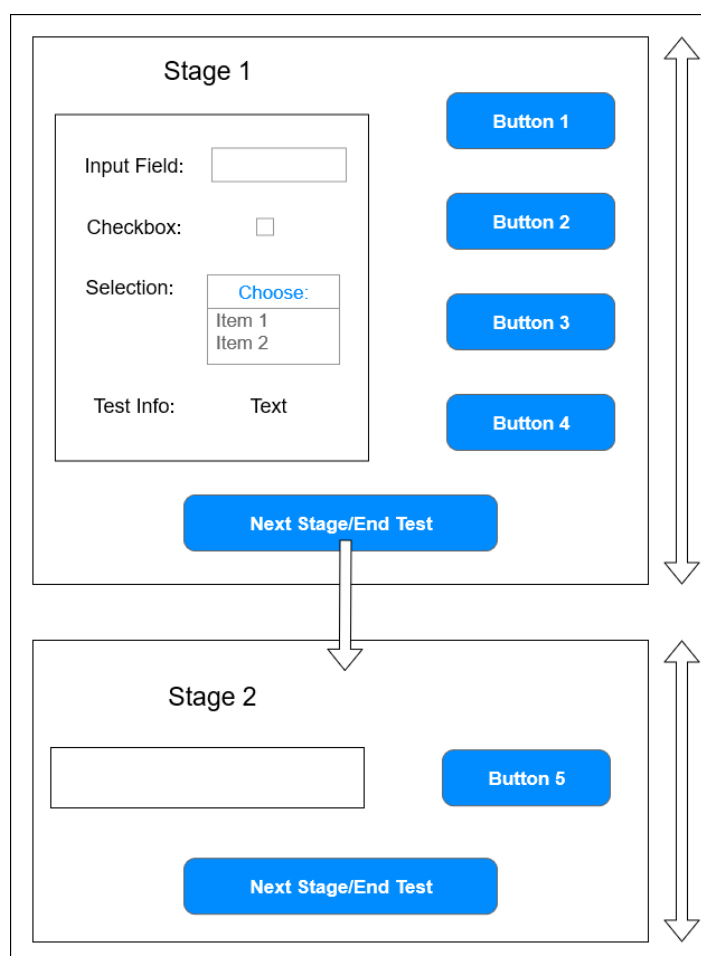


Abbildung 6 Mockup des Testcontrollers

Der Testcontroller ist in Stages unterteilt. Mit einem Button kann der Nutzer eine laufende Stage beenden bzw. die nächste Stage starten. Der Nutzer kann vordefinieren, was geschieht, wenn eine neue Stage beginnt, indem er auswählt, welche GameObjects in der Szene aktiviert oder deaktiviert werden sollen und welche GameObjects eine bestimmte Funktion ausführen sollen. Vor und

während jeder Stage kann der Nutzer Daten, wie z. B. Beobachtungen oder Probandendaten, eingeben. Diese Daten sind Attribute, welche vorher in Unity festgelegt werden. Sie können aus Texteingabefeldern, Checkboxen oder Auswahlfeldern bestehen.

Auf der rechten Seite des Testcontrollers befinden sich Buttons, welche beliebige nutzerdefinierte Funktionen in der Szene auslösen. Zum Beispiel kann der Nutzer damit einen Gegenstand in der Szene erscheinen lassen.

Die verschiedenen Stages, Attribute und Buttons können in der Unity Szene erstellt werden. Um dies zu vereinfachen, erweitert das Toolkit die Inspektoroberfläche um einen Custom Inspector, welcher es ermöglicht, diese Elemente über einen Knopfdruck zu erstellen. (Siehe Kapitel 4.2.1.6 Interface)

4.2.1.2 Telemetrie

Das Telemetriemodul bildet den Schwerpunkt der Funktionalität des Toolkits. In diesem Modul werden relevante Daten während eines Experimentes gesammelt und gespeichert. Dies geschieht über ein Event-System. Im Toolkit kann der Nutzer eigene Events mit dynamischen Eigenschaften erstellen oder auf eine Auswahl an Standard-Events zurückgreifen. Die Events werden von einem Empfängerskript zunächst als Objekte abgespeichert und später in ein Format umgewandelt, welches die Weiterverwertung der Daten außerhalb von Unity ermöglicht. Das Telemetriemodul ermöglicht ebenfalls den Anschluss von externen Geräten, wie zum Beispiel Temperatursensoren.

Während des Ablaufs eines Experiments gibt es verschiedene Datentypen, die für die spätere Auswertung relevant sind. Zum einen gibt es Daten, welche sich nicht aus dem Testablauf selbst ergeben, sondern die manuell eingegeben werden müssen. Dazu zählen zum Beispiel Name und Alter des aktuellen Probanden, Antworten auf Fragen oder Beobachtungen des Studienleiters. Zum anderen gibt es Daten, die sich aus dem eigentlichen Ablauf des Experimentes ergeben, zum Beispiel die aktuelle Position des Probanden in der Testumgebung oder mit welchen Objekten er interagiert. Für diesen Datentyp wird im Telemetriemodul das Eventsystem eingeführt.

Das Event System

Ein Event bzw. Ereignis wird von Hedtstück als eine Zustandsveränderung eines Systems bezeichnet, welche zu einem festen Zeitpunkt (*Timestamp*) eintritt und welches oft Attribute enthält, die weitere Informationen beinhalten (Hedtstück, 2017, S. 12 f.). Im Toolkit dienen Events dazu, relevante Ereignisse während eines Experiments zu beschreiben. Zum Beispiel kann ein Event dann entstehen, wenn ein Proband einen Gegenstand aufhebt. Attribute können in diesem Fall der Name des Gegenstands, der aufgehoben wird, und der Zeitpunkt, zu dem er aufgehoben wird, sein. Events werden zunächst vom Nutzer des Toolkits vordefiniert und mit Attributen versehen. Während der Laufzeit werden die Events dann von einem „Event Sender“, einer Komponente, die in der Szene liegt, abgesendet. Der Event Sender versieht die Attribute zunächst mit aktuellen Werten. Die Werte können über ein Skript vom Nutzer festgelegt werden, oder automatisch, wenn das Event und der Event Sender über das “Track Object” Interface erstellt wurden (Siehe Kapitel 4.2.1.6 Interface). Die gesendeten Events werden vom „Event Receiver“ gesammelt, einer statischen Klasse, die eine Liste mit Referenzen auf alle ausgesendeten Events enthält.

Nach Abschluss eines Versuchs werden alle Events, zusammen mit den im Testcontroller eingegebenen Attributen, zum späteren Abruf in einer Textdatei gesichert. Zur Formatierung der Daten wird die JavaScript Object Notation, kurz JSON, verwendet. Bei JSON handelt es sich um ein Format zum Datenaustausch zwischen Anwendungen. Es besitzt eine ähnliche Syntax wie Programmiersprachen der C-Familie, ist aber davon unabhängig (Crockford, 2018). JSON wurde gewählt, weil es eine bekannte und lesbare Syntax besitzt, die sich in Unity verhältnismäßig leicht aufbauen lässt. Es ist außerdem möglich, JSON Dateien in der Statistiksprache R zu verarbeiten, welche für das Modul Visualisierung verwendet wird.

Begrenzung der Datenmenge

Der Event Receiver erhält während des Experiments fortlaufend neue Events. Gerade bei Tracking-Events, die mehrmals pro Sekunde erstellt werden können, und bei sehr langen Experimenten, kann dies schnell zu großen Datenmengen

und schließlich zu einer Überladung des verfügbaren Speichers führen. Der Nutzer kann deshalb ein Limit für gespeicherte Events festlegen. Wird dieses Limit erreicht, gibt es mehrere Möglichkeiten, wie das Toolkit damit umgehen kann.

Eine Möglichkeit ist ein sogenanntes „Sliding Window“. Nach Erreichen des eingestellten Maximums an Events wird das jeweils früheste gespeicherte Event gelöscht, um das neueste Event zu speichern. Es gibt also ein sich zeitlich fortbewegendes Fenster an gespeicherten Daten, welches immer weiter nach hinten verschoben wird (Hedtstück, 2017, S. 27).

Eine weitere Möglichkeit ist eine Reduktion der bisherigen Daten beim Erreichen des Limits. Hierbei wird jedes zweite aufgezeichnete Event aus der Liste entfernt. Damit gehen zwar Informationen verloren, aber es wird weiterhin der komplette Ablauf des Experiments von Anfang bis Ende mit verringerter Präzision abgedeckt.

Alternativ kann der Nutzer einstellen, dass bei Erreichen des Maximums eine JSON-Datei erstellt wird, welche alle bisherigen Events enthält. Die Aufzeichnung wird dann neu begonnen und später in einer weiteren JSON-Datei gespeichert, wenn der Speicher wieder voll oder das Experiment zu Ende ist. Diese Methode sorgt dafür, dass der Arbeitsspeicher nicht überfüllt wird und trotzdem keine Daten verloren gehen, sollte also eingesetzt werden, wenn die Genauigkeit der Daten eine wichtige Rolle spielt. Bei dieser Methode muss der Nutzer jedoch darauf achten, genug freien Festplattenspeicher zu besitzen und die generierten Dateien gegebenenfalls auf ein anderes Medium zu verschieben.

Anbindung externer Hardware

Nicht alle für ein Experiment relevanten Daten stammen aus der virtuellen Umgebung selbst. So können Variablen, wie die aktuelle Außentemperatur, der Blutdruck des Probanden usw., wichtig sein. Um auch diese Daten aufzeichnen zu können, besitzt das Toolkit die Funktionalität, externe Hardware anzuschließen und von dieser Daten zu empfangen. Dazu wird eine serielle

Schnittstelle verwendet, die den Bitstrom ausliest und zu lesbaren Werten umwandelt.

Es können nicht alle Geräte Messdaten über den Serial Port senden. Viele Geräte verwenden z. B. eigene Treibersoftware und könnten nicht ohne Weiteres mit einer Unity Anwendung kommunizieren. Die serielle Datenübertragung ist dagegen universell, was es ermöglicht, Skripte zu erstellen, die nicht hardwarespezifisch sind. Beispielsweise können so Sensoren ausgelesen werden, die über einen Arduino an den PC angeschlossen sind. Die ausgelesenen Werte können anschließend zum Beispiel als Parameter von Events verwendet werden.

4.2.1.3 VR-Integration

Das VR-Integrationsmodul enthält Skripte, welche sich auf VR-spezifische Daten und Interaktionen beziehen. Es stellt die Schnittstelle des Toolkits zum SteamVR-Plugin dar.

Eine Funktion dieses Modules ist die Aufzeichnung der Blickrichtung des Spielers. Diese wird anhand der Ausrichtung des VR-Headsets festgestellt und als ein Rotationsvektor ausgegeben. Die genaue Blickrichtung kann über die Rotation jedoch nur ungenau ermittelt werden. Ist eine genauere Feststellung der Blickrichtung erforderlich, kann deshalb Eye-Tracking Hardware verwendet werden.

Erweiternd kann durch die Blickrichtung festgestellt werden, ob der Spieler derzeit einen bestimmten Gegenstand oder Bereich der Umgebung betrachtet. Hierzu wird ein Spherecast von der Position des Headsets aus gestartet. Trifft dieser auf ein Objekt mit dem entsprechenden Skript, gibt dieses Objekt ein Event ab, das anzeigt, dass es derzeit angesehen wird.

Ebenfalls werden über die VR-Controller ausgeübte Aktionen erfasst, wie zum Beispiel das Aufheben von Gegenständen oder die Interaktion mit Schaltern. Hierzu werden Tracking-Skripte erstellt, die sich auf das "SteamVR Input" System beziehen. "SteamVR Input" ist seit 15.05.2018 Teil von SteamVR und ersetzt die alten hardwarespezifischen Skripte. Es ermöglicht dem Nutzer die Verwendung von beliebigen Controllermodellen, ohne dass der Entwickler

manuell Unterstützung für diese einbauen muss (Valve Software, 2018). In Unity lassen sich über “SteamVR Input” Aktionen definieren, die mit den Controllern ausgeführt werden können, wie z. B. “Greifen” oder “Teleport”. Diese Aktionen liest der Aktionstracker aus und versendet sie als Event.

4.2.1.4 Visualisierung

Im Visualisierungsmodul können die im Telemetriemodul gesammelten Daten ausgewertet werden. Das Visualisierungsmodul befindet sich außerhalb der Unity Entwicklungsumgebung. Stattdessen wird R, eine Programmiersprache für statistische Berechnungen und Visualisierung verwendet. R ist Open-Source und lässt sich mit Plugins um zahlreiche Funktionen erweitern, die auch komplexe statistische Auswertungen ermöglichen (The R Foundation, 2019). Das Visualisierungsmodul verwendet die vom Toolkit generierten JSON-Daten und verarbeitet diese zu verschiedenen Visualisierungen und Datenstrukturen. Generell können bei individuellen Experimenten die Methoden der Datenauswertung sehr verschieden sein, weshalb das Visualisierungsmodul hauptsächlich darauf ausgelegt ist, Funktionen anzubieten, welche das Weiterverarbeiten der Daten in R vereinfachen. Außerdem werden für einige Standardevents des Telemetriemoduls Funktionen bereitgestellt, welche diese Events auswerten und visualisieren. Diese Funktionen können auch als Beispiele verstanden werden, wie die JSON-Daten in VR verwertet werden können.

Positions-Heatmap

Eine mögliche Visualisierung ist die Positions-Heatmap, welche auf einer zweidimensionalen Ebene darstellt, an welchen Orten sich ein Objekt während des Experiments wie lange befunden hat. Sie kann z. B. verwendet werden, um zu zeigen, an welchen Orten einer Szene sich ein Proband am meisten aufgehalten hat oder welche Strecke er verwendet hat, um sich zwischen zwei Orten zu bewegen. Der Funktion müssen ein Attribut und die gewünschte Größe des abgedeckten Bereiches übergeben werden. Aus diesen Daten wird dann automatisch eine Heatmap erstellt.

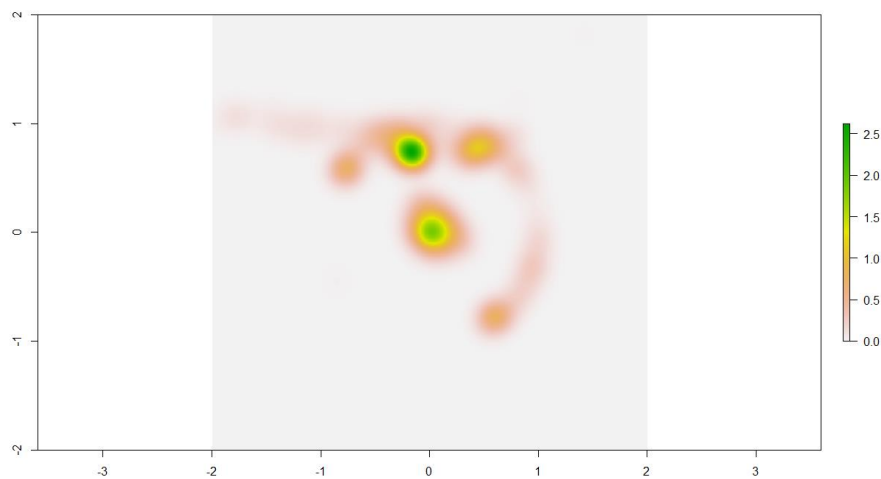


Abbildung 7 Beispiel einer in R generierten Heatmap (Eigene Erstellung)

Objektbetrachtung

Neben der Heatmap wird ein Skript bereitgestellt, welches es ermöglicht, die vom Probanden betrachteten Objekte während des Experiments auszuwerten. Das Skript fasst die entsprechenden Daten zusammen, sortiert sie, und kann schließlich daraus ein Balkendiagramm erstellen, welches zeigt, welche Objekte der Proband am häufigsten betrachtet hat.

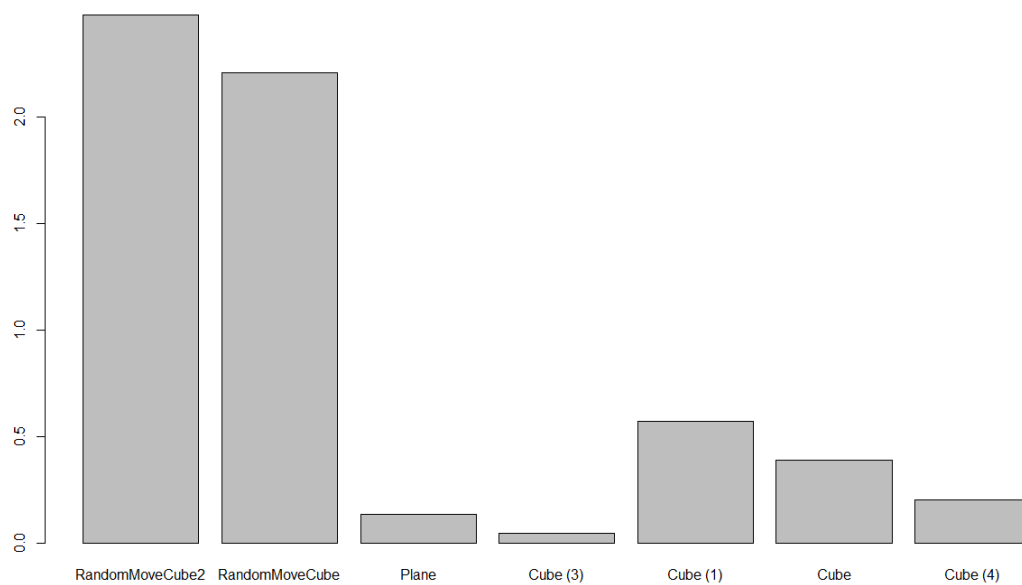


Abbildung 8 Beispiel eines generierten Balkendiagramms zur Objektbetrachtung

4.2.1.5 Wiedergabe

Das Wiedergabemodul ermöglicht es, den Ablauf eines Versuchs wiederherzustellen und innerhalb von Unity zu betrachten. Dazu verwendet es Daten aus dem Telemetriemodul, wie z. B. Transformationsdaten von Objekten. Die Wiedergabe eines vergangenen Versuchs findet nicht im kompilierten Programm, sondern im Unity Editor statt.

Wird eine JSON-Textdatei eingelesen, werden anschließend alle aktiven Skripte in der Szene temporär deaktiviert. Über die Wiedergabesteuerung kann dann zwischen verschiedenen gespeicherten Zeitpunkten des Versuchs gewechselt werden. Die aufgezeichneten GameObjects in der Szene erhalten alle aufgezeichneten öffentlichen Variablenwerte, wie z. B. Positions- und Rotationsdaten, welche sie zu diesem Zeitpunkt während des Experiments hatten. Dadurch kann beispielsweise die Blickrichtung des Probanden zu einem bestimmten Zeitpunkt beobachtet werden, ohne dass der Tester dazu in der Textdatei nachschauen muss. Die Szene kann frei aus allen Blickwinkeln betrachtet werden. Eine Echtzeit-Wiederherstellung des Testablaufs ist ebenfalls möglich. Die Bedienung des Wiedergabemodul wird als Teil des Interfacemoduls umgesetzt.

4.2.1.6 Interface

Das Toolkit erweitert die Oberfläche von Unity um ein eigenes Interface, in dem Benutzer verschiedene Einstellungen vornehmen und Prefabs spawnen können. Die Editorskripte, welche diese Oberfläche erzeugen, bilden das Interface-Modul.

Objekttracking

In der "Track Object" Oberfläche kann der Nutzer ein Objekt in der Szene anwählen, bei welchem er bestimmte Variablen in regelmäßigen Abständen aufzeichnen will. Der Nutzer kann, nachdem er in der Szene auf ein Objekt geklickt hat, zunächst die Komponenten des Objektes auswählen, von denen er Variablen aufzeichnen möchte.

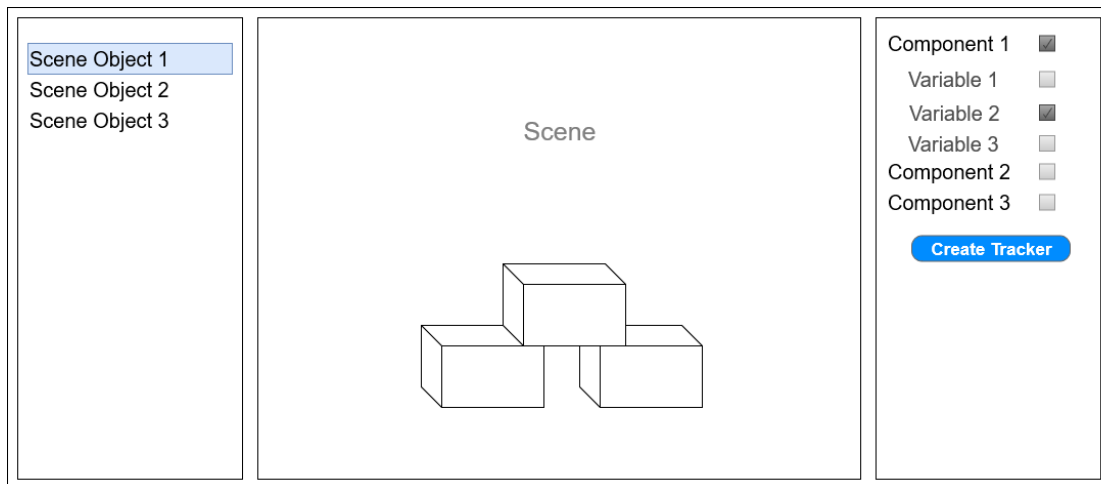


Abbildung 9 Mockup der Track Object Oberfläche (Eigene Erstellung)

Wählt er eine Komponente aus, zeigt das Interface alle zur Aufzeichnung geeigneten öffentlichen Variablen der jeweiligen Komponente an. Der Nutzer kann eine beliebige Anzahl von Variablen anwählen, welche aufgezeichnet werden sollen. Bei einem Klick auf “Create Tracker” wird die Aufzeichnung des Objektes automatisch eingerichtet. Hierzu wird ein Event erstellt, welches die aufzuzeichnenden Variablen als Attribute besitzt. Dem aufzuzeichnenden Objekt wird eine Event Sender Komponente angehängt, welche standardmäßig in einem 1-Sekunden Intervall ein Event mit den aufzuzeichnenden Werten aussendet. Der Nutzer kann diesen Wert anpassen oder auch den Timer komplett deaktivieren, um das Event manuell zu beliebigen Zeitpunkten zu versenden.

Eventerstellung

Neben der automatischen Erstellung über das Tracking-Interface können Events auch manuell definiert werden. Ein Event kann wie ein reguläres Unity Asset über „Create“ im Kontextmenü erstellt werden. Wird ein erstelltes Event angeklickt, können dessen Eigenschaften im Inspektor festgelegt werden. Indem der Nutzer den Namen und Datentyp eines Eventparameters eingibt, wird dieser definiert. Bei automatisch erstellten Events erscheint dieses Interface nicht, damit keine Fehler entstehen, wenn deren Werte manipuliert werden.

Wiedergabesteuerung

Die Wiedergabesteuerung ermöglicht die Kontrolle des Wiedergabemoduls. Es handelt sich um ein Editorfenster, welches verschiedene Steuerungselemente enthält.

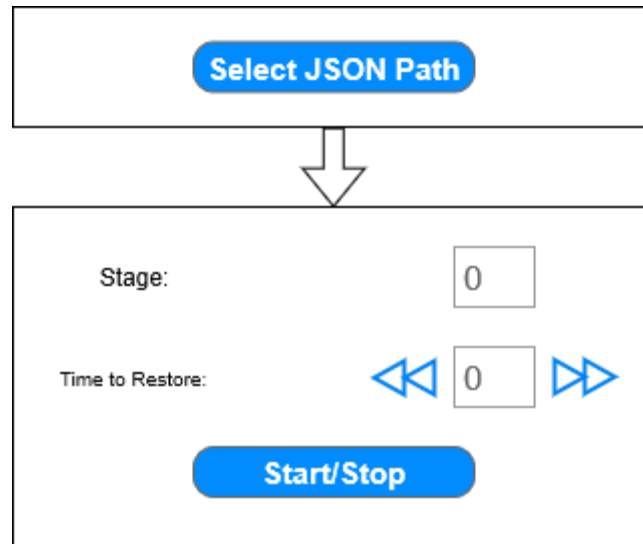


Abbildung 10 Aufbau der Wiedergabesteuerung (Eigene Erstellung)

Um das Wiedergabemodul zu aktivieren, muss der Nutzer zunächst in Unity das Wiedergabefenster öffnen und dann die Szene starten. Anschließend kann er eine JSON-Datei von seiner Festplatte auswählen. Wurde eine JSON Datei ausgewählt, öffnet sich das eigentliche Steuerungsinterface. In diesem kann die wiederzugebende Stage sowie ein Zeitpunkt ausgewählt werden, welcher wiederhergestellt werden soll. Zwei Buttons ermöglichen das Vor- und Zurückspringen um jeweils eine Sekunde. Über den Start/Stop Button kann der Ablauf des Experiments in Echtzeit wiedergegeben werden.

4.3 Bedienungshilfen

Um das Kriterium der Bedienbarkeit zu gewährleisten, besitzt das Toolkit einen Quickstart-Guide sowie eine Beispielszene, welche Nutzern die Bedienung des Toolkits näherbringen sollen.

4.2.1 Beispielszene

Um die Funktionsweise des Toolkits zu demonstrieren und zu erklären, besitzt das Toolkit neben der Dokumentation auch eine Standardszene, in welcher sich eine Art simuliertes Experiment durchführen lässt, welches alle Funktionen des Toolkits nutzt. Das Experiment besteht aus einer Szene mit drei Stages, in denen unterschiedliche Objekte aktiviert werden. Zwei GameObjects, bei welchen es sich um Würfel handelt, die sich zufällig durch die Szene bewegen, werden in den Stages abwechselnd aktiviert. Eine der Stages besitzt außerdem ein Zeitlimit von 3 Sekunden.

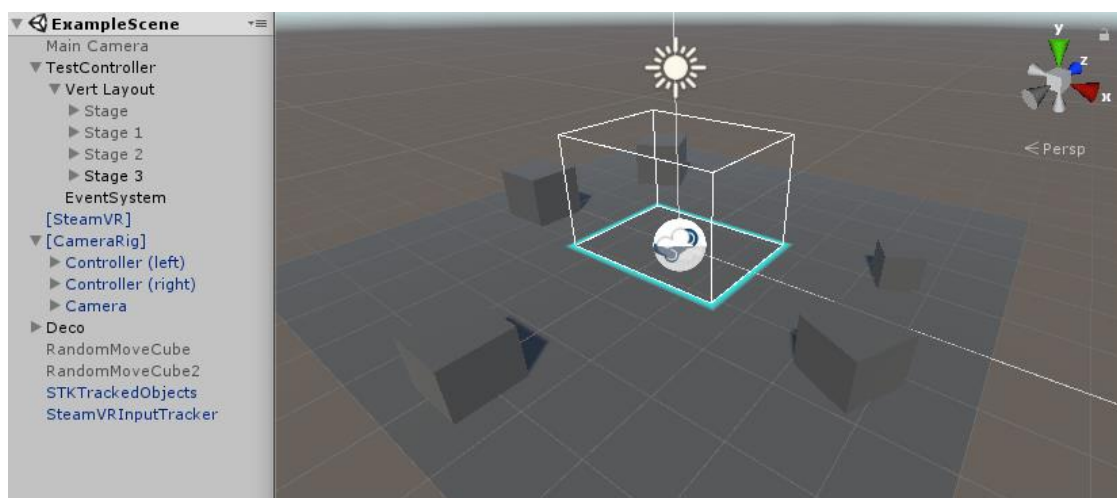


Abbildung 11 Die Beispielszene (Eigene Erstellung)

Auf allen beweglichen Objekten befinden sich bereits Tracker, sodass der Testablauf mit der Wiedergabefunktion komplett wiederhergestellt werden kann. Außerdem werden alle VR-Eingaben sowie die betrachteten Objekte aufgezeichnet.

4.2.2 Quickstart-Anleitung

Bei der Quickstart-Anleitung handelt es sich um eine Gebrauchsanweisung für das Toolkit, welche Nutzern erklärt, wie man das Toolkit verwenden kann, um ein Experiment aufzusetzen. Hierzu durchläuft die Anleitung schrittweise die notwendigen Schritte. Es werden zunächst die notwendigen Vorbereitungen zum Einsatz des Toolkits, wie die Installation des SteamVR-Plugins und dem Einstellen der neuesten .NET-Runtime, erklärt. Anschließend wird nacheinander auf die verschiedenen Funktionen des Toolkits eingegangen und beschrieben, wie diese in eine bestehende Szene einbaut werden können. Mithilfe der Quickstart-Anleitung soll es möglich sein, ein komplettes Experiment aufzusetzen. Die Quickstart-Anleitung befindet sich im Anhang dieser Arbeit.

4.4 Technischer Aufbau

In diesem Kapitel wird beschrieben, wie das Toolkit technisch strukturiert und aufgebaut ist.

4.4.1 Klassen

Im folgenden Kapitel soll mithilfe von Klassendiagrammen und Beschreibungen der Aufbau der verschiedenen im Toolkit umgesetzten Klassen erklärt sowie Zusammenhänge zwischen den Klassen aufgestellt werden. Die Klassen sind nach den im Konzept definierten Modulen aufgeteilt. Bei allen Klassen werden nur die wichtigsten Funktionen und Variablen erklärt. Um den Aufbau der Klassen darzustellen, werden für die C#-Skripte Klassendiagramme nach UML 2.0-Standard verwendet. (Siehe Bell, 2019)

Generell haben Klassen des Toolkits das Kürzel „STK“ vor ihrem Namen, damit sie in Projekten, in denen das Toolkit verwendet wird, leicht von internen Klassen unterschieden werden können.

4.4.1.1 Klassen ohne Modulzugehörigkeit

Einige Klassen des Toolkits erfüllen Funktionen, die sich nicht eindeutig einem der Module aus dem Konzept zuordnen lassen. Hierzu gehören beispielsweise Klassen, die Funktionen aus mehreren Modulen zusammenführen oder die von mehreren Modulen verwendet werden.

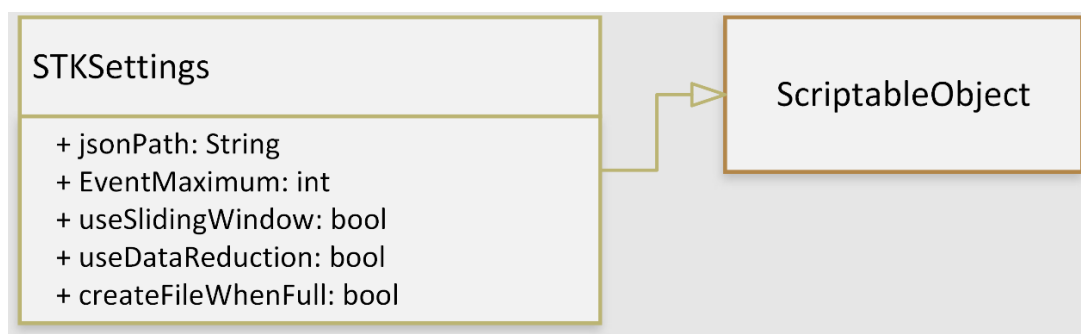


Abbildung 12 Klassendiagramm von STKSettings (Eigene Erstellung)

„**STKSettings**“ speichert projektweite Einstellungen für das Toolkit. Es handelt sich bei der Klasse um ein **ScriptableObject**, von dem sich eine Instanz im

„Resources“ Ordner befindet. In dieser Instanz werden die Variablenwerte bzw. Einstellungen für das Toolkit gespeichert.

In einigen Skripten der verschiedenen Module wird mit den eingebauten Arrays von Unity gearbeitet. Teilweise kann es passieren, dass diese während der Laufzeit geändert werden müssen. Außerdem kann es passieren, dass durch eine Nutzereingabe ein Objekt gelöscht wird, auf das es eine Referenz in einem Array gibt. Diese verweist dann auf null und löst unter Umständen Exceptions aus. Unitys eingebaute Arrays sind außerdem nicht sehr flexibel, weshalb oft längere Schleifen notwendig sind, um mit solchen Situationen umzugehen. Um dies zu vereinfachen, wurde die Klasse **STKArrayTools** erstellt.

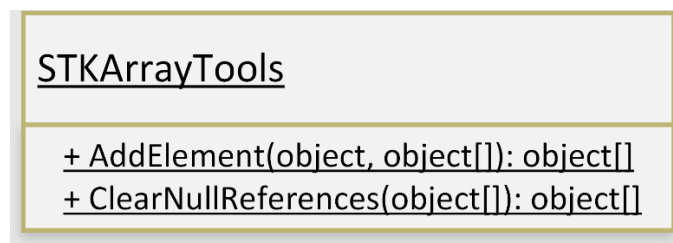


Abbildung 13 Klassendiagramm von *STKArrayTools* (Eigene Erstellung)

„**STKArrayTools**“ ist eine statische Klasse, die zwei Funktionen zur Bearbeitung von Arrays enthält. Mit „AddElement“ kann einem Array ein Element an letzter Stelle angefügt werden. „ClearNullReferences“ durchläuft ein Array und untersucht, ob Elemente des Arrays auf null Verweisen. Wenn ja, werden diese aus dem Array entfernt. Beide Funktionen geben nach Durchlauf das bearbeitete Array zurück.

4.4.1.2 Testaufbaumodul

Das Testaufbaumodul enthält die Skripte, welche die Struktur des Experiments bestimmen, und Steuerungselemente, um dessen Ablauf zu kontrollieren.

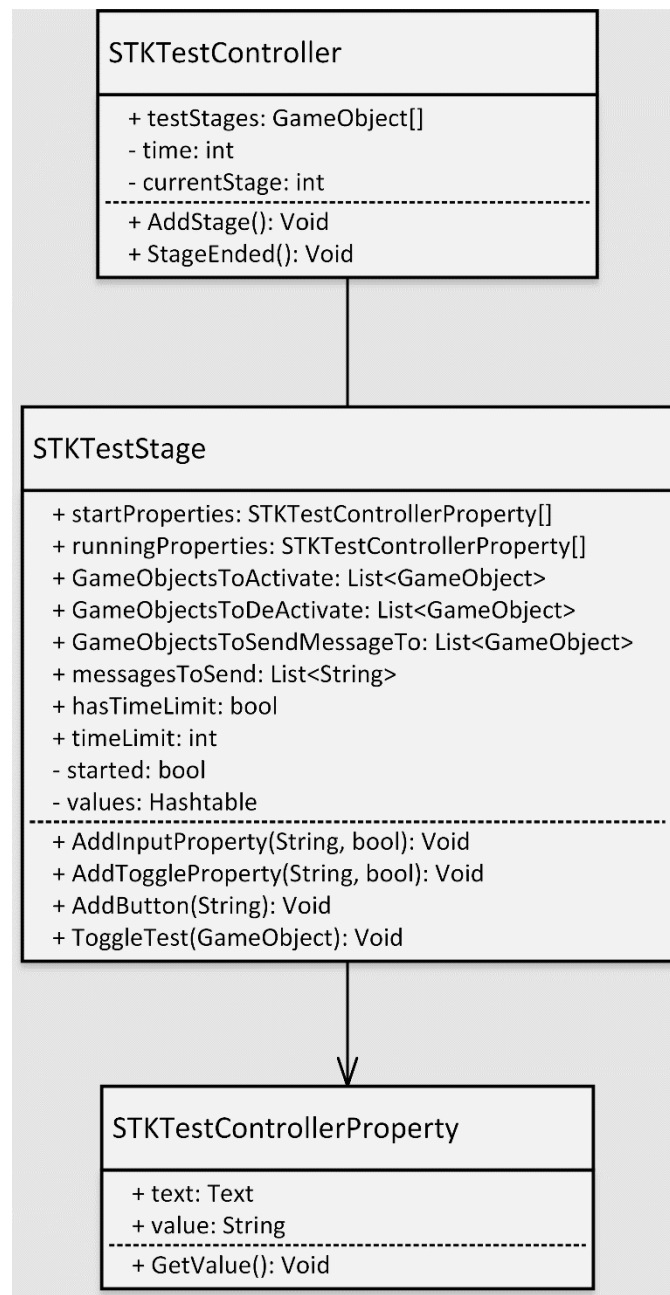


Abbildung 14 Klassendiagramm Testaufbaumodul (Eigene Erstellung)

„**STKTestStage**“ beschreibt einen Abschnitt eines Experiments. In den Variablen „**GameObjectsToActivate**“ und „**GameObjectsToDeActivate**“ befinden sich Referenzen zu Objekten, die beim Start der Stage aktiviert bzw. deaktiviert werden sollen. Die Listen „**GameObjectsToSendMessageTo**“ und

„messagesToSend“ enthalten Referenzen zu GameObjects, bzw. Nachrichten, die diese beim Start erhalten sollen. Die Stage wird gestartet oder beendet, wenn die „ToggleTest“ Funktion ausgeführt wird.

Die Klasse enthält außerdem die Variablen „startProperties“ und „runningProperties“. Hierbei handelt es sich um Attribute, also Werte, die der Nutzer über das Interface vor bzw. während einer bestimmten Stage eingeben kann. Neue Attribute können über die Funktionen „AddInputProperty“ und „AddToggleProperty“ definiert werden. Diese werden über die Klasse **„STKTestControllerProperty“** definiert. Ein Attribut besitzt einen Namen und einen Wert. Dieser Wert wird über ein Eingabefeld vom Nutzer kontrolliert und hat zunächst den Typ String. Bei der späteren Konvertierung ins JSON-Format wird dieser String jedoch darauf untersucht, ob es sich um eine Zahl handelt (Siehe 4.4.1.3 Telemetriemodul).

Die Stages werden alle in der Klasse **„STKTestController“** gesammelt. Diese Klasse hat die Hauptfunktion, zu kontrollieren, welche Stage derzeit aktiv ist. Über sie werden außerdem neue Stages mittels der „AddStage“-Funktion definiert.

4.4.1.3 Telemetriemodul

Im Telemetriemodul befinden sich alle Klassen die im Zusammenhang mit der Aufzeichnung von Daten sowie deren Speicherung stehen.

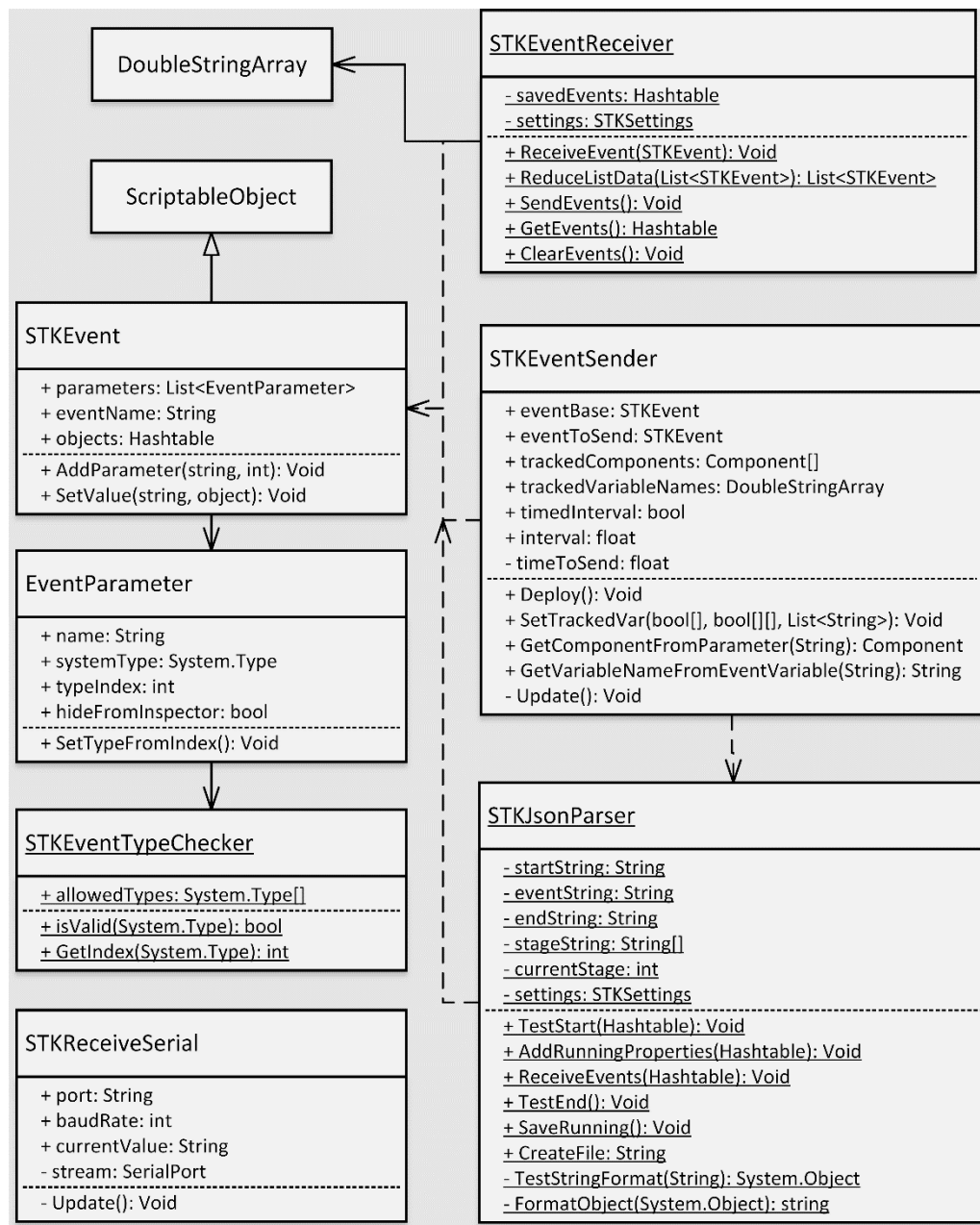


Abbildung 15 Klassendiagramm des Telemetriemoduls (Eigene Erstellung)

In der „**STKEvent**“-Klasse wird der Aufbau eines Events definiert. Ein Event hat ein „Parameters“ Array, in welchem die Namen und die Datentypen der Parameter definiert sind, die das Array erhalten kann. Die tatsächlichen Werte,

die definiert werden, wenn das Event versandt wird, befinden sich im „objects“-Hashtable. Es besitzt als Variablen außerdem einen Timestamp sowie einen Namen, durch den sich das Event von anderen abgrenzen lässt.

Bei der Definition eines Events können die Parameter entweder über den Inspektor oder die Funktion `AddParameter` definiert werden. Letztere wird beispielsweise bei der automatischen Eventerstellung von `STKTrackObject` verwendet (siehe 4.4.1.6 Interfacemodul).

Diese Parameter werden in der „**EventParameter**“-Klasse definiert. Sie enthält den Namen und die Klasse eines Parameters, aber nicht den Wert. Die Klasse besitzt die Funktion „`SetTypeFromIndex`“, welche es ermöglicht, den Typ des Parameters mittels des Indexes im statischen Array „`allowedTypes`“ zu setzen. Dieses ist Teil der Klasse „**STKEventTypeChecker**“. Neben dem Array der möglichen Parametertypen besitzt die Klasse die Funktion „`isValid`“, mit der sich überprüfen lässt, ob eine Variable einen zulässigen Typ besitzt, sowie „`GetIndex`“, mit der man ermitteln kann, an welcher Stelle im Array sich ein bestimmter Typ befindet.

Folgende Variablentypen sind für die Verarbeitung im Toolkit valide: *Integer*, *Float*, *String*, *Boolean*, *Vector2*, *Vector3*, *Vector4* und *Quaternion*. Diese Typen wurden gewählt, weil deren Werte sich ohne Probleme in einer Textdatei darstellen lassen.

Events werden über die Klasse „**STKEventSender**“ mit Werten versehen und schließlich versendet. Über die Boolean „`timedInterval`“ kann der Nutzer festlegen, ob der Eventsender automatisch in einem bestimmten Intervall Events aussendet. Wird ein Event versendet, wird zunächst eine Kopie des Basisevents erstellt. Dieser Kopie werden dann über die „`AddObject`“ Funktion die Variablen übergeben, die aufgezeichnet werden sollen.

Anschließend wird das Event über die „`ReceiveEvent`“-Funktion an den „**STKEventReceiver**“ übergeben. Im Event Receiver werden die Events gespeichert und gesammelt. Hierfür wird ein Hashtable verwendet, der aus Listen von Events besteht.

Am Ende jeder Stage schickt der Event Receiver alle gesammelten Events an die „**STKJSONParser**“-Klasse. Der JSON-Parser ist dafür verantwortlich, aus den gesammelten Daten eine JSON-Syntax zu bilden und diese schließlich in einer Datei zu speichern. Erhält der JSON-Parser einen Hashtable von Events, durchläuft dieser alle Events in der „ReceiveEvents“-Funktion und formatiert deren Parameterwerte über die „FormatObject“-Funktion abhängig von ihrem Format zu korrekten JSON-Strings. Diese Strings werden zu einem „EventString“ zusammengesetzt. Neben Events nimmt der JSON-Parser ebenfalls Attribute vom Test-Controller entgegen. Dies geschieht zum einen, wenn eine Stage über die „TestStart“-Funktion gestartet wird, und zum anderen, wenn eine Stage beendet wird über „AddRunningProperties“. Hierbei entstehen zwei Variablen, „StartString“ und „EndString“, welche am Ende einer Stage mit dem „EventString“ kombiniert und ins „StageString“ Array eingefügt werden. Die Stage-Strings werden am Ende in der „CreateFile“-Funktion zu einem einzigen String kombiniert und schließlich in einer Datei am festgelegten Pfad gespeichert.

„**STKReceiveSerial**“ ist eine weitere Klasse des Telemetriemoduls. Sie verwendet die „System.IO“ Bibliothek von .NET, um Daten von einer seriellen Schnittstelle, also z. B. einem externen Sensor, zu empfangen. Der Nutzer kann den Port und die Baud-Rate festlegen. In der „currentValue“ Variable befindet sich dann der Wert, der in der Update-Methode ausgelesen wird.

4.4.1.4 VR-Integrationsmodul

Im Integrationsmodul befinden sich die Skripte, die die VR-Eingaben sowie die Objektbetrachtung aufzeichnen. Beide Skripte verwenden das Eventsystem und enthalten deshalb Referenzen auf das „STKEventSender“ Skript.

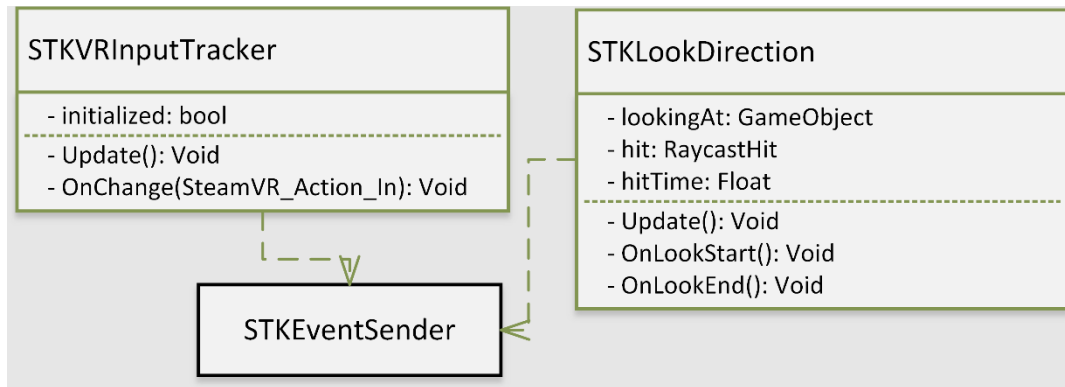


Abbildung 16 Klassendiagramm des VR-Integrationsmoduls (Eigene Erstellung)

„**STKVRInputTracker**“ erstellt Events, wenn der Proband eine Eingabe macht, die vom SteamVR Eingabesystem erfasst wird. Hierzu wird zunächst in der „Update“ Methode darauf gewartet, dass SteamVR Input initialisiert wurde. Ist dies der Fall, werden alle möglichen Input-Aktionen durchlaufen und jeder Aktion wird ein „OnChangeListener“ hinzugefügt. Dies sorgt dafür, dass bei jeder möglichen Eingabe die „OnChange“ Funktion aufgerufen wird. Diese konfiguriert dann über den Eventsender ein Event, welches den Namen und Wert der Aktion enthält.

Die Klasse „**STKLookDirection**“ sendet Events aus, die die Information enthalten, wann und wie lange der Proband ein bestimmtes Objekt betrachtet. Dazu wird in der „Update“ Methode ein Sphercast von der Position des HMDs aus gesendet. In der „OnLookStart“ Funktion wird eine Zeitmessung gestartet, die so lange läuft, bis der Spieler nicht mehr das gleiche Objekt betrachtet. Zu diesem Zeitpunkt wird dann „OnLookEnd“ aufgerufen, welches ein Event konfiguriert und absendet, welches das betrachtete Objekt und die Zeit, wie lange es betrachtet wurde, enthält.

4.4.1.5 Wiedergabemodul

Das Wiedergabemodul enthält alle Skripte, die für die Wiedergabe eines Experiments aus gespeicherten JSON-Daten notwendig sind.

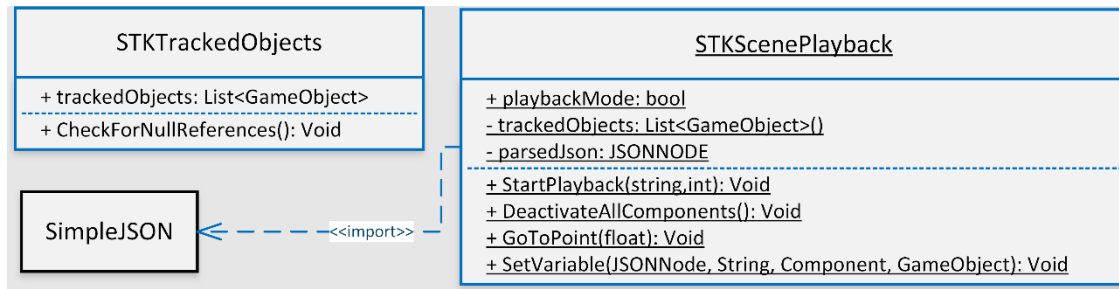


Abbildung 17 Klassendiagramm des Wiedergabemoduls (Eigene Erstellung)

Das Wiedergabemodul ist simpel aufgebaut. Es besteht aus „**STKScenePlayback**“, einer Klasse, welche die Szene zu einem bestimmten Zeitpunkt wiederherstellen kann, und „**STKTrackedObjects**“, einer Klasse, die eine Liste von Objekten in der Szene speichert, die wiederhergestellt werden können. Es wird außerdem die externe Bibliothek „SimpleJSON“ von Markus Göbel verwendet, um Daten aus der JSON Datei in einen in C# lesbaren Datenbaum umzuwandeln (Göbel 2019).

„**STKTrackedObjects**“ liegt auf einem Objekt in der Szene und enthält eine Liste von GameObjects, die Eventsender besitzen und Tracking-Events aussenden. Mittels der Funktion „CheckForNullReferences“ wird die Liste vor dem Start des Wiedergabemoduls auf fehlende Objekte und Objekte, die keinen Eventsender mehr besitzen, überprüft.

„**STKScenePlayback**“ ist eine statische Klasse und bildet den Kern des Wiedergabemoduls. Wird die Szene im Wiedergabemodus gestartet, wird die „DeactivateAllComponents“ Funktion aufgerufen. Diese durchläuft sämtliche Objekte der Szene und deaktiviert deren Skripte. So wird sichergestellt, dass die Szene wiederhergestellt werden kann, ohne dass diese durch laufende Skripte beeinflusst wird.

Wurde vom Nutzer eine JSON-Datei ausgewählt, wird „StartPlayback“ aufgerufen. Diese Funktion liest die JSON-Datei über „SimpleJSON“ aus und speichert die ausgelesenen Informationen in „parsedJSON“.

Wählt der Nutzer einen bestimmten Zeitpunkt zur Wiedergabe aus, wird die „GoToPoint“ Funktion aufgerufen. Für jedes relevante GameObject wird in dieser Funktion das Tracking-Event, das dem gewählten Zeitpunkt am nächsten ist, ausgewählt.

Anschließend werden die Attribute des jeweiligen Events ausgelesen, konvertiert und auf die Variablen des GameObjects angewendet. Dies findet in der „SetVariable“-Funktion statt.

4.4.1.6 Interfacemodul

Da das Interfacemodul die Unity Oberfläche erweitert, kommen hier Editorskripte zum Einsatz. Die Skripte in diesem Modul interagieren nicht miteinander, sondern beziehen sich auf Klassen der anderen Module.

„**STKEventEditor**“ ist ein „Custom Property Drawer“ für die EventParameter Klasse. Diese Klasse gibt also vor, wie die Variablen eines Events im Inspektor aussehen sollen. Es wird abhängig davon, ob der Parameter automatisch gesetzt wurde, zwischen eine Darstellung gewechselt, bei der Name und Typ durch den Nutzer geändert werden können und einer, bei der dies nicht möglich ist.

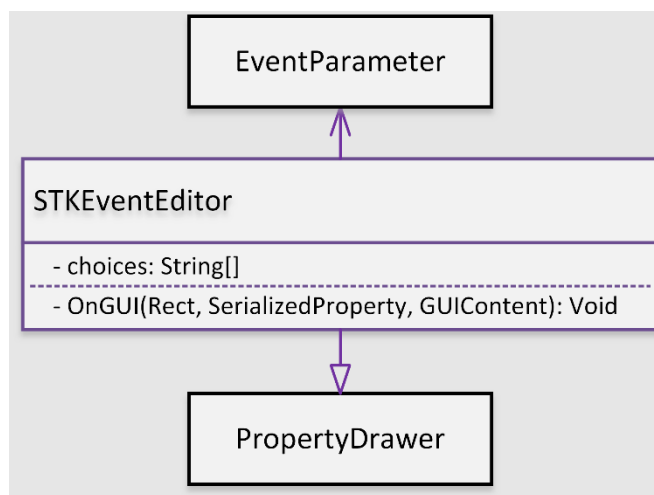


Abbildung 18 Klassendiagramm STKEventEditor (Eigene Erstellung)

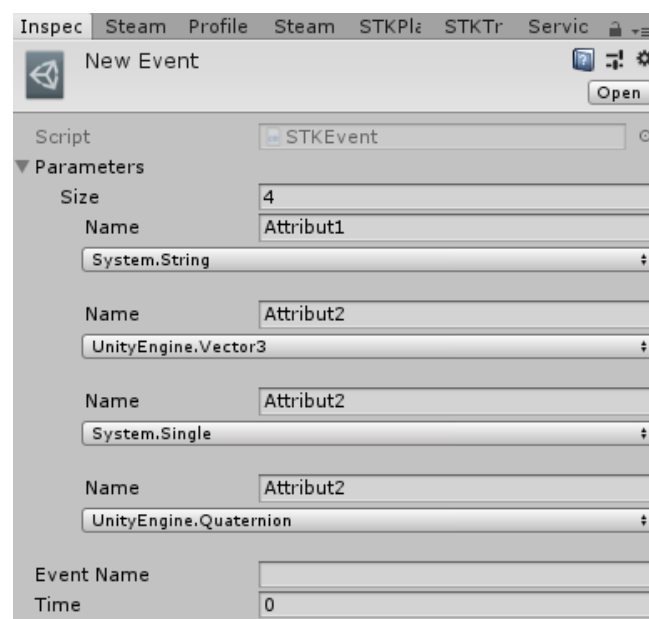


Abbildung 19 Das von Eventparameter erstellte Event Interface (Eigene Erstellung)

Das EditorWindow „**STKTrackEditor**“ ermöglicht es, im Editor Variablen eines GameObjects auszuwählen und anschließend auf Knopfdruck ein passendes Event sowie einen Eventsender zu erstellen, welcher diese Variablen dann in regelmäßigen Abständen abfragt und als Objekte eines Events einsetzt. In der *OnGUI* Funktion werden unter Verwendung von Reflections sämtliche Komponenten und Variablen eines GameObjects abgefragt. Davon werden alle, die einen kompatiblen Datentypen haben, anschließend im Interface angezeigt.

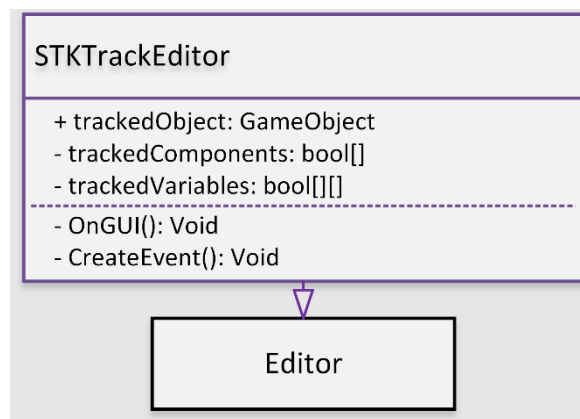


Abbildung 20 Klassendiagramm STKTrackEditor (Eigene Erstellung)

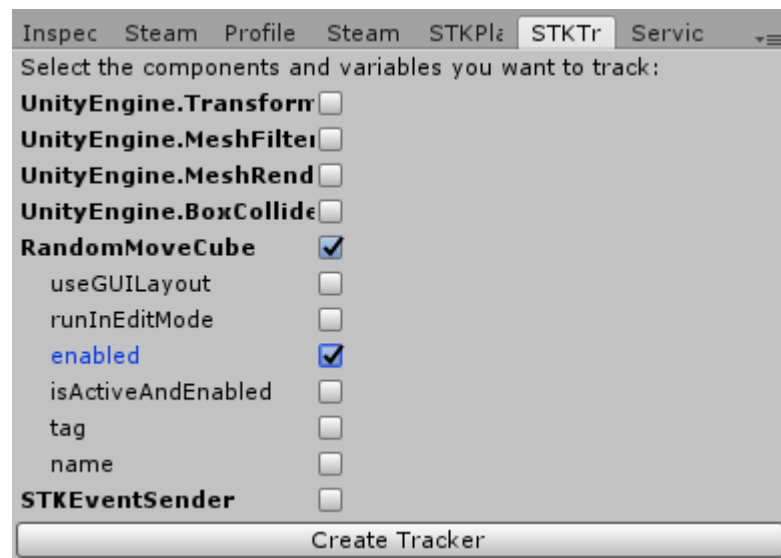


Abbildung 21 Das von STKTrackEditor erstellte Tracking Interface (Eigene Erstellung)

Die *CreateEvent* Funktion erstellt ein neues Event und definiert dessen Parameter auf Basis der vom Nutzer ausgewählten Variablen. Es wird außerdem ein Eventsender erstellt und als Komponente dem gewählten GameObject hinzugefügt.

„**STKTestControllerEditor**“ definiert die Anzeige der „Testcontroller“-Komponente im Inspector. In der OnInspectorGUI Funktion wird das normale Interface durch einen Button „Add Stage“ ersetzt, welcher, wenn er gedrückt wird, ein Stage-Prefab spawnt und eine Referenz darauf im Stages-Array des Testcontrollers ablegt.

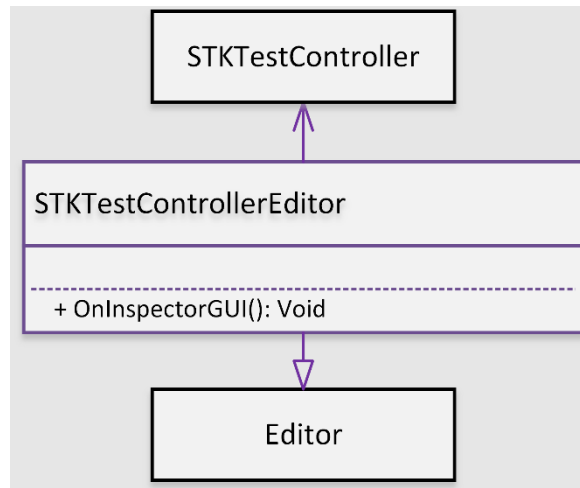


Abbildung 22 Klassendiagramm STKTestController (Eigene Erstellung)

„**STKTestStageEditor**“ ersetzt das Inspektor-Interface von „STKTestStage“. Es erstellt ein Eingabefeld, in dem der Nutzer den Namen eines Attributs eingeben kann. Wurde ein Name eingegeben, werden zwei Buttons angezeigt, die das Erstellen einer startProperty bzw. einer runningProperty ermöglichen.

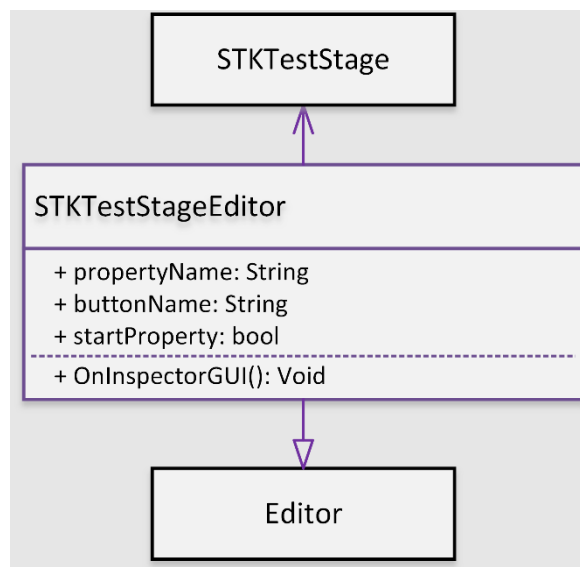


Abbildung 23 Klassendiagramm STKTestStageEditor (Eigene Erstellung)

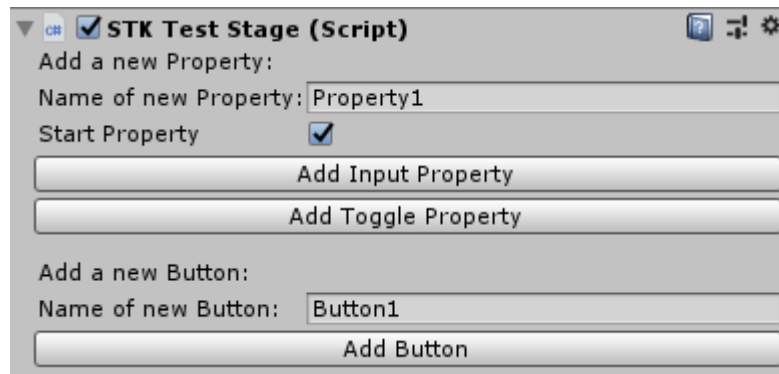


Abbildung 24 Das von STKTestStageEditor erstellte Stage Interface (Eigene Erstellung)

„**STKPlaybackEditor**“ erstellt ein EditorWindow, welches die Steuerung des Wiedergabemoduls ermöglicht. Es werden Interfaceelemente definiert, die die Eingabe eines Pfades und die spätere Kontrolle der Szenenwiedergabe ermöglichen. Die Eingaben werden in Anweisungen für die „STKScenePlayback“-Klasse übersetzt.

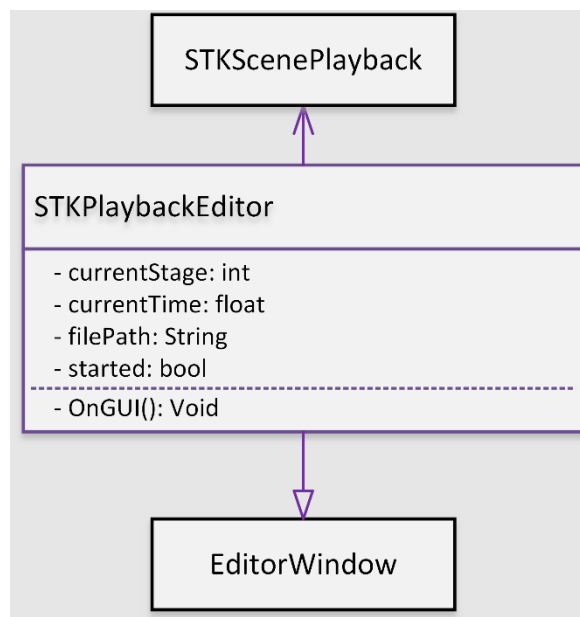


Abbildung 25 Klassendiagramm STKPlaybackEditor (Eigene Erstellung)

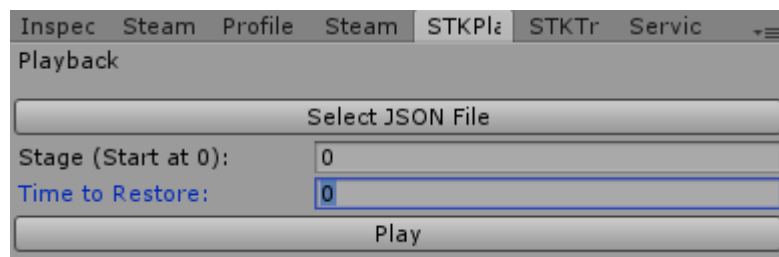


Abbildung 26 Das von STKScenePlayback erstellte Wiedergabe Interface (Eigene Erstellung)

4.4.2 Aufbau des Visualisierungsmoduls

Das Visualisierungsmodul befindet sich außerhalb der Unity-Umgebung und besteht komplett aus Funktionen für die Statistiksprache R.

Für den Import von JSON-Daten verwendet das Toolkit die R-Bibliothek „jsonlite“. Da diese standardmäßig nur einzelne JSON-Dateien importieren kann und der Import eines ganzen Datensatzes dadurch mühsam wäre, wurde die „**SaveFilesToWorkspace**“-Funktion erstellt, welche einen Ordner entgegennimmt und aus diesem sämtliche JSON-Dateien ausliest. Die hieraus entstandenen Daten werden in eine Liste geschrieben und als Variable zurückgegeben.

Mit der Funktion „**CreatePositionalHeatmap**“ lässt sich eine zweidimensionale Heatmap aus Positionsdaten erstellen. Sie nimmt einen Datensatz und eine Größenangabe entgegen und verwendet die „KernSmooth“ sowie „Raster“ Bibliotheken, um daraus eine Heatmap zu erstellen.

Die „**CreateObjectLookGraph**“-Funktion erstellt aus den Objektbetrachtungsdaten ein Balkendiagramm, welches anzeigt, welche Objekte am meisten vom Probanden betrachtet wurden. Es sammelt dafür sämtliche Daten und addiert die zusammengehörenden Zeiten. Die Betrachtungszeiten werden neben dem Balkendiagramm als Variable ausgegeben.

4.5 Aufbau des JSON-Formats

Die Daten aus dem Telemetriemodul werden zunächst in der „STKEventReceiver“-Klasse in einer Hierarchie aus Arrays gespeichert. Am Ende eines Experiments wird diese Hierarchie ins JSON-Format übersetzt. Um das Format zu definieren, musste zunächst festgelegt werden, wie die Probandentests grundlegend aufgebaut sind. Grundsätzlich bestehen im Toolkit erstellte Probandentests immer aus einem oder mehreren Abschnitten, in welchen sich eine Liste von Parametern sowie mehrere Listen von Events befinden (Siehe 4.2.1.1 Testaufbau). Der Aufbau eines Versuchs, wie er im Toolkit erstellt werden kann, lässt sich in einer hierarchischen Struktur wie folgt darstellen:



Diese Hierarchie wurde anschließend ins JSON-Format übertragen:

```
{
  "Stage 0": {
    "Attribut": "Wert",
    "TimeStarted": "HH:MM:SS",
    "DateStarted": "YYYY.MM.DD",
    "Event Typ": [{
      "Time": 61.96576,
      "Attribut": "Wert"
    }]
  }
}
```

Auf dieser Basis aufbauend wurde das STKJsonParser Skript umgesetzt, welches die Unity-internen Event Daten zu JSON konvertiert (Siehe 4.4.1.3 Telemetriemodul).

5 Umsetzung des Toolkits

In diesem Kapitel wird erläutert, welche Besonderheiten und Probleme während der Umsetzung des Toolkits aufgetreten sind und wie diese angegangen wurden. Es wird insbesondere auf Teile des Toolkits eingegangen, die aus verschiedenen Gründen im Vergleich zum ursprünglichen Konzept geändert werden mussten.

5.1 ReceiveSerial Klasse

Unitys Implementierung der Serialport Klasse von .NET ist leider unvollständig. Deshalb können zum Beispiel keine Events eingesetzt werden, die feuern, wenn serielle Daten empfangen werden. Dies hat die Umsetzung deutlich erschwert. Eine Abfrage in der Update Methode über *SerialPort.Readline* war ebenfalls keine gute Lösung, da so das Programm komplett anhalten würde, wenn der Serialport keine Daten sendet. In VR würde das auch bei Verzögerungen von wenigen Millisekunden sichtbare Ruckler verursachen.

Mit dem 2018.2 Update wurde jedoch erstmals die Möglichkeit eingeführt, Unity Projekte auf eine .NET 4.0 äquivalente Runtime zu updaten. Durch die hiermit verfügbare Funktion *BytesToRead* konnte erreicht werden, dass der *ReadLine* Befehl nur ausgeführt wird, wenn es tatsächlich etwas zu lesen gibt. Damit wurde .NET 4.0 allerdings auch zu einer Anforderung, um externe Geräte mit dem Toolkit verwenden zu können. Mit Unity 2018.3 soll die .NET 4.0 Runtime aber ohnehin der Standard für Unity Projekte werden (Peterson, 2018).

5.2 Feststellen welche Objekte betrachtet werden

Prinzipiell kann es in Virtual Reality interessant sein, auszuwerten, was der Proband in der virtuellen Umgebung betrachtet. Deswegen war von Anfang an eine Funktion geplant, die diese Daten aufzeichnet und speichert. Dabei ergaben sich jedoch einige Schwierigkeiten. Eine eindeutige Feststellung, welches Objekt betrachtet wird, lässt sich eigentlich nur über Eye-Tracking Hardware treffen, da der Nutzer nicht zwangsläufig in die exakte Richtung seiner Kopfdrotation schaut. Eine solche Hardware ist jedoch in der Standardausführung der HTC Vive nicht verbaut. Zwar kann eine solche Hardware manuell verbaut werden, aber da dies im Rahmen dieser Thesis nicht praktikabel war, wurde darauf verzichtet und stattdessen die ungenauere Methode über die Rotation des HMDs verwendet. Bei dieser Methode stellte sich die Frage, wie man sie trotz der inhärenten Ungenauigkeit so sinnvoll wie möglich implementieren kann. Anfangs wurden Raycasts verwendet, die von der Mitte des HMDs ausgesendet wurden. Diese Methode hat jedoch die Schwäche, dass sich ein Objekt exakt in der Mitte des Sichtfelds befinden muss, um registriert zu werden. Da man nicht davon ausgehen kann, dass der Proband konstant nur exakt nach vorne schaut, wäre es sinnvoller, wenn ein kleiner Bereich in der Mitte des Sehbereichs auf Kollisionen abgefragt wird, um eine bessere Abschätzung zu erhalten. Deswegen wurden stattdessen Spherecasts mit einem Durchmesser von 10cm verwendet, die statt einem einzelnen Strahl einen kugelförmigen Collider aussenden. Dadurch werden Objekte, die nur um wenige Grad von der HMD-Richtung entfernt sind, ebenfalls erfasst.

5.3 Performance beim Erstellen der JSON-Strings

Nach der Fertigstellung der STKJsonFormatter-Klasse wurde das Telemetriemodul auf seine Performance getestet. Probleme ergaben sich hierbei während der Generierung des JSON Strings nach Abschluss einer Stage. Waren mehrere Tracker-Eventsender aktiv, so betrug die Anzahl an Events schon nach einigen Minuten mehrere Tausend. Der JSON-Generator hatte Schwierigkeiten diese Menge zu verarbeiten, weshalb es oft sekunden- oder sogar minutenlang dauerte, bis der JSON-String fertig generiert war. Diese Situation war nicht akzeptabel, da in dieser Zeit auch das Bild im VR-HMD einfro. Um dieses Problem zu lösen, wurde zunächst der Ansatz getestet, die JSON-Formatierung in einen separaten Thread auszulagern. Viele der Funktionen konnten jedoch nicht in einem anderen Thread ausgeführt werden, ohne sie komplett umzuschreiben. Letztendlich musste der Code darauf untersucht werden, welche Funktion oder welcher Befehl für die Performance-Probleme verantwortlich war. Es konnte festgestellt werden, dass es die String-Additionen waren, welche zu viel Performance kosteten. Diese wurden zunächst auf ein Minimum an Additionsooperationen reduziert, was jedoch nicht ausreichte. Den Durchbruch brachte schließlich die „StringBuilder“-Klasse von .NET, zusammen mit dem „Append“ Befehl. Wurde dieser statt einer einfachen Addition verwendet, konnte der JSON-String deutlich performanter aufgebaut werden, sodass es jetzt selbst bei großen Eventmengen weniger als eine Sekunde dauert, bis der JSON-String generiert ist.

5.4 Aufzeichnung der Temperaturdaten

Um die Anbindung von externen Geräten zu testen, wurde ein Arduino Uno-äquivalentes Board (ELEGOO Uno R3) sowie ein digitaler Temperatur- und Luftfeuchtigkeitssensor verwendet. Dieser wurde auch später im Probandentest eingesetzt, sowie testweise im Pseudexperiment eingebaut.

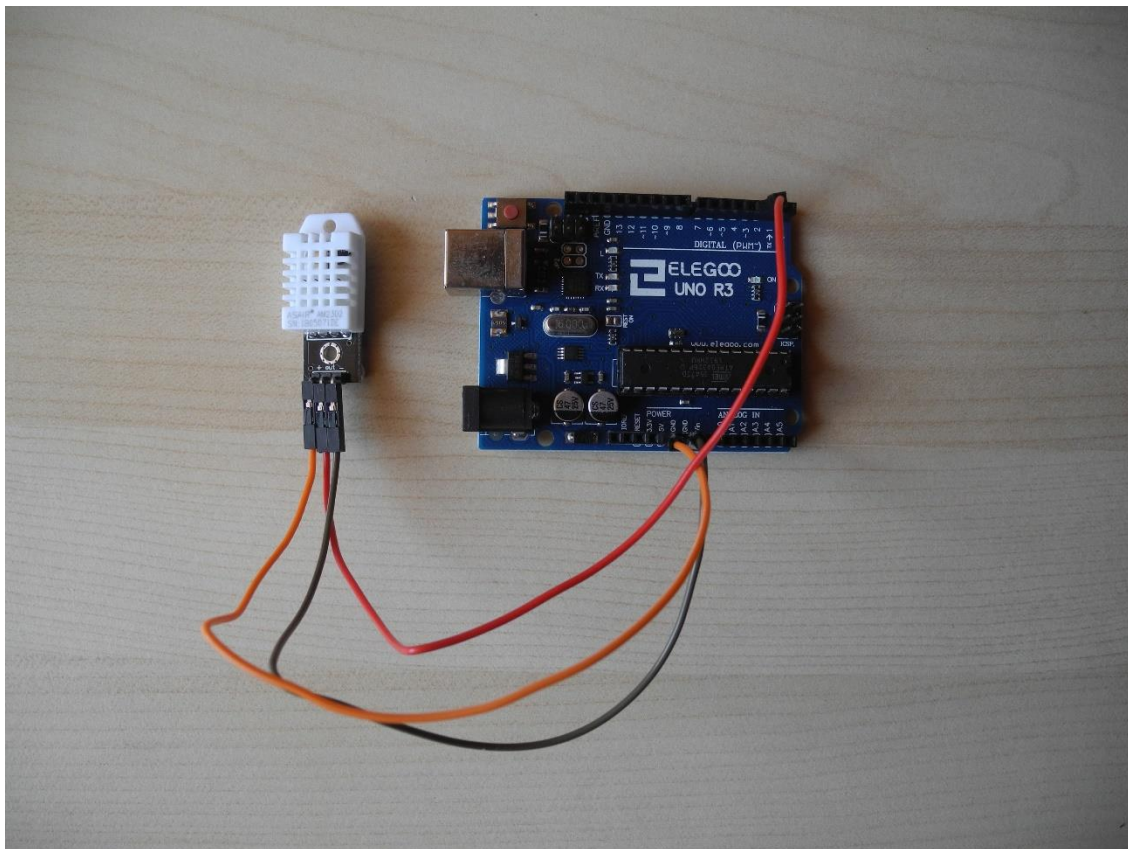


Abbildung 27 Arduino Board mit angeschlossenem Temperatursensor (Eigene Erstellung)

Der Arduino wurde mit einem simplen Skript bespielt, welches alle zwei Sekunden die aktuelle Temperatur und Luftfeuchtigkeit über den COM4-Port sendet. Temperatur und Luftfeuchtigkeit werden in einem gemeinsamen String gesendet und zur Unterscheidung durch den Buchstaben „H“ getrennt. Zum Beispiel ergäbe sich bei einer Temperatur von 42°C und einer Luftfeuchtigkeit von 21% der String „42H21“.

5.5 Codedokumentation

Die Dokumentation des Programmcodes war zunächst in Form von Kommentaren und dem Kapitel „Technischer Aufbau“ geplant und wurde auch so umgesetzt. Während der Evaluation kam jedoch der Vorschlag, den Code mittels eines Dokumentationstools umfassender zu dokumentieren, um sicherzustellen, dass auch weitere Personen später problemlos das Toolkit erweitern können. Aus diesem Grund wurde noch kurz vor Abschluss der Arbeit eine HTML-basierte Dokumentation erstellt. Hierfür wurde das Tool „Sandcastle Help File Builder“ (SHFB) von EWSoftware und Microsoft verwendet. SHFB ist ein Plugin für Visual Studio und erstellt automatisch eine Dokumentation in einem Format nach Wahl. Als Basis nutzt es hierfür XML-basierte Kommentare im Code, welche Klassen, Funktionen und öffentliche Variablen beschreiben (EWSoftware, 2019).

Der Code enthielt bereits zuvor Kommentare, welche die Funktionsweise von den meisten Klassen und Funktionen beschrieben, also mussten diese Kommentare lediglich zu XML-Kommentaren umgewandelt werden, damit sie in die Generierung der Dokumentation miteinbezogen wurden. Hieraus konnte eine HTML-basierte Codedokumentation exportiert werden.

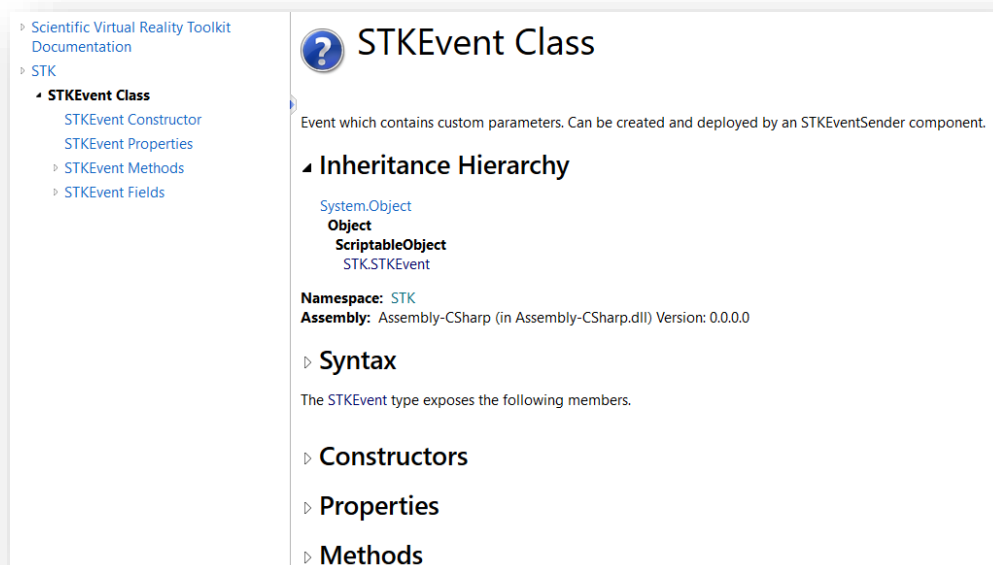


Abbildung 28 Dokumentation der STKEvent Klasse (Eigene Erstellung)

6 Evaluation des Toolkits

Nachdem zuvor die Konzeption und Umsetzung des Toolkits dargestellt wurden, soll in diesem Kapitel nun die entstandene Software evaluiert werden. Hierfür wurde das Toolkit in einem von dieser Arbeit unabhängigen Probandentest praktisch eingesetzt. Nach dem Probandentest wurde ein weiterer Test in Form eines Pseudoexperiments durchgeführt. Aus diesen Einsätzen des Toolkits wurden Schlüsse gezogen, wie gut das Toolkit im realen Einsatz funktioniert und an welchen Stellen es noch verbessert werden könnte.

6.1 Methodik

Ziel der Evaluation war es, allgemeines Feedback über das Toolkit zu sammeln und dabei insbesondere auf die Erfüllung der Erfolgskriterien zu achten. Den Testern wurde hierfür das Toolkit als Unity Package sowie eine Kopie der Quickstart-Anleitung zur Verfügung gestellt. Um festzustellen, wie gut das Toolkit während des Aufbaus und der Durchführung der jeweiligen Experimente verwendet werden konnte, wurden die beiden Testpersonen bei der Umsetzung ihrer Umgebungen beobachtet. Dabei wurde auch verbal Feedback gesammelt. Nach Abschluss der Experimente wurde den Testern ein Fragebogen vorgelegt (Siehe Anhang: Fragenkatalog), in welchem sie die verschiedenen Aspekte des Toolkits bewerten und auf entstandene Probleme sowie Verbesserungsmöglichkeiten hinweisen konnten.

6.2 Einsatz des Toolkits in einem Probandentest

Für die Evaluation in Hinsicht auf die Zielkriterien wurde das Toolkit in einem wissenschaftlichen Probandentest zur Datenaufzeichnung und Steuerung verwendet. Hierbei handelte es sich um einen mehrtägigen Test, der im Rahmen von Bianca Schneiders Masterthesis „Temperaturwahrnehmung in Virtual Reality und ihre Auswirkungen auf die Glaubwürdigkeit von Materialien“ (Schneider 2019) durchgeführt wurde. In diesem Test wurden Probanden im ersten Schritt verschiedene Umgebungen in VR gezeigt, deren Temperaturen sie schätzen sollten. Anschließend wurden ihnen nacheinander

sechs verschiedene Würfel vorgelegt, wovon jeweils drei aus Acryl und Aluminium bestanden. Diese wurden auf bestimmte Temperaturen reguliert. In VR wurde immer ein Granitwürfel gezeigt. Die Probanden sollten die einzelnen Würfel berühren und schätzen, welche Temperatur sie besitzen. Außerdem wurde gefragt, wie glaubwürdig das physische Material im Vergleich zur gezeigten Textur war. Außerdem sollte herausgefunden werden, ob sich durch die Veränderung von Objekttemperaturen die Glaubwürdigkeit von virtuellen Materialien verändert (Schneider 2019). Das Experiment fand im Januar 2019 statt.

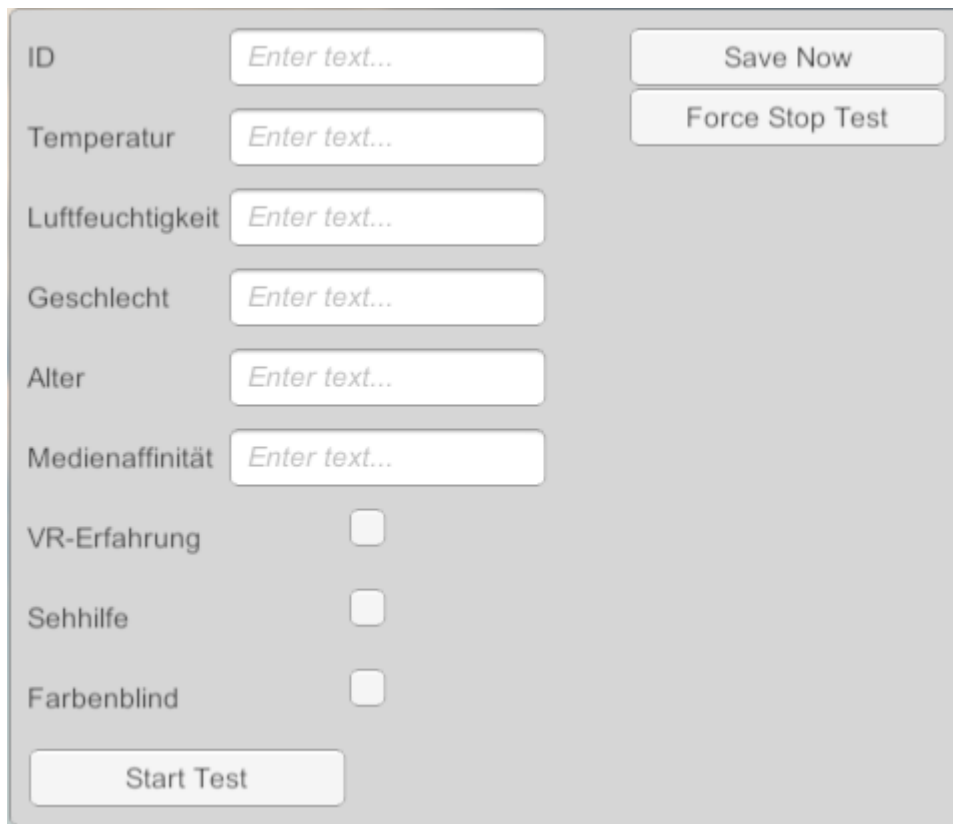
6.2.1 Einbau des Toolkits in die Testsoftware

Das Toolkit wurde zunächst bei einem Treffen mit Bianca Schneider am 19.12.2018 in die Testsoftware eingebaut. Der Aufbau des Experiments bestand aus insgesamt vier Umgebungen, zwischen denen gewechselt wurde. Zunächst wurden dem Probanden nacheinander drei Umgebungen gezeigt: Ein Wald, eine Wüste und eine Winterlandschaft. In der letzten Umgebung, dem neutralen Raum, fanden sechs verschiedene Schritte des Experiments statt, in denen der Proband die Temperatur jedes Würfels abschätzen sollte und nach dessen Glaubwürdigkeit gefragt wurde. Es gab also insgesamt 9 Abschnitte des Experiments, die als Stages im Toolkit umgesetzt werden sollten.

Bei der Diskussion des Experimentablaufs und der zu analysierenden Daten ergab sich schnell, dass einige Funktionen des Toolkits, wie Objekttracking und das Wiedergabemodul, nicht für die Durchführung des Probandentests notwendig oder sinnvoll waren. Deshalb konnten mithilfe des Probandentests nicht alle Funktionen des Toolkits getestet werden.

Das Toolkit sollte von Bianca Schneider weitestgehend selbstständig eingebaut werden. Hierfür erhielt sie eine erste Version der Kurzanleitung, welche die grundlegenden Funktionen und deren Implementation erklärt. Die Anleitung war jedoch zum Zeitpunkt des Einbaus noch sehr unvollständig, weshalb viele der Funktionen erklärt werden mussten. Mit Erklärungen konnten die Funktionen aber größtenteils problemlos eingebaut werden. Beim Einbau

wurden noch einige Bugs im Toolkit festgestellt. Beispielsweise funktionierte das Feature, am Anfang einer Stage eine Funktion auszuführen, nicht.



The image shows a web-based form for data entry. It features six text input fields with placeholder text 'Enter text...' for 'ID', 'Temperatur', 'Luftfeuchtigkeit', 'Geschlecht', 'Alter', and 'Medienaffinität'. To the right of these fields are two buttons: 'Save Now' and 'Force Stop Test'. Below the text fields are three checkboxes for 'VR-Erfahrung', 'Sehhilfe', and 'Farbenblind'. At the bottom left is a 'Start Test' button.

Abbildung 29 Ein vom Toolkit generiertes Interface, welches für das Experiment verwendet wurde. (Bianca Schneider/Eigene Erstellung)

In den darauffolgenden Wochen wurden die gefundenen Bugs und Kritikpunkte nach und nach im Programmcode behoben. Ende Dezember hatte sich der Fragenkatalog für den Probandentest noch einmal geändert. Aus diesem Grund wurde das Experiment mit der neuesten Version des Toolkits neu aufgesetzt. Die Aufteilung der Testabschnitte konnte in der neuen Version deutlich einfacher gestaltet werden, da die Funktion hinzugefügt wurde, Attribute in den Testcontroller einzufügen, welche während des Ablaufs eines Testabschnitts eingegeben werden können. Zuvor waren Eingaben nur möglich, bevor der Start-Button gedrückt wurde, was dazu führte, dass eine hohe Anzahl an Stages notwendig war.

6.2.2 Durchführung des Probandentests

Der Probandentest wurde am 08.01.19, 09.01.19 und 11.01.19 im „Smartspace“ Labor der Hochschule Furtwangen durchgeführt. Insgesamt wurden 33 Probanden getestet, an jedem Tag waren es 11 Probanden.



Abbildung 30 Durchführung eines Probandentests (Eigene Erstellung)

Die Eingabe der Daten über die Testcontroller Oberfläche funktionierte weitestgehend problemlos. Ein Problem, das sich nach einigen Durchläufen ergab, war die Schwierigkeit, fehlerhafte Eingaben später zu korrigieren. Beispielsweise wurde manchmal versehentlich die nächste Stage gestartet, bevor die Hauttemperatur des Probanden eingetragen wurde. Die einzige Möglichkeit, dies zu beheben, war eine nachträgliche manuelle Änderung der JSON-Datei. Aus diesem Umstand ergab sich die Idee, dass eine Rückgängig-Funktion, die zur vorherigen Stage springt, sehr nützlich wäre.

Alle eingegebenen und gemessenen Daten wurden erfolgreich in JSON-Dateien gespeichert. Bei späterer Auswertung ergab sich jedoch das Problem, dass die

einggegebenen Daten alle als Strings formatiert waren. Dies ist für die anschließende Analyse in R unpraktisch, wenn zum Beispiel Durchschnittswerte von Zahlen gebildet werden sollen, die aber als Strings markiert sind. Eigentlich war eine Funktion, die die Daten automatisch formatiert, im Toolkit vorhanden, aber diese wurde nicht korrekt aufgerufen.

<pre>"Stage0":{ "TimeStarted": "14:55:29", "DateStarted": "2019.1.8", "Alter": "25", "ID": "7", "Temperatur": "16", "VR-Erfahrung": "True", "Sehhilfe": "True", "Medienaffinität": "5", "Raumtemperatur": "16", "Luftfeuchtigkeit": "39.90", "Geschlecht": "Weiblich", "Handtemperatur": "31", "Farbenblind": "False", "Umgebung": "Wüste", "Gefühlte Temperatur": "40"}</pre>	<pre>"Stage0":{ "TimeStarted": "14:55:29", "DateStarted": "2019.1.8", "Alter": 25, "ID": 7, "Temperatur": 16, "VR-Erfahrung": "TRUE", "Sehhilfe": "TRUE", "Medienaffinität": 5, "Raumtemperatur": 16, "Luftfeuchtigkeit": 39.90, "Geschlecht": "Weiblich", "Handtemperatur": 31, "Farbenblind": "FALSE", "Umgebung": "Wüste", "Gefühlte Temperatur": 40}</pre>
--	--

Abbildung 31 Die falsche generierte Formatierung (links) und die korrigierte Formatierung (rechts) der JSON-Datei (Eigene Erstellung)

Um die Daten nachträglich zu korrigieren, wurde deshalb eine R-Funktion geschrieben, welche eine aus einer JSON-Datei generierte Liste annimmt und die Daten in dieser automatisch in das korrekte Format konvertiert. Nach den Tests wurde der ursprüngliche Fehler im Toolkit ebenfalls korrigiert.

Beim Import der Daten in R wurde außerdem schnell deutlich, dass es zu aufwendig war, alle Dateien einzeln zu importieren. Um den Aufwand des Imports in R zu minimieren, wurde deshalb eine Funktion erstellt, die alle JSON-Dateien in einem Ordner einliest und daraus eine Liste erstellt.

6.2.3 Befragung nach Durchführung des Probandentests

Einige Wochen nach der Durchführung des Experiments wurde ein Fragebogen aufgesetzt, um detailliertes Feedback zum Einsatz des Toolkits im Experiment zu erhalten. Der Fragebogen soll die Nutzer systematisch nach ihren Eindrücken

und Erfahrungen mit dem Toolkit befragen (Herczeg, 2018, S. 263). Hierfür wurde ein Fragekatalog erstellt, mit dem insbesondere herausgefunden werden soll, ob und inwieweit die zuvor definierten Kriterien erfüllt werden konnten. Außerdem sollten mit dem Fragebogen Ideen gesammelt werden, wie das Toolkit in Zukunft erweitert und verbessert werden könnte. Fragen zu Elementen, die im Experiment letztendlich nicht zum Einsatz kamen, wie z. B. Tracking Events, wurden bewusst ausgelassen. Die Fragen wurden schriftlich über eine Google Forms-Umfrage gestellt. Es wurde, da es sich um eine qualitative Befragung ähnlich eines Experteninterviews handelt, zu einem großen Teil auf Freitext-Fragen gesetzt. Außerdem wurden den Fragen Kategorien zugeordnet, welche anzeigen sollen, auf welche Kriterien sich die Frage bezieht. Neben den zuvor definierten Kriterien gibt es außerdem die Kategorie „Feedback“ für Fragen, welche allgemeines Feedback abfragen, das sich nicht zwingend auf ein einzelnes Kriterium beziehen muss. Die gestellten Fragen und deren Antworten finden sich im Anhang unter „Fragekatalog“.

6.2.3.1 Auswertung der Antworten von Bianca Schneider

Für die Auswertung der Interviewantworten wurde die Methode der qualitativen Inhaltsanalyse verwendet. Hierfür werden in diesem Kapitel die Antworten von Bianca Schneider zitiert und auf deren Bedeutung im Hinblick auf die Evaluation des Toolkits untersucht. Fragen, bei denen keine Antwort gegeben wurde, werden nicht aufgelistet. Bei längeren Textantworten werden Antwortteile, welche für ein bestimmtes Kriterium relevant sind, farblich hervorgehoben. Fragen, welche nicht beantwortet wurden, werden in diesem Abschnitt ausgelassen.

Frage 1: *Wie nützlich war das Testcontroller-Interface, um den Ablauf des Experimentes zu steuern? (1 = Sehr nützlich, 6 = Überhaupt nicht nützlich)*

Antwort: 1

Das Testcontroller-Interface wurde von Bianca Schneider als sehr nützlich empfunden. Dies lässt sich sowohl als positives Feedback für das Toolkit allgemein interpretieren, als auch für das Kriterium der Bedienung.

Frage 2: Sind zu irgendeinem Zeitpunkt während des Tests Performanceprobleme (Ruckler, Abstürze) aufgetreten, welche vermutlich auf das Toolkit zurückzuführen waren?

Antwort: Nein

Aus der Frage 2 lässt sich ablesen, dass während des Tests keine auffälligen Performanceprobleme aufgetreten sind.

Frage 4: Traten während des Experiments zu irgendeinem Zeitpunkt Fehler (Fehlermeldungen, Bugs) auf, die auf das Toolkit zurückzuführen waren?

Antwort: Nein

Ebenfalls wurden während des Tests keine Fehler des Toolkits festgestellt. Da Fehlermeldungen auch durch Performanceprobleme wie Memory Leaks usw. entstehen können, lässt sich diese Antwort auch als positives Feedback für die Erfüllung des Performance-Kriteriums deuten.

Frage 6: War die automatisch generierte JSON-Formatierung für die anschließende Analyse der Daten nützlich? (1 = Ja, sehr; 6 = Nein, überhaupt nicht)

Antwort: 4

Frage 7: Auf welche Schwierigkeiten bist du bei der Verarbeitung der JSON-Daten gestoßen?

Antwort: Die Verarbeitung der Daten war eher auf R ausgelegt. Da ich aber nicht über gute Programmierkenntnisse verfüge, bin ich damit nicht zurechtgekommen.

Daher entschied ich mich für die Auswertung für eine Kombination aus Excel und Tableau Public. Hier traten aber teilweise Import-Fehler auf und in Tableau wurden die Daten nicht so angeordnet wie im JSON-Format. Insgesamt ist Tableau für einfache Übersichten gut geeignet, für komplexere Visualisierungen aber nicht. In Excel konnte aber recht gut mit den Daten gearbeitet werden.

Kodierung: Bedienbarkeit, Feedback

Im Hinblick auf die Nützlichkeit der JSON-Daten antwortet Bianca Schneider mit einer 4, also einer eher geringen Nützlichkeit. Die Entscheidung, nicht R zu

verwenden, zeigt, dass R möglicherweise wegen der Programmierlastigkeit und hohen Einarbeitungszeit die Bedienbarkeit des Toolkits einschränkt. Ebenfalls zeigt sich, dass das JSON-Format, obwohl es syntaktisch korrekt formatiert war, von Programmen unterschiedlich interpretiert wird und Datensätze möglicherweise nach dem Import noch stark bearbeitet werden müssen. In einer möglichen Erweiterung des Toolkits sollte überlegt werden, ob die JSON-Formatierung noch insoweit optimiert werden kann, dass möglichst viele Programme optimal damit umgehen können. Denkbar wäre auch der Umstieg auf ein anderes Format wie CSV.

Frage 8: Findest du das Toolkit insgesamt nützlich für die Durchführung von Experimenten in VR? (1 = Ja, sehr nützlich; 6 = Nein, überhaupt nicht nützlich)

Antwort: 1

Frage 9: Welche Funktionen und Aspekte des Toolkits fandest du am nützlichsten?

Antwort: Ich finde das Toolkit sehr praktisch. Auch ohne Programmierkenntnisse schafft man es, in sehr kurzer Zeit eine Umgebung einzurichten. Wahrscheinlich hätte ich ohne das Toolkit weitaus länger gebraucht, meine Experimente einzurichten. Änderungen kann man gut nachholen und ein Fragebogenformular ist auch sehr schnell erstellt. Das meiste ist leicht verständlich, was man nicht sofort versteht kann man nachlesen. Die Tracking-Funktion, mit der man später die Umgebung nochmal betrachten kann (auch wenn ich sie nicht benötigt habe), empfinde ich als sehr nützlich.

Kodierung: Bedienbarkeit, Feedback

Frage 10: Würdest du, wenn du ein weiteres VR-Experiment aufsetzen musst, erneut das Toolkit verwenden?

Antwort: Ja

Das Feedback zur Nützlichkeit des Toolkits selbst ist positiv ausgefallen. Das Toolkit wurde insgesamt als praktisch empfunden. In Bezug auf die Bedienbarkeit merkte Bianca Schneider an, dass keine Programmierkenntnisse

erforderlich waren, um die Umgebung einzurichten. Außerdem bezeichnet sie die Bedienung als schnell und leicht verständlich.

Frage 7: Welche Aspekte des Toolkits sollten deiner Meinung nach verbessert werden?

Antwort: Es wäre gut, wenn man während des Tests einen Schritt zurückgehen könnte. Dabei sollten aber alle Eingaben gespeichert bleiben. Während der Tests kam es bei mir häufiger vor, dass ich versehentlich die nächste Stage gestartet habe. Vergessene Eingaben konnte ich dann erst in der JSON-Datei einfügen. Das war etwas umständlich. Während des 2. Experiments sahen die Probanden immer die gleiche Umgebung. Beim Wechsel in die neue Stage wurde dieses Bild aber ausgeblendet (er sah die Skybox). Das hat die Probanden teilweise verwirrt. Das sollte behoben werden.

Kodierung: Bedienbarkeit, Feedback

In Bezug auf Verbesserungsmöglichkeiten wurden zwei Punkte genannt. Zum einen wurde eine Rückgängig-Funktion vorgeschlagen, welche das Problem fehlerhafter oder vergessener Eingaben im Testcontroller beheben würde. Außerdem wurde kritisiert, dass Probanden im Testablauf immer wieder in eine leere Umgebung zurückgeworfen wurden. Dies lag daran, dass die verschiedenen Umgebungen aktiviert wurden, wenn eine Stage aktiv war. War gerade keine Stage aktiv, wurde deshalb nur eine Skybox angezeigt. Dieses Problem ließe sich unter Umständen auch dadurch beheben, die Umgebungen über eine Message zu aktivieren statt über die aktivierbaren Objekte der Stage. Alternativ könnte eine neutrale Umgebung verwendet werden, welche zwischen den Stages erscheint.

Frage 8: Welche Funktionen, die bisher nicht im Toolkit existieren, fändest du noch nützlich?

Antwort: Weitere Eingabemöglichkeiten wären hilfreich, z. B. Antwortboxen, in denen ich mehr sehen kann als nur die letzten paar Zeichen. Eventuell wäre es auch geschickt, wenn mehrere Antwortmöglichkeiten (Kästen), die sich auf eine Frage

beziehen, nebeneinander angebracht werden könnten. Also ein Verschieben bzw. Positionieren der Fragen und Antworten. Praktisch wäre auch ein Visualisierungs- und Auswertungstool, welches ein externes Programm wie R unnötig macht.

Die letzte Frage war dazu da, Ideen zu sammeln, wie das Toolkit in Zukunft weiterentwickelt und erweitert werden könnte. Bianca Schneider schlug hier vor, den Testcontroller um weitere Eingabe- sowie Formatierungsmöglichkeiten zu erweitern. Außerdem schlug sie ein Toolkit-internes Visualisierungstool vor, welches komplett innerhalb der Unity-Umgebung funktionieren würde.

6.2.4 Auswertung der Testdaten

Da das Eventsystem nicht verwendet wurde, enthielten die vom Toolkit ausgegebenen JSON-Daten nur die Werte, welche im Testcontroller eingegeben wurden (Probandendaten, Temperaturschätzungen usw.), sowie die automatisch aufgezeichneten Temperaturdaten beim Start einer Stage. Bei der Auswertung der Daten entschied sich Bianca Schneider dagegen, R zu verwenden und setzte stattdessen auf Microsoft Excel sowie die Statistiksoftware Tableau Public. In Excel führte sie mit den Daten eine Varianzanalyse (ANOVA) durch und erstellte verschiedene Diagramme und Visualisierungen. Tableau wurde dafür verwendet, die entstandenen Daten zu betrachten und zu vergleichen.

Allerdings erforderten die Daten, insbesondere in Excel, wo sie vor dem Import noch zu CSV konvertiert wurden, einiges an Umstrukturierungsarbeit. Ein Konflikt entstand darin, dass Schneider bei der Eingabe der Daten in den Testcontroller für Dezimalzahlen ein Komma statt einem Punkt verwendet hatte, weshalb diese Werte nicht korrekt als Dezimalzahlen erkannt wurden.

Gefühlte Temperaturen der Würfel				
Würfel	Gruppe	Ø Gefühlte Temperatur	P-Wert	
1	1	5,18°C	0,7096	
	2	7,27°C		
	3	6,64°C		
2	1	10,36°C	0,6510	
	2	11,91°C		
	3	12,55°C		
3	1	20,73°C	0,5331	
	2	21,45°C		
	3	23,55°C		
4	1	9,64°C	0,2577	
	2	14,55°C		
	3	13,27°C		
5	1	16,09°C	0,0957	
	2	18,18°C		
	3	19,73°C		
6	1	21,27°C	0,6953	
	2	22,82°C		
	3	22,91°C		

Abbildung 32 Eine in Word erstellte Tabelle von Werten, die über eine Varianzanalyse in Excel berechnet wurden. Sie zeigt die Unterschiede der gefühlten Temperaturen der Würfel zwischen den Gruppen und Würfeln. (Bianca Schneider)

Parallel hat auch Daniel Hepperle, auf dessen Arbeit Bianca Schneiders Thesis aufbaut, für ein wissenschaftliches Paper mit den Daten gearbeitet. Er nutzte R, um die Daten mittels „MakeTable“ zunächst in einen Dataframe und anschließend in eine CSV-Datei zu konvertieren. Für die weitere Verarbeitung nutzte er Excel und die R-basierte Statistiksoftware Jamovi. Hepperle konnte die Daten für verschiedene Visualisierungen und Auswertungen nutzen.

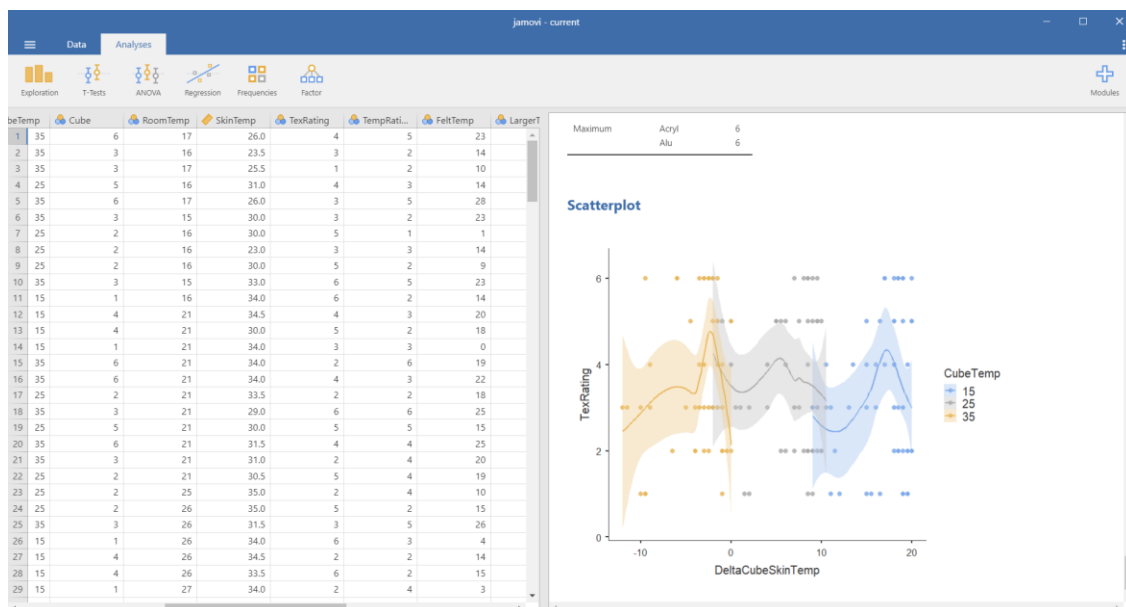


Abbildung 33 Eine von Daniel Hepperle in Jamovi erstellte Visualisierung. Links sind die importierten Daten in Tabellenform zu sehen. (Daniel Hepperle)

6.3 Einsatz des Toolkits in einem Pseudoexperiment

Da im praktischen Einsatz des Toolkits im Experiment von Bianca Schneider nur ein Bruchteil der Funktionen umgesetzt werden konnten, wurde beschlossen, das Toolkit ein zweites Mal zu evaluieren. Hierzu wurde ein Pseudoexperiment konzipiert, welches ein wissenschaftliches Experiment simulieren soll, das die Funktionen des wissenschaftlichen Toolkits möglichst komplett ausnutzt.

Das Pseudoexperiment wurde von Andreas Siess an der Hochschule Karlsruhe durchgeführt. Auf Basis der Quickstart-Anleitung, sollte er das Toolkit möglichst selbstständig einbauen, ohne dass dabei Hilfe vom Toolkitersteller benötigt wurde. Am Ende des Experiments wurden die Ergebnisse mit einem weiteren Fragebogen evaluiert.

6.3.1 Einbau des Toolkits in die Testszene

Das Toolkit wurde in eine von Andreas Siess zuvor erstellte Unity-Szene eingebaut. In dieser Szene befanden sich einige interaktive Objekte, wie ein Ball, welcher aufgehoben werden konnte.

Nach dem Import des Toolkit-Packages erschien eine Fehlermeldung im STKVRInputTracker-Skript. Dies ließ sich darauf zurückzuführen, dass in der Testszene eine ältere Version des SteamVR-Plugins verwendet wurde, welche noch nicht das für das Skript erforderliche VR-Inputsystem besaß. Das Plugin wurde deshalb auf die neueste im Unity Assetstore verfügbare Version aktualisiert. Hier trat jedoch erneut eine Fehlermeldung im selben Skript auf. Wie sich herausstellte, wurde das Input-System in der aktuellsten Version 2.2 im Vergleich zur bei der Entwicklung des Toolkits verwendeten Version 2.0 erneut verändert. Da es nicht auf Anhieb möglich war, den Fehler im Skript für die neueste Version zu beheben, musste schließlich Version 2.0 aus den Projektdateien des Toolkits verwendet werden.

Als nächstes wurde die Teststeuerung in die Szene implementiert. Dies war ohne Probleme möglich. Ebenso konnten ohne Weiteres Tracking-Events auf Objekte, wie den Ball, gelegt werden. Siess erstellte außerdem ein benutzerdefiniertes

Event, welches ausgelöst wird, wenn der Ball aufgehoben wurde. Aus den generierten JSON-Dateien konnte in R eine Heatmap erstellt werden, welche die Position des Balls während des Experimentablaufs anzeigt.

Ein Problem trat bei der Verwendung der Wiedergabe-Funktion auf. Zunächst konnte die JSON-Datei zwar geladen werden, aber die Position des Balls in der Szene veränderte sich nicht. Es stellte sich heraus, dass dies daran lag, dass die Wiedergabefunktion nur einen STKEventSender an jedem GameObject abfragte, was dazu führte, dass der Ball, welcher zwei Event Sender besaß, nicht abgespielt wurde. Dieses Problem wurde nach dem Experiment durch eine Anpassung des STKScenePlayback-Skriptes behoben, sodass auch mehrere Event Sender kein Problem mehr im Toolkit darstellen.

Die ausgegebenen JSON-Daten wurden anschließend in R geladen, und dazu verwendet, aus den aufgezeichneten Daten des Balls eine positionale Heatmap zu erstellen.

6.3.2 Befragung nach Durchführung des Pseudoexperiments

Nach der Durchführung des Pseudoexperiments hat Andreas Siess ebenfalls einen Fragebogen ausgefüllt. Dieser enthielt neben den Fragen, die schon nach Bianca Schneiders Test gestellt wurden, zusätzliche Fragen, welche sich auf die Teile des Toolkits beziehen, die bei Schneiders Experiment nicht zum Einsatz kamen. Wie schon im vorherigen Kapitel werden die Fragen mit den zugehörigen Antworten nachfolgend aufgelistet.

Frage 1: *Konntest du das Toolkit auf Basis der Anleitung und Tooltips in dein Experiment einbauen, ohne dass dabei zu irgendeinem Zeitpunkt fremde Hilfe notwendig war?*

Antwort: *Nein*

Frage 2: *Wenn Nein, bei welchen Aspekten des Toolkits hattest du Schwierigkeiten, diese ohne fremde Hilfe zu verwenden?*

Antwort: *bitte, wenn möglich, Dateipfade zu den einzelnen Komponenten angeben*

Laut Siess war es nicht möglich, das Toolkit komplett ohne fremde Hilfe anzuwenden. Spezifisch kritisiert er die fehlenden Dateipfade in der Quickstart-Anleitung. Die Anleitung wurde auf Basis dieses Feedback erweitert, sodass deutlicher ist, wo sich Bedienelemente und Skripte innerhalb des Toolkits befinden.

Frage 3: *Wie nützlich war das Testcontroller-Interface, um den Ablauf des Experimentes zu steuern? (1 = Sehr nützlich, 6 = Überhaupt nicht nützlich)*

Antwort: 1

Frage 4: *Wie nützlich war das Event-System, um Ereignisse während des Experiments aufzuzeichnen? (1 = Sehr nützlich, 6 = Überhaupt nicht nützlich)*

Antwort: 1

Sowohl der Testcontroller als auch das Eventsystem wurden als sehr nützlich empfunden.

Frage 5: *Sind zu irgendeinem Zeitpunkt während des Tests Performanceprobleme (Ruckler, Abstürze) aufgetreten, welche vermutlich auf das Toolkit zurückzuführen waren?*

Antwort: Nein

Frage 7: *Traten während des Experiments zu irgendeinem Zeitpunkt Fehler (Fehlermeldungen, Bugs) auf, die auf das Toolkit zurückzuführen waren?*

Antwort: Ja

Während des Experiments wurden, wie in Bianca Schneiders Experiment, keine Performanceprobleme beobachtet. Die aufgetretenen Fehler werden in Kapitel 6.2.2 beschrieben.

Frage 9: *Konnten externe Geräte problemlos in das Experiment eingebunden werden?*

Antwort: Ja

Frage 11: War die automatisch generierte JSON-Formatierung für die anschließende Analyse der Daten nützlich? (1 = Ja, sehr; 6 = Nein, überhaupt nicht)

Antwort: 1

Frage 13: Findest du das Toolkit insgesamt nützlich für die Durchführung von Experimenten in VR? (1 = Ja, sehr nützlich; 6 = Nein, überhaupt nicht nützlich)

Antwort: 1

Frage 14: Welche Funktionen und Aspekte des Toolkits fandest du am nützlichsten?

Antwort: Tracking-Funktion der Position, Replay-Funktion des Experiments, einfaches UI zur Angabe sonstiger Parameter am Start

Frage 15: Würdest du, wenn du ein weiteres VR-Experiment aufsetzen musst, erneut das Toolkit verwenden?

Antwort: Ja

Das Feedback für die Nützlichkeit der JSON-Formatierung sowie der Nützlichkeit des Toolkits an sich fiel durchweg positiv aus. Als nützlichste Aspekte wurden die Tracking- und Wiedergabefunktion sowie der Testcontroller genannt.

Frage 16: Welche Aspekte des Toolkits sollten deiner Meinung nach verbessert werden?

Antwort: Etwas mehr Doku wäre schön, ansonsten ein sehr gutes Tool!!!

Als Punkt, der verbessert werden sollte, nannte Siess die Dokumentation des Toolkits. Während des Pseudoexperiments in Karlsruhe fiel hier auch der konkrete Vorschlag, neben den Codekommentaren und dem technischen Aufbau eine Dokumentation aller Klassen und Funktionen innerhalb des Projektes zur Verfügung zu stellen. Auf Basis dieses Feedbacks wurde die Entscheidung getroffen, noch eine HTML-basierte Codedokumentation aufzusetzen (Siehe 5.5 Codedokumentation).

6.4 Erfüllung der Kriterien

In diesem Kapitel soll betrachtet werden, inwieweit die im Kapitel 4.1 definierten Kriterien in der finalen Version des Toolkits umgesetzt werden konnten.

Erweiterbarkeit

Das Toolkit wurde so konzipiert, dass insbesondere der Kern des Telemetriemoduls möglichst vielfältig einsetzbar ist und sich auch für etwaige Erweiterungen eignet. So wurde das Eventsystem sowohl für die Tracking-Funktion als auch für die Aufzeichnung der SteamVR-Eingaben verwendet, ohne dass hierzu die Skripte des Eventsystems abgewandelt werden mussten. Um zu ermöglichen, dass auch andere Personen später problemlos das Toolkit erweitern können, wurden die wichtigsten Funktionen komplett kommentiert. Außerdem wurde eine Codedokumentation erstellt.

Bedienbarkeit

Um die Bedienbarkeit zu gewährleisten, wurde eine Quickstart-Anleitung erstellt, welche die nötigen Schritte erklärt, um die Funktionen des Toolkits zu implementieren. Außerdem enthalten die Interface-Elemente des Toolkits Tooltips, um die zugehörigen Funktionen zu erklären.

Die Bedienbarkeit des Toolkits wurde erstmals beim Einbau des Toolkits in die Testumgebung von Bianca Schneiders Versuch auf die Probe gestellt. Hier ergaben sich noch einige Schwachstellen, wie zum Beispiel notwendige Schritte des Einbaus, die in der Anleitung nicht erklärt wurden. Diese wurden anschließend korrigiert. Ebenfalls stellte die R-Statistiksprache ein Problem für die Bedienbarkeit dar, da sie eine, im Vergleich zur Bedienung des Toolkits, hohe Einarbeitungszeit erfordert.

Performance

Die Performance ist ein Kriterium, auf welches schon während der Entwicklung des Toolkits oft geachtet werden musste. Systeme wie die JSON-Speicherung und die serielle Datenübertragung mussten mehrere Iterationen durchlaufen, bis sie akzeptable Performance erreichten. Systeme wie die Datenbegrenzung durch Sliding Windows verhindern ein signifikantes Verschlechtern der Performance, wenn die Datensammlung über lange Zeiträume durchgeführt wird.

In Bianca Schneiders Versuch konnten keine mit dem Toolkit in Zusammenhang stehenden Performanceprobleme beobachtet werden. Dies war allerdings zu erwarten, da das Eventsystem, und damit der performancekritischste Teil des Toolkits, nicht zum Einsatz kam. Es wurden allerdings auch im späteren Pseudoexperiment keine Performanceprobleme festgestellt.

Anbindung externer Hardware

Die Anbindung externer Hardware konnte über das Herstellen einer seriellen Verbindung im Toolkit umgesetzt werden. In Bianca Schneiders Versuch konnte diese Funktion erfolgreich genutzt werden, um mit einem Sensor die Lufttemperatur sowie Luftfeuchtigkeit aufzuzeichnen. Limitierungen ergaben sich jedoch dadurch, dass nicht alle Geräte über diese Methode angeschlossen werden können. So wurde ein separates Thermometer im Versuch genutzt, um die Handtemperatur der Probanden aufzuzeichnen. Dieses mit dem Toolkit zu verbinden war nicht möglich, da es eine proprietäre App für Mobiltelefone nutzt, um die gemessenen Temperaturen anzuzeigen. Diese Limitierung war jedoch zu erwarten, da es eine große Anzahl verschiedener Gerätetreiber und keine universellen Standards gibt. Mit der seriellen Schnittstelle konnte jedoch sichergestellt werden, dass eine möglichst hohe Kompatibilität zu verschiedensten Geräten besteht.

7 Fazit

Über die sechsmonatige Bearbeitungszeit der Thesis ist ein wissenschaftliches Toolkit entstanden, welches es über einfache Bedienelemente ermöglicht, Experimente für VR-Umgebungen in Unity zu erstellen.

In dieser Arbeit wurden hierfür zunächst die theoretischen Grundlagen erklärt, auf denen das Konzept beruht. Hierbei wurde erst ein Überblick über das Themenfeld Virtual Reality und den aktuellen Stand der Technik gegeben. Anschließend wurden die Erhebung und Verarbeitung von Daten in der Wissenschaft betrachtet, da die Sammlung und der Umgang mit Daten zentrale Funktionen des entstandenen Toolkits sind. Im nächsten Kapitel wurden zwei Programme untersucht, ViSTA und Unity Analytics, die eine potenzielle Grundlage für das Toolkit hätten sein können. Da die untersuchten Programme als ungeeignet befunden wurden, wurde die Entscheidung getroffen, das Toolkit als eigenständiges Unity-Plugin umzusetzen. Für die spätere Evaluation wurden mehrere Kriterien definiert, welche das Toolkit erfüllen sollte: Erweiterbarkeit, Bedienbarkeit, Performance und Anbindung externer Hardware. Erweiterbarkeit bedeutet, dass das Toolkit explizit darauf ausgelegt sein sollte, auch außerhalb dieser Arbeit abgewandelt und erweitert zu werden. Das Toolkit sollte außerdem möglichst wenig Einarbeitung erfordern und größtenteils über das Unity Interface bedienbar sein, was im Kriterium der Bedienbarkeit definiert ist. Des Weiteren sollten die Funktionen des Toolkits für die Anwendung in VR möglichst wenig Rechenaufwand erfordern, damit eine gute Performance erreicht werden kann. Als abschließendes Kriterium sollte es möglich sein, externe Geräte, wie Sensoren, mit dem Toolkit zu verbinden. Hierbei sollte eine möglichst gute Kompatibilität zu einer hohen Anzahl von unterschiedlichen Geräten erreicht werden.

Auf Basis der im Grundlagenteil gewonnenen Kenntnisse, sowie der definierten Zielsetzung, wurde ein modulbasiertes Konzept eines wissenschaftlichen Toolkits für VR erstellt. Den Kern des entstandenen Toolkits bildet das Telemetriemodul, welches über ein Eventsystem verschiedene Forschungsdaten

innerhalb und außerhalb der virtuellen Welt aufzeichnen und speichern kann. Im Vergleich zu Software mit ähnlicher Funktionalität, wie Unity Analytics, bietet das Eventsystem des Toolkits eine höhere Flexibilität in der Generierung und Aufzeichnung der Daten sowie einen offenen, erweiterbaren Quellcode. Die im Toolkit generierten Daten werden als JSON-Dateien gespeichert und können vielfältig ausgewertet werden, zum Beispiel mit den R-Funktionen des Visualisierungsmoduls. Sie können ebenfalls verwendet werden, um den Ablauf eines aufgezeichneten Experiments wiederherzustellen und erneut aus beliebigen Blickwinkeln zu betrachten. Außerdem ermöglicht es die Generierung einer Benutzeroberfläche, welche die Unterteilung und Steuerung des Experiments vereinfacht. Das Toolkit ist darauf abgestimmt, zusammen mit dem SteamVR-Plugin zu funktionieren und bietet darauf zugeschnittene Funktionen, wie die automatische Aufzeichnung von SteamVR-Input Aktionen und vom Probanden betrachtete Szenenobjekte.

Das Toolkit wurde in zwei Experimenten, einer Masterthesis zur Temperaturwahrnehmung in VR und einem Pseudoexperiment, praktisch eingesetzt und evaluiert. Durch den Einsatz und die anschließende Befragung der Tester zeigte sich, dass das entstandene Toolkit als ein nützliches Werkzeug in der Erstellung und Durchführung von Experimenten empfunden wurde. Ebenso wurden in den Experimenten auch Schwachstellen identifiziert, welche teilweise noch im Rahmen der Thesis behoben werden konnten. Abschließend wurden auf Basis der Evaluation und des umgesetzten Toolkits erneut die zuvor definierten Kriterien betrachtet. Hierbei wurde festgestellt, dass insgesamt alle Kriterien erfüllt werden konnten. Die Erweiterbarkeit wurde durch eine umfassende Dokumentation des Programmcodes und der Funktionsweise, sowie einen komplett offenen Quellcode gewährleistet. Während den Tests konnten keine Performanceprobleme ausgemacht werden. Das Anbinden externer Hardware wurde durch eine serielle Verbindung ermöglicht. Die Bedienbarkeit wurde ebenfalls positiv bewertet, wobei sich bei diesem Kriterium auch Schwächen zeigten, da die Verarbeitung der automatisch generierten Daten nicht immer intuitiv möglich war und beispielsweise Konvertierungen erforderte.

In einer weiterführenden Arbeit am Toolkit könnte ein besonderes Augenmerk auf den Aspekt der Verarbeitung der entstandenen Daten gelegt werden, sodass mit diesen ebenso intuitiv umgegangen werden kann, wie mit der Benutzeroberfläche des Toolkits selbst.

Das hier entstandene Toolkit soll eine Basissoftware für Experimente in VR darstellen. Es ist explizit darauf ausgelegt, in Zukunft erweitert und ausgebaut zu werden. Im Programmcode wurde darauf geachtet, alle Funktionen zu dokumentieren und die Module so zu erstellen, dass sie möglichst unabhängig voneinander funktionieren. In den Befragungen nach den Experimenten wurden außerdem Features, wie ein internes Visualisierungssystem, vorgeschlagen, welche als Basis für eine Erweiterung des Toolkits dienen können.

Virtual Reality ist eine Technologie, die sich auch in den kommenden Jahren stark weiterentwickeln wird, vielleicht sogar in Richtungen, die aktuell noch gar nicht vorherzusehen sind. Um diese neuen Technologien und deren Auswirkungen auf Menschen leichter untersuchen zu können, wurde das Scientific Virtual Reality Toolkit erstellt.

Literaturverzeichnis

Buchquellen

Chen, M., Mao, S., & Zhang, Y. (2014). *Big Data: Related Technologies, Challenges and Future Prospects*. Cham, Heidelberg, New York, Dordrecht, London: Springer.

Dörner, R., Broll, W., Grimm, P., Jung, B., & Göbel, M. (2013). Einleitung. In *Virtual und Augmented Reality (VR/AR) / Grundlagen und Methoden der Virtuellen und Augmentierten Realität*. (S. 1-32). Berlin, Heidelberg: Springer Verlag.

Drachen, A., Seif El-Nasr, M., & Canossa, A. (2013). Game Analytics - The Basics. In A. Drachen, M. Seif El-Nasr, & A. Canossa, *Game Analytics / Maximizing the Value of Player Data* (S. 13-40). London, Dordrecht, Heidelberg, New York: Springer Verlag.

Fayyad, U., Gregory, P.-S., & Smyth, P. (1996). From data mining to knowledge discovery in databases. *AI Magazine*, 17.3, S. 37-51.

Grimm, P., Herold, R., Hummel, J., & Broll, W. (2013). VR-Eingabegeräte. In R. Dörner, W. Broll, P. Grimm, & B. Jung, *Virtual und Augmented Reality (VR/AR): Grundlagen und Methoden der Virtuellen und Augmentierten Realität* (S. 97-126). Berlin, Heidelberg: Springer-Verlag.

Grimm, P., Herold, R., Reiners, D., & Cruz-Neira, C. (2013). VR-Ausgabegeräte. In R. Dörner, W. Broll, P. Grimm, & B. Jung, *Virtual und Augmented Reality (VR/AR) / Grundlagen und Methoden der Virtuellen und Augmentierten Realität* (S. 127-156). Berlin, Heidelberg: Springer-Verlag.

Han, J., Kamber, M., & Pei, J. (2012). *Data Mining / Concepts and Techniques*. Waltham: Morgan Kaufmann Publishers.

Hedtstück, U. (2017). *Complex Event Processing: Verarbeitung von Ereignismustern in Datenströmen*. Berlin: Springer-Verlag.

- Herczeg, M. (2006). *Interaktionsdesign: Gestaltung interaktiver und multimedialer Systeme*. München: Oldenbourg Wissenschafts Verlag.
- Herczeg, M. (2018). *Software-Ergonomie (4. Auflage)*. Berlin, Boston: Walter de Gruyter GmbH.
- Inmon, W. (2005). *Building the Data Warehouse - Fourth Edition*. Indianapolis: Wiley.
- Jaeschke, G., Leissler, M., & Hemmje, M. (2005). Modeling Interactive, 3-Dimensional Information Visualizations Supporting Information Seeking Behaviors. In S.-O. Tergan, & T. Keller, *Knowledge and Information Visualization* (S. 119-134). Berlin, Heidelberg: Springer-Verlag.
- Jerald, J. (2016). *The VR Book*. San Rafael: ACM.
- Kaya, M. (2009). Verfahren der Datenerhebung. In A. Sönke, D. Klapper, U. Konradt, A. Walter, & J. Wolf, *Methodik der empirischen Forschung (3. Auflage)* (S. 49-65). Wiesbaden: Springer-Verlag.
- Keller, T., & Tergan, S.-O. (2005). Visualizing Knowledge and Information:. In T. Keller, & S.-O. Tergan, *Knowledge and Information Visualization: Searching for Synergies* (S. 1-27). Berlin, Heidelberg: Springer-Verlag.
- Mazza, R. (2009). *Introduction to Information Visualization*. London: Springer-Verlag.
- Medler, B. (2013). Visual Game Analytics. In A. Drachen, M. Seif El-Nasr, & A. Canossa, *Game Analytics / Maximising the Value of Player Data* (S. 403-435). New York: Springer Verlag.
- Rack, O., & Christophersen, T. (2009). Experimente. In S. Albers, D. Klapper, U. Konradt, W. Achim, & J. Wolf, *Methodik der empirischen Forschung (3. Auflage)* (S. 17-32). Wiesbaden: Springer Verlag.
- Skiena, S. S. (2017). *The Data Science Design Manual*. Cham: Springer Nature.

- Slater, M., & Wilbur, S. (1997). A framework for immersive virtual environments (FIVE): Speculations on the role of presence in virtual environments. *Presence: Teleoperators & Virtual Environments*, 6(6), S. 603-616.
- Sutherland, I. E. (1965). The Ultimate Display. *Multimedia: From Wagner to virtual reality* (S. 506-508). New York: Norton.
- Wiegand, H. E. (1998). *Wörterbuchforschung: Untersuchungen zur Wörtbuchbenutzung, zur Theorie, Geschichte, Kritik und Automatisierung der Lexikographie*. Berlin, New York: De Gruyter.

Internetquellen

- Bell, D. (01.02.2019). *The Class Diagram*. Abgerufen von IBM am 01.02.2019:
<https://www.ibm.com/developerworks/rational/library/content/RationalEdge/sepo4/bell/>
- Bitkom e.V. (08.02.2019). *Zukunft der Consumer Technology – 2018*. Abgerufen von Bitkom am 08.02.2019:
<https://www.bitkom.org/sites/default/files/file/import/180822-CT-Studie-2018-online.pdf>
- Clifton, C. (08.10.2018). *Data Mining : Encyclopædia Britannica*. Abgerufen von Encyclopædia Britannica am 08.10.2018:
<https://www.britannica.com/technology/data-mining>
- Crockford, D. (15.11.2018). *Introducing JSON*. Abgerufen von JSON am 15.11.2018:
<https://json.org/>
- EWSsoftware. (15.02.2019). *SHFB*. Abgerufen von Github am 15.02.2019:
<https://github.com/EWSsoftware/SHFB>
- Göbel, M. (21.02.2019). *SimpleJSON*. Abgerufen von Github am 21.02.2019:
<https://github.com/Bunny83/SimpleJSON>
- HTC. (22.11.2018). *Vive Virtual Reality System*. Abgerufen von Vive am 22.11.2018:
<https://www.vive.com/us/product/vive-virtual-reality-system/>

HTC. (13.02.2019). *The New Vive Pro*. Abgerufen von HTC Vive am 13.02.2019:
<https://www.vive.com/us/pro-eye/>

Peterson, J. (11.07.2018). *Scripting Runtime Improvements in Unity 2018.2*.
Abgerufen von Unity Blog am 19.10.2018:
<https://blogs.unity3d.com/2018/07/11/scripting-runtime-improvements-in-unity-2018-2/>

RWTH Aachen. (13.10.2018). *ViSTA Virtual Reality Toolkit*. Abgerufen von
Webseite der RWTH Aachen am 13.10.2018: <http://www.itc.rwth-aachen.de/cms/IT-Center/Forschung-Projekte/Virtuelle-Realitaet/Infrastruktur/~fgmo/ViSTA-Virtual-Reality-Toolkit/>

The R Foundation. (02.01.2019). *What is R?* Abgerufen von The R Project for
Statistical Computing am 02.01.2019: <https://www.r-project.org/about.html>

Unity Technologies. (16.10.2018). *Manual: Unity Analytics*. Abgerufen von Unity
am 16.10.2018: <https://docs.unity3d.com/Manual/UnityAnalytics.html>

Unity Technologies. (16.10.2018). *Unity Analytics*. Abgerufen von Unity am
16.10.2018: <https://unity.com/de/solutions/analytics>

Valve Corporation. (03.02.2019). *Steam-Hard- & Softwareumfrage: January 2019*.
Abgerufen von Steam am 03.02.2019:
<https://store.steampowered.com/hwsurvey>

Valve Software. (13.11.2018). *SteamVR Input*. Abgerufen von Github am 13.11.2018:
<https://github.com/ValveSoftware/openvr/wiki/SteamVR-Input>

Sonstige Quellen

Schneider, B. (2019). *Temperaturwahrnehmung in Virtual Reality und ihre Auswirkungen auf die Glaubwürdigkeit von Materialien*, Furtwangen: Hochschule Furtwangen

Abbildungsverzeichnis

Abbildung 1 Ein CAVE Ausgabesystem.....	5
Abbildung 2 Die HTC Vive	6
Abbildung 3 Ablauf eines empirischen Forschungsprozesses	8
Abbildung 4 Erstellungsprozess einer Visualisierung.....	11
Abbildung 5. Beispiel einer automatisch generierten Visualisierung in Unity Analytics.....	13
Abbildung 6 Mockup des Testcontrollers	18
Abbildung 7 Beispiel einer in R generierten Heatmap	24
Abbildung 8 Beispiel eines generierten Balkendiagramms zur Objektbetrachtung	24
Abbildung 9 Mockup der Track Object Oberfläche	26
Abbildung 10 Aufbau der Wiedergabesteuerung	27
Abbildung 11 Die Beispielszene	28
Abbildung 12 Klassendiagramm von STKSettings.....	30
Abbildung 13 Klassendiagramm von STKArrayTools.....	31
Abbildung 14 Klassendiagramm Testaufbaumodul	32
Abbildung 15 Klassendiagramm des Telemetriemoduls	34
Abbildung 16 Klassendiagramm des VR-Integrationsmoduls	37
Abbildung 17 Klassendiagramm des Wiedergabemoduls.....	38
Abbildung 18 Klassendiagramm STKEventEditor	40
Abbildung 19 Das von Eventparameter erstellte Event Interface.....	40
Abbildung 20 Klassendiagramm STKTrackEditor	41
Abbildung 21 Das von STKTrackEditor erstellte Tracking Interface.....	41
Abbildung 22 Klassendiagramm STKTestController	42
Abbildung 23 Klassendiagramm STKTestStageEditor	42
Abbildung 24 Das von STKTestStageEditor erstellte Stage Interface	43
Abbildung 25 Klassendiagramm STKPlaybackEditor	43
Abbildung 26 Das von STKScenePlayback erstellte Wiedergabe Interface.....	43
Abbildung 27 Arduino Board mit angeschlossenem Temperatursensor	50
Abbildung 28 Dokumentation der STKEvent Klasse	51

Abbildung 29 Ein vom Toolkit generiertes Interface, welches für das Experiment verwendet wurde.....	54
Abbildung 30 Durchführung eines Probandentests.....	55
Abbildung 31 Die falsche generierte Formatierung (links) und die korrigierte Formatierung (rechts) der JSON-Datei	56
Abbildung 32 Eine Tabelle von Werten, die über eine Varianzanalyse in Excel berechnet wurden. Sie zeigt die Unterschiede der gefühlten Temperaturen der Würfel zwischen den Gruppen und Würfeln.	62
Abbildung 33 Eine von Daniel Hepperle in Jamovi erstellte Visualisierung. Links sind die importierten Daten in Tabellenform zu sehen.	62

Eidesstattliche Erklärung

Ich erkläre hiermit an Eides statt, dass ich die vorliegende Master-Thesis selbständig und ohne unzulässige fremde Hilfe angefertigt habe. Alle verwendeten Quellen und Hilfsmittel, sind angegeben.

Friesenheim, den

Felix Gähr

Eingesetzte Software

Für die Erstellung der Thesis und des Toolkits wurde folgende Software eingesetzt:

- Unity 2018.2.9f1
- Microsoft Visual Studio Community 2017
- SteamVR Unity Plugin - v2.0
- RStudio 1.1.456
- Microsoft Word
- Microsoft Visio
- Github
- Smartgit
- SHFB v2018.12.10.0

Inhalt des Datenträgers

Auf dem beiliegenden Datenträger befinden sich folgende Inhalte:

- Scientific Virtual Reality Toolkit (Unity Projekt) + Dokumentation
- Scientific Virtual Reality Toolkit (Unity Package)
- Quickstart Anleitung
- Digitale Kopie des Thesisdokuments

Fragenkatalog

Nachfolgend werden die Fragen aufgelistet, welche zur Evaluation des Toolkits verwendet wurden, sowie deren Antwortmöglichkeiten und Zuordnung zu den Bewertungskriterien. Blau hinterlegte Fragen wurden nur nach dem Pseudoexperiment gestellt, da die dort genannten Features in Bianca Schneiders Versuch nicht zum Einsatz kamen bzw. im Falle der externen Datenübertragung nicht von ihr eingebaut wurden.

Frage	Antwortmöglichkeiten	Kategorie
Konntest du das Toolkit auf Basis der Anleitung und Tooltips in dein Experiment einbauen, ohne dass dabei zu irgendeinem Zeitpunkt fremde Hilfe notwendig war?	Ja, Nein	Bedienbarkeit
Wenn Nein, bei welchen Aspekten des Toolkits hattest du Schwierigkeiten, diese ohne fremde Hilfe zu verwenden?	Freitext	Bedienbarkeit
Wie nützlich war das Testcontroller-Interface, um den Ablauf des Experimentes zu steuern?	1 = Sehr nützlich, 6 = Überhaupt nicht nützlich	Bedienbarkeit, Feedback
Wie nützlich war das Event-System, um Ereignisse während des Experiments aufzuzeichnen?	1 = Sehr nützlich, 6 = Überhaupt nicht nützlich	Bedienbarkeit, Feedback
Sind zu irgendeinem Zeitpunkt während des Tests Performanceprobleme (Ruckler, Abstürze) aufgetreten, welche vermutlich auf das Toolkit zurückzuführen waren?	Ja, Nein	Performance
Wenn ja, welche Performanceprobleme sind aufgetreten?	Freitext	Performance
Traten während des Experiments zu irgendeinem Zeitpunkt Fehler (Fehlermeldungen, Bugs) auf, die auf das Toolkit zurückzuführen waren?	Ja, Nein	Performance, Feedback
Wenn Ja, welche Fehler sind aufgetreten?	Freitext	Performance, Feedback

Konnten externe Geräte problemlos in das Experiment eingebunden werden?	Ja, Nein, Ich habe keine externen Geräte angeschlossen	Anbindung externer Hardware, Bedienbarkeit
Wenn Nein, welche Probleme sind beim Anschluss externer Geräte aufgetreten?	Freitext	Anbindung externer Hardware, Bedienbarkeit
War die automatisch generierte JSON-Formatierung für die anschließende Analyse der Daten nützlich?	1 = Ja, sehr, 6 = Nein, überhaupt nicht	Bedienbarkeit, Feedback
Auf welche Schwierigkeiten bist du bei der Verarbeitung der JSON-Daten gestoßen?	Freitext	Bedienbarkeit, Feedback
Findest du das Toolkit insgesamt nützlich für die Durchführung von Experimenten in VR?	1 = Ja, sehr nützlich, 6 = Nein, überhaupt nicht nützlich	Feedback
Würdest du, wenn du ein weiteres VR-Experiment aufsetzen müsstest, erneut das Toolkit verwenden?	Ja, Nein	Feedback
Welche Aspekte des Toolkits sollten deiner Meinung nach verbessert werden?	Freitext	Bedienbarkeit, Performance, Feedback
Welche Funktionen, die bisher nicht im Toolkit existieren, fändest du noch nützlich?	Freitext	Erweiterbarkeit, Feedback

Quickstart-Anleitung für das VR Scientific Toolkit

Das VR Scientific Toolkit vereinfacht die Durchführung von Experimenten in VR-Umgebungen in Unity. Es ermöglicht unter anderem die Aufzeichnung von verschiedenen Forschungsdaten, sowie deren Speicherung in einer JSON-Datei. In dieser Quickstart-Anleitung wird erklärt, wie Sie das Toolkit in Ihr Projekt einbinden und die wichtigsten Funktionen nutzen können.

Installation

Stellen Sie vor der Installation des Toolkits sicher, dass das SteamVR-Plugin installiert ist und setzen Sie in den Player Settings die Variable „**Scripting Runtime Version**“ auf „**.NET 4.x Equivalent**“.

Anschließend importieren Sie das Unity Package, indem Sie im Asset-Browser „Import Package“ wählen und hier das Toolkit auswählen. Sie finden in Ihrem Projekt nun den „VrScientificToolkit“ Ordner, in welchem sich Skripte und Prefabs befinden, die Sie für verschiedene Funktionen nutzen können. Für eine Demonstration der Fähigkeiten des Toolkits können Sie die „**ExampleScene**“ betrachten, in welcher diese bereits beispielhaft implementiert sind.

Festlegen der Einstellungen

Nachdem Sie das Plugin installiert haben, sollten Sie zunächst die Einstellungen des Toolkits an Ihre Bedürfnisse anpassen. Die Einstellungen sind im **Resources-Ordner** unter dem Namen „**STKSettings**“ zu finden. Hier können Sie verschiedene Einstellungen, wie den Speicherort für die generierten JSON-Dateien, auswählen.

Viele Interface Elemente sind mit Tooltips ausgestattet, die weitere Informationen zu deren Funktion bieten. Hierzu können Sie einfach die Maus über das entsprechende Element halten.

Testcontroller

Der Testcontroller dient der Steuerung des Experiments. Über ihn lassen sich während des Experiments Daten eingeben und Funktionen ausführen. Er ist für den Probanden in VR nicht sichtbar, sondern wird nur dem Testleiter auf dem Bildschirm angezeigt.

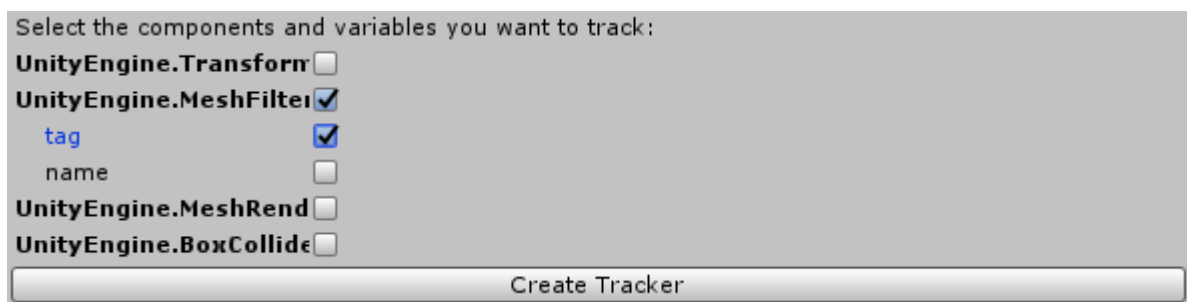


Beispiel eines Testcontrollers

Um den Testcontroller zu erstellen, ziehen Sie ihn aus dem **Prefabs-Menü** in die Szene. Klicken Sie anschließend im Inspektor auf **„Add Stage“**, um einen ersten Experimentabschnitt hinzuzufügen. Über Abschnitte bzw. Stages definiert sich der Aufbau des Experiments, auch in der automatisch generierten JSON-Datei. Für diesen Abschnitt können Sie, erneut über den Inspektor, Properties hinzufügen. Dabei handelt es sich um Eingabefelder, die zum Zeitpunkt des Experiments dazu genutzt werden können, Daten, wie zum Beispiel den Namen eines Probanden, einzugeben. Außerdem können im Inspektor Buttons erstellt werden, welche beliebige Funktionen ausführen können.

Tracking

Über die Tracking Funktion können beliebige Variablen von Objekten aufgezeichnet und in einer JSON-Datei gespeichert werden. Bevor Sie die Tracking-Funktion verwenden, ziehen Sie bitte das **„STKTrackedObjects“** Prefab in die Szene. Um Variablen eines Objektes aufzuzeichnen, öffnen Sie zunächst das Tracking-Interface über **Window/VR Scientific Toolkit/Track Object**. Das Tracking Interface zeigt immer die Komponenten des aktuell in der Szene angewählten Objektes an. Wird eine Komponente ausgewählt, können zu dieser Komponente gehörende, öffentliche Variablen ausgewählt werden.



Das Tracking Interface

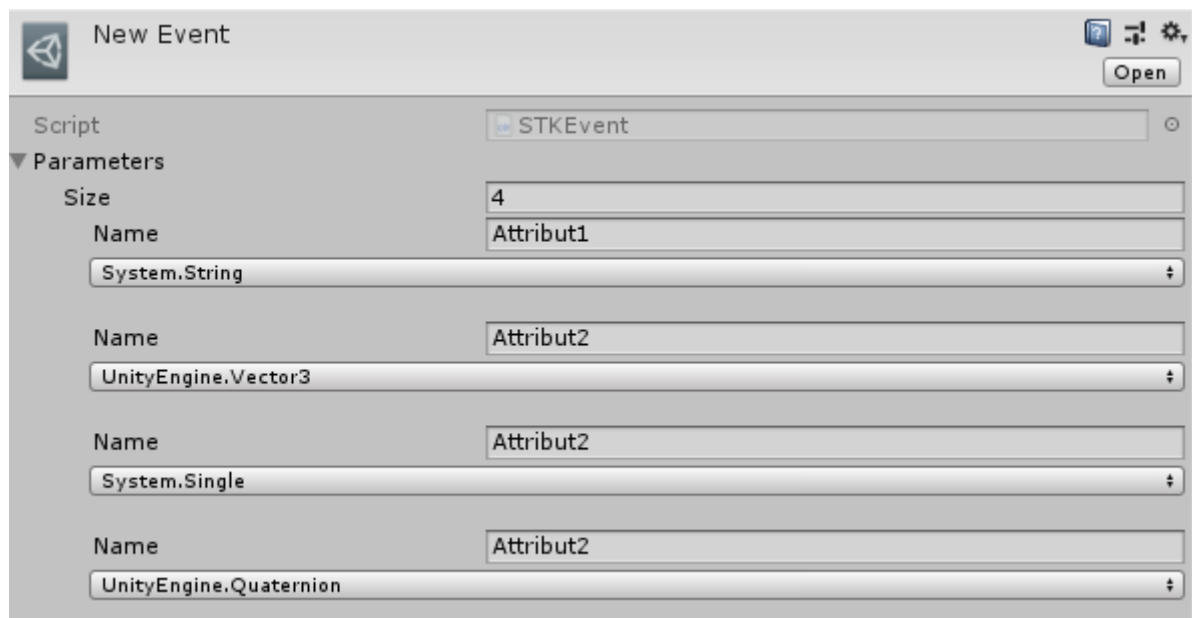
Nach einem Klick auf **„Create Tracker“** erhält das Objekt ein Skript, welches die gewählten Variablen aufzeichnet. Im Inspektor kann ein Intervall eingestellt werden, in dem die Aufzeichnung erfolgt. Damit das Event im definierten Intervall abgesendet wird, setzen Sie außerdem **„Timed Interval“** auf True. Das Event kann ebenfalls

manuell über einen Aufruf der **Deploy-Funktion** des Eventsender Skriptes abgesendet werden.

Definition eigener Events

Neben automatisch generierten Tracker-Events können Sie auch eigene Events erstellen, welche zum Beispiel ausgelöst werden können, wenn ein bestimmter Gegenstand in der Szene aufgehoben wird. Diese Events können beliebige Attribute enthalten.

Ein eigenes Event können Sie über das Kontextmenü im „Project“ Fenster von Unity über die Punkte **Create/VR Scientific Toolkit/STKEvent** definieren. Wählen sie dieses Event aus, um Parameter zu definieren. Bei jedem Parameter können Sie den Namen sowie einen Datentyp aus einer Liste von kompatiblen Datentypen auswählen.



Definition eines Events

Um dieses Event zu versenden, wird eine „**STKEventSender**“ Komponente auf einem GameObject benötigt. Legen Sie hier das erstellte Event als „**eventBase**“ fest. Um einem Eventattribut während des Testablaufs einen Wert zuzuteilen, können Sie „**SetEventValue**“ aufrufen. Die Funktion wird anschließend über „**Deploy**“ dem Eventreceiver zur Speicherung übergeben.

Wiedergabe eines Experimentes

Es ist möglich, den Ablauf eines Experimentes wiederherzustellen, um ihn erneut in der Unity Szene betrachten zu können. Hierfür werden die Daten von Tracker-Events genutzt, um sämtliche getrackten Variablen wieder auf den Wert eines bestimmten Zeitpunktes zu setzen. Um das Wiedergabe-Interface zu öffnen, wählen Sie den Menüpunkt **Window/VR Scientific Toolkit/JSON Playback**. Anschließend deaktivieren Sie die VR-Unterstützung in den Player-Einstellungen von Unity, damit keine Konflikte

mit den SteamVR-Treibern entstehen. Wenn das Playback-Fenster geöffnet ist und im Vordergrund liegt, startet der Wiedergabemodus, sobald Sie die Unity Szene starten. Hier müssen Sie zunächst eine JSON-Datei auswählen, aus der Sie einen Ablauf wiederherstellen möchten. Haben Sie eine Datei ausgewählt, können Sie über die Steuerungselemente die Stage und den Zeitpunkt auswählen, den sie wiederherstellen möchten. Es ist ebenfalls möglich, den Ablauf einer Stage in Echtzeit abzuspielen.

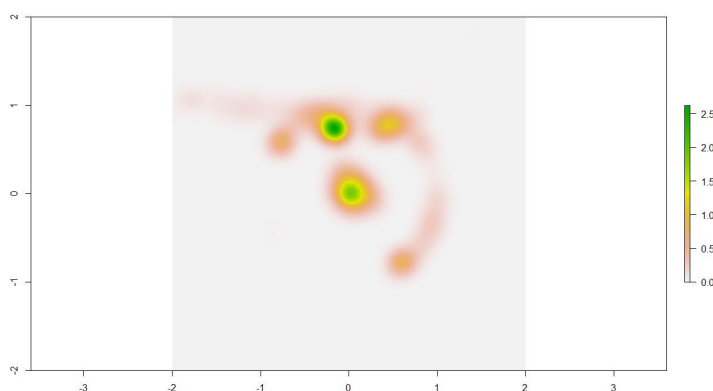
Einbau externer Geräte

Das Toolkit bietet die Klasse „**STKReceiveSerial**“, welche es ermöglicht, Daten über eine serielle Schnittstelle von einem externen Gerät auszulesen. Hierfür legen Sie das Skript auf ein beliebiges Skript und geben den gewünschten Port und die BAUD-Rate an. In „CurrentValue“ befindet sich dann die zuletzt ausgelesene Line, welche z. B. über ein Tracker-Event aufgezeichnet werden kann.

Datenverarbeitung und Visualisierungen

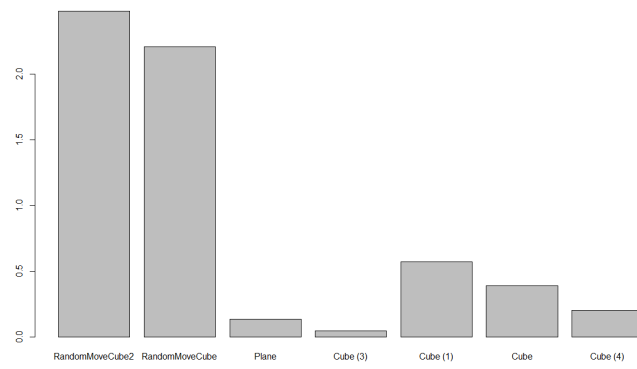
Das Toolkit erstellt automatisch JSON-Dateien, welche alle aufgezeichneten Informationen enthalten. Diese können in verschiedener Statistiksoftware verwendet werden, um die Daten zu analysieren. Im Toolkit enthalten sind einige Funktionen, die die Weiterverarbeitung der Daten mit der Statistiksprache R vereinfachen sollen. Diese befinden sich im „**Visualization**“ Ordner. Mithilfe der Funktion „**SaveFilesToWorkspace**“ lässt sich ein ganzer Ordner von JSON-Dateien in R importieren. Die Dateien werden zu einer Listenvariable konvertiert. Beispielfhaft befinden sich außerdem zwei Visualisierungsskripte im Ordner.

„**CreatePositionalHeatmap**“ erstellt eine Top-Down Heatmap von einem getrackten Objekt, welche anzeigt, an welchen Positionen sich ein Objekt wie lange in der Szene befunden hat.



Beispiel einer generierten Heatmap

„**CreateObjectLookGraph**“ verwendet Daten des „ObjectLook“ Standardevents, um ein Balkendiagramm zu erstellen, welches zeigt, wie lange welche Objekte vom Probanden betrachtet wurden.



Beispiel eines generierten Balkendiagramms