UNIVERSITY OF APPLIED SCIENCES
HOCHSCHULE MITTWEIDA

Angewandte Computer- und Biowissenschaften

Professur Medieninformatik

Bachelorarbeit

Entwicklung und Implementierung eines virtuellen Labors zur Durchführung biochemischer Experimente am Beispiel einer Verdünnungsreihe mittels Pipettieren

Kerstin Knura

Mittweida, den 11. Dezember 2017

Erstprüfer: Prof. Dr. rer. nat. Marc Ritter
Zweitprüfer: Prof. Dr. rer. nat. Petra Radehaus
Co-Betreuer: Manuel Heinzig, M. Sc.

Knura, Kerstin
Entwicklung und Implementierung eines virtuellen Labors zur Durchführung biochemischer Experimente am Beispiel einer Verdünnungsreihe mittels Pipettieren
Bachelorarbeit, Angewandte Computer- und Biowissenschaften
Hochschule Mittweida– University of Applied Sciences, Dezember 2017

Inhaltsverzeichnis

Abbildungsverzeichnis V

1 Einführung und Motivation 1

1.1 Virtuelle Realität . . . 1
1.2 BIOTACMI . . . 2
1.3 Aufgabenstellung . . . 3
1.4 Aufbau der Arbeit . . . 4

2 Grundlagen 5

2.1 Grundlagen des Pipettierens . . . 5
2.1.1 Ablauf . . . 6
2.1.2 Fehlerquellen . . . 7
2.2 Virtuelle Realität . . . 7
2.2.1 Allgemeine Funktionsweise . . . 9
2.2.2 Vergleichbare Applikationen . . . 10
2.2.3 HTC Vive . . . 11
2.3 Design . . . 12
2.3.1 3D-Objekte . . . 13
2.3.2 2D- Elemente . . . 15
2.4 Unity . . . 16
2.4.1 Grundlagen . . . 17
2.4.2 Steam VR . . . 19
2.5 Schnittstellen . . . 19
2.6 XML . . . 20

2.7 Lernmethoden 20

3 Konzeptionierung des virtuellen Labors 23

3.1 Fehlerquellen 23
3.2 Objekterstellung 24
3.3 Szenenbau 26
3.3.1 Menü 27
3.3.2 Fragenkatalog 29
3.3.3 Virtuelles Labor 30
3.3.4 Auswertung 32
3.4 Einbindung der VR 32
3.5 Zusammenfassung 33

4 Implementierung 35

4.1 Vorarbeit 36
4.2 Objekterstellung 37
4.3 Objekteinbindung 39
4.4 Steuerung 40
4.4.1 Virtual Reality-System 40
4.4.2 Interaktionen 41
4.5 Auswertung 45

5 Evaluation 49

5.1 Aufbau des Experiments 49
5.1.1 Versuchsablauf 50
5.1.2 Versuchsauswertung 50
5.2 Identifizierte Probleme 51
5.2.1 Objekte 51
5.2.2 Controller 51
5.2.3 Format 52
5.2.4 Performanz 52
5.2.5 Allgemeine Probleme 53
5.3 Fazit 53

6 Zusammenfassung und Ausblick **55**

6.1 Anpassungs- und Verbesserungsmöglichkeiten 55

6.2 Fazit 56

Literaturverzeichnis **VII**

Abbildungsverzeichnis

2.1 Stadien des Vorwärts-Pipettierens 7
2.2 Fertiges 3D-Modell eines Erlenmeyerkolbens 13
2.3 Beispiel einer UV-Map 14
2.4 Beispiel-Textur eines 3D-Modells 15
2.5 Unity-Projekt-Ansicht 17

3.1 3D-Modell-Entstehungs-Zyklus 26
3.2 Ansicht des Hauptmenüs in der Unity-Engine 28
3.3 Ansicht der Szene Fragenkatalog 30
3.4 Ansicht des virtuellen Labors 32
3.5 Auswertungs-Panel 33

4.1 Use-Case-Diagramm 36
4.2 Klassendiagramm Menü 42
4.3 Klassendiagramm Fragenkatalog 44
4.4 Klassendiagramm Game 46
4.5 Klassendiagramm Auswertung 47

1 Einführung und Motivation

Die Computertechnik und damit einhergehend auch die virtuelle Realität haben sich in den letzten Jahren rasant entwickelt. Systeme, die es dem Nutzer ermöglichen, künstliche Welten zu erleben und neue interaktive Umgebungen zu arrangieren, wurden fortlaufend verbessert und sind mittlerweile, preislich und technisch gesehen, auch für die Allgemeinheit zugänglich. Ferner wird die virtuelle Realität hinsichtlich wirtschaftlicher und wissenschaftlicher Aspekte in vielen differenzierten Gebieten genutzt. Unter anderem in Sektoren wie dem Flugzeug- und Automobilbereich, in der Entertainment-Branche und in den Naturwissenschaften, letztere Anwendung beispielsweise bezüglich Simulationen in virtuellen Laboratorien. Diese Anwendung bietet eine kostensparende und ohne den Einsatz von realen Ressourcen anwendbare Alternative, welche zudem, bei gefährlicheren Experimenten, ohne die Gefährdung des Anwenders durchführbar ist. Derartige reproduzierte Pendants tragen in Kombination mit theoretisch abgefragtem Wissen überdies dazu bei, Lehrinhalte zu vermitteln, zu repetieren und zu manifestieren. Die Herausforderung bei der Erstellung einer derartigen, virtuellen Umgebung liegt darin, Lehrinhalte schlüssig und realitätsnah zu übermitteln und eine gute Grundlage für die Weiterentwicklung und Ausweitung der erarbeiteten Software zu schaffen.

1.1 Virtuelle Realität

Virtuelle Realität beschreibt eine digitale, reproduzierte Umgebung. Die Generierung findet in Echtzeit statt [Hau10, S.13]. Mit jener Umgebung kann der Nutzer in vielfältiger Form interagieren sowie diese manipulieren.

Im Idealfall sind simulierte Realität und die auszuführenden Interaktionen so umgesetzt, dass Prozesse greifbar sind und sich der Nutzer wie in der konventionellen, realen Umgebung verhalten und bewegen kann. Diese Immersion entsteht durch das Potential des Systems, sich an die Einflussmöglichkeiten des Nutzers anzupassen und

diesen aktiv einzubinden. Somit ist eine naturalistische Betrachtungsmöglichkeit von Objekten charakteristisch für die virtuelle Realität. Vorteile hinsichtlich bisheriger virtueller Visualisierungen bringt die VR gegenüber der Darstellung komplexer Daten und Illustrationen. [Bri09, S.6f.]

Diese sind mit Hilfe der virtuellen Realität ansehnlicher und anschaulicher darstellbar. Des Weiteren kann der Nutzer die Inhalte durch die modellierte Realität einfacher verarbeiten. Hinzu kommt eine Kostenreduktion bezüglich wiederholbarer Simulationen, da die Verwendung hierbei keine realen Ressourcen behinhaltet.

1.2 BIOTACMI

Die Arbeitsgruppe *BIOTACMI*, welche sich aus den Professuren der Biotechnologie (*BIOT*), der Angewandten Chemie (*AC*) und der Medieninformatik (*MI*) an der *Hochschule Mittweida* zusammensetzt, hat das Bestreben, Lehrinhalte durch virtuelle Visualisierungen zu vermitteln und diese zu bekräftigen. Hierbei soll eine anschauliche und unmissverständliche Lernumgebung entstehen. Diese können Dozenten und Professoren als unterstützendes Hilfsmittel zu Vorlesungen und Seminaren einsetzen. Hinzu kommt der Mehrwert für Studierende der Naturwissenschaften, da die geschaffenen Applikationen als Lernhilfe für sie nutzbar sind.

Eines der anfänglichen Projekte bildet ein virtuelles Labor, welches sich eines VR-Systems bedient. In diesem sind in Zukunft diverse biochemische Experimente durchführbar und Studierende der *Hochschule Mittweida* haben die Möglichkeit, unbeschränkt Versuche virtuell zu repetieren. Die Versuchsauswahl ist dabei an den jeweiligen Lernmodulen der Hochschule angepasst und soll dazu führen, dass gewonnene Erkenntnisse und Erfahrungen im realen Experiment, hinsichtlich weniger gehäufter Fehler, ihren Beitrag leisten. Als Entwicklungsumgebung dient die *Unity Engine*, die umgesetzte Programmierung erfolgte mit `C#`. So wurde bislang, im Rahmen dieser Abschlussarbeit und in Kooperation mit weiteren Projektpartnern, eine Verdünnungsreihe mittels Pipettieren im beschriebenen Labor umgesetzt. Zudem beinhaltet die Software eine Abfrage des im Unterricht behandelten theoretischen Wissens. Während der Entwicklung der Applikation spielte die Handhabung und Steuerung der Software eine essenzielle Rolle sowie die detailgetreue Nachahmung eines realen Labors, mit allen für den Versuch benötigten Objekten.

In Kooperation mit Gabor Schulze entwickelte sich das Projekt dahingehend weiter, dass Hintergrundprozesse und der exakte Versuchsablauf der Verdünnungsreihe implementiert wurden. Zudem entstand ein Softwarekern, welcher die Möglichkeit bietet, modulare Bestandteile dynamisch in die *Unity*-Szene laden zu können. Diese Bestandteile beschreiben Geometriedaten, Verhaltensweisen sowie Konfigurationsdateien, welche für alternative Anwendungsfälle und Szenarien erweiterbar und veränderbar sind. Die Realisierung der Erweiterung erfolgte mittels XML-Dateien. [Sch17]

Ein weiteres Projekt der *BIOTACMI* beschreibt ein Tool, welches katalytische Vorgänge am Beispielfall der elektrochemischen CO_2-Reduktion visualisiert. Hierbei kristallisiert sich zudem heraus, in wie weit sich mit jenem Tool chemische Reaktionen effektiv animieren und simulieren lassen. Die geforderte Software bedient sich auch hier der *Unity Engine* und der Programmiersprache `C#`. Des Weiteren soll auch dieses Projekt in Zukunft mit Hilfe des Softwarekerns dynamisch verwendbar sein. [Hei17]

1.3 Aufgabenstellung

Der Fokus der Arbeit liegt auf der Entwicklung und Implementierung eines virtuellen Labors zur Durchführung mikrobiologischer Experimente. Hierbei wird als Beispiel-Experiment eine Verdünnungsreihe mittels Pipettieren verwendet. Zudem soll das Projekt derart erweiterbar sein, dass weitere Experimente, welche vor allen Dingen an den Lehrplan der *Hochschule Mittweida* angelehnt sind, eingebaut werden können. Das Projekt ist speziell an die *HTC Vive Virtual Reality Brille* und der dazu gehörigen Hardware ausgelegt und wird mittels der *Unity Engine* und der Programmiersprache `C#` realisiert.

Themenschwerpunkte der Arbeit bilden die Einarbeitung in die Thematik des Pipettierens, vor allem in Bezug auf die am häufigsten auftretenden Fehlerquellen im realen Experiment. Die Implementierung und Entwicklung des interaktiven Labors umfasst alle, für die Ausführung des Experiments benötigten, 3D-Modelle, Animationen sowie die interaktive Steuerung dieser. Eine intuitive Benutzeröberfläche und ein nachvollziehbarer Levelbau sollen eine bestmögliche Interaktion und Orientierung in der virtuellen Umgebung gewährleisten. Hinzu kommt eine klar verständliche Ausgabe und Darstellung der im Experiment begangenen Fehler. Erweitert wird

der Umfang der Arbeit durch eine Abfrage der für das Experiment essentiellen theoretischen Grundlagen und deren Auswertung.

Ein ausgereiftes Designkonzept und die dazu gehörige Implementierung münden in einer Evaluation der entstandenen Software. Hierbei beschreibe diese die analytische Betrachtung der Funktionsweisen sowie der, während der Implementierung aufgetretenen, Probleme und deren Lösungswege.

1.4 Aufbau der Arbeit

Im Rahmen dieser Bachelorarbeit entsteht die Basis für das Design und die Implementierung eines intuitiv bedienbaren und interaktiv nutzbaren virtuellen Labors. Der Grundlagenabschnitt (Kapitel 2) bezieht sich auf die wesentlichen Begrifflichkeiten, die der Leser benötigt, um ein grundlegendes Verständnis für die folgende Konzeption (Kapitel 3) und Implementierung (Kapitel 4) zu entwickeln. Unter anderem erörtert dieser Abschnitt die Grundlagen des Pipettierens und die größten, damit zusammenhängenden, Fehlerquellen, die allgemeine Funktionsweise der virtuellen Realität und vergleichbare Applikationen. Hinzu kommt, in wie fern die virtuelle Realität zum Lernfortschritt beitragen kann sowie eine Beschreibung der verwendeten Software. Im darauf folgenden Abschnitt werden die getroffenen Design- und Konzeptionsentscheidungen, hinsichtlich der zu erforschenden virtuellen Realität und die damit einhergehende gestalterische Umsetzungen, eruiert (Kapitel 3). Dies wird durch die in Kapitel 4 erläuterte Implementierung ergänzt. Das Kapitel der Implementierung bezieht sich auf die Umsetzung der einzelnen Szenen in der *Unity Engine* und den dazugehörigen Interaktionen sowie der internen Steuerung der Applikation. Eine Evaluation (Kapitel 5), hinsichtlich positiver und negativer Aspekte sowie während der Implementierung aufgetretene Probleme, rundet die entstandene Arbeit ab. Abschließend folgen eine Zusammenfassung sowie ein damit einhergehender Ausblick auf die Zukunft der Software (Kapitel 6).

2 Grundlagen

Nach der Einordnung der Aufgabenstellung und der Festlegung der zu behandelnden Themenschwerpunkte soll das nun folgende Kapitel ein grundlegendes Verständnis für die Thematik, den Aufbau und die Funktionsweise des virtuellen Labors vermitteln. Zunächst werden die Grundlagen des Pipettierens erörtert, gefolgt von den, in diesem Zusammenhang möglicherweise auftretenden, Fehlerquellen bei der Durchführung eines solchen Experimentes. Eine kurze Einführung in die Funktionsweise der virtuellen Realität und eine Auseinandersetzung mit vergleichbaren Applikationen sowie der verwendeten Hardware, führt schließlich zu einem Überblick über die Designtheorie der eingesetzten Komponenten. Dem folgt eine Anschauung der verwendeten Engine *Unity*, in welcher abschließend alle Bestandteile zusammenführt werden und welche Zugriff auf das verwendete *SteamVR*-Plugin zulässt. Der Abschluss zeigt auf, welche Vorteile das Lernen mit Hilfe der virtuellen Realität bringt und in wie weit die VR den Nutzer, hinsichtlich seiner Wahrnehmung, beeinflussen kann.

2.1 Grundlagen des Pipettierens

Das virtuelle Labor verwendet als Beispielfall eine Verdünnungsreihe mittels Pipettieren. Um maßanalytische Arbeiten vollziehen zu können, werden Pipetten mit verschiedenen Volumina verwendet, je nach Flüssigkeitsumfang und Art der Lösung [MMSS12, S.9].

Bezüglich des hier beschriebenen Versuchsaufbaus der Verdünnungsreihe ist es am sinnvollsten, sogenannte Mikroliterpipetten zu verwenden. Diese besitzen die Eigenschaft auch kleine Flüssigkeitsvolumen möglichst genau pipettieren zu können. Mikroliterpipetten verwenden auswechselbare Pipetten-Spitzen. Diese bestehen aus Glas oder Polypropylen. Zudem kann durch Drehen eines Rädchens am oberen

Teil der Pipette das gewünschte Volumen eingestellt werden. Das Maximum und Minimum des Volumens lässt sich am Pipetten-Rumpf ablesen. [MMSS12, S.23f.]

Anfänglich entstehen bei dieser Versuchsreihe im realen Experiment viele Fehler, durch welche es zu einer Schädigung der Pipette kommen kann oder weitere Versuchsgegenstände, wie beispielsweise Pipetten-Spitzen oder Reaktionsgefäße, unnötigerweise verbraucht werden. Um Fehler zu vermeiden, sind Wiederholungen des Versuches zwingend. Damit ein erfolgreicher Experimentverlauf gegeben ist, sind gewisse Schritte auszuführen, bzw. Schrittfolgen einzuhalten.

2.1.1 Ablauf

Die zu verwendende Pipette wird dahingehend ausgewählt, dass sie dem zu pipettierenden Volumen am Nächsten steht. Des Weiteren ist mittels des Drehrads, am oberen Teil der Pipette, das genaue Volumen einzustellen. Passend zur Pipetten-Farbe lässt sich die jeweilige Pipetten-Spitze auswählen, welche sich in einer Box mit derselben Farbgebung befindet, soweit Pipette und Spitze vom selben Hersteller stammen.

Der Anwendungsfall bezieht sich auf das Vorwärts-Pipettieren, daher werden die Abläufe diesbezüglich im Nachfolgenden beschrieben. Der Aufnahme- und Abgabeknopf ist zunächst bis zum ersten Widerstand zu drücken. In diesem Stadium kommt die Pipette mit der Flüssigkeit in Berührung. Bei der Aufnahme wird der Aufnahme-/ Abgabeknopf langsam losgelassen und die Flüssigkeit kann in die Spitze einströmen. Dabei darf diese nicht zu tief in die Flüssigkeit eintauchen. Die Pipette ist stets senkrecht zu halten.

Die in der Pipette enthaltene Flüssigkeit ist nun in das nächste, bereitstehende Gefäß zu übertragen. Bei kleinen Mengen sind Reaktionsgefäße zu verwenden, im bestehenden Fall Reagenzgläser. Damit die Flüssigkeit in der Spitze leichter heraus fließen kann, wird die Pipette bei der Ausgabe leicht schräg am Rand des Gefäßes abgelegt. Die Abgabe jeglicher Flüssigkeit aus der Spitze, gelingt durch das gemächliche Drücken des Abgabeknopfes bis zum zweiten Widerstand. Außerhalb des Gefäßes, wenn jegliche Flüssigkeit aus der Spitze entfernt wurde, wird der Druck auf dem Ausgabeknopf gelöst und die verwendete Spitze in einem dafür vorgesehenen Behälter entfernt. [Ewa17]

2.1.2 Fehlerquellen

Zu Beginn der reales Experiment entstehen bei Verdünnungsreihen mittels Pipettieren viele Fehler, die nur durch Übung in Form von mehrmaliger Wiederholung des Versuches vermeidbar sind. Eine Beschreibung dieser potentieller Fehler findet sich im Nachfolgenden:

Das Hinlegen bzw. das Neigen der Pipette während der Versuchsreihe ist zu unterlassen. Dies führt zu Abweichungen in der aufgenommenen und abgegebenen Flüssigkeitsmenge und zur Verschmutzung der Pipette. Auch die Bestimmung der, für das zu pipettierende Volumen, korrekten Pipette muss mit Bedacht vorgenommen werden. Hierbei sollte die obere Volumengrenze der Pipette möglichst nahe an dem zu pipettierenden Volumen liegen. Einstellungen des passenden Volumens sind mittels eines Drehrads am oberen Teil der Pipette vorzunehmen. Das Überdrehen der Volumeneinstellung ist zu unterlassen, da dies zur Schädigung der Pipette führt. Die korrekte Wahl der Spitze lässt sich an der entsprechenden Farbe der Pipette und des Spitzenbehälters, soweit jene vom selben Hersteller stammen, erkennen. Sollte Flüssigkeit aus der Spitze austreten, ist diese entweder defekt oder die Wahl der Spitze war inkorrekt. Ein langsames und kontinuierliches Arbeitstempo gewährleistet, dass keine Fehler durch die zu schnelle Aufnahme oder Abgabe der Flüssigkeit entstehen. Damit keine Luft in die Flüssigkeit gelangt, wird der oberste Bedienknopf vor der Aufnahme bis zum ersten Widerstand gedrückt. Das Ablassen der Flüssigkeit erfolgt an der Wand des Gefäßes, bzw. wenn es sich um besonders kleine Mengen handelt in

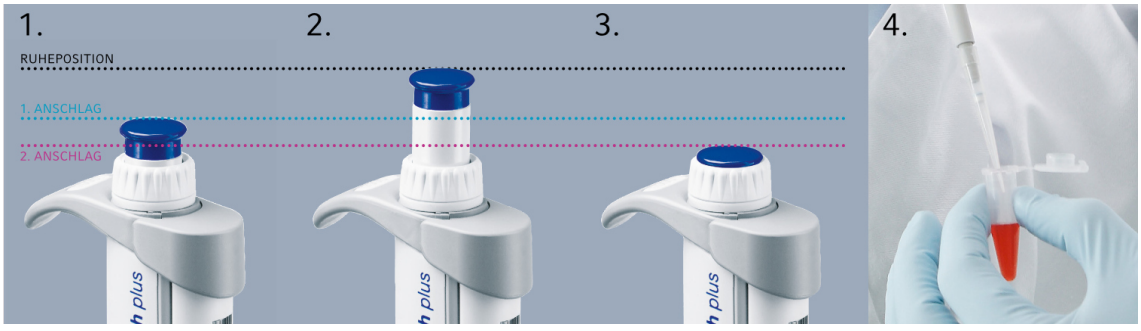

Abbildung 2.1: Stadien des Druckknopfes der Pipette während des Vorwärts-Pipettierens, Quelle: https://online-shop.eppendorf.de/eshopdownload/downloadbykey/112928_Userguide_1, Besucht am: 10.12.2017

Mitten der Flüssigkeit. Hierbei ist der Bedienknopf bis zum zweiten Widerstand zu betätigen. Dieser wird so lange gedrückt gehalten, bis jegliche Lösung aus der Spitze heraus geflossen ist und die Spitze nicht mehr mit der Flüssigkeit in Berührung steht. [Epp13, S.15f.]

2.2 Virtuelle Realität

Schon während der 1960er Jahre beschrieben Forscher, wie beispielsweise Ivan Sutherland, verschiedene technische Systeme und Anwendungen, welche es dem Nutzer ermöglichen, mit einer virtuellen Welt zu interagieren [Rol08, S.130]. Jaron Larnier prägte Ende der 1980er Jahre den Begriff der *Virtual Reality*, dieser fasst die damals aktuelle Entwicklung zur Gestaltung computergenerierter Erlebniswelten und die Thematik Mensch-Computer-Schnittstelle unter einem Begriff zusammen [BKP02, S.2]. Bis heute durchlief die virtuelle Realität eine starke Entwicklung und findet Anwendung in der Visualisierung, Interaktion und Manipulation von komplexen Daten [Bri09, S.6].

Die wichtigsten Merkmale der virtuellen Realität sind zum einen das Arbeiten und Interagieren in einer Echtzeitumgebung, zum anderen die Erstellung einer realistischen Darstellung, welche durch eine naturalistische Betrachtungsweise von Objekten entsteht. Hinzu kommt die Interaktivität, welche die Fähigkeit des Systems beschreibt, sich den Einflussmöglichkeiten des Nutzers anzupassen und diesen aktiv einzubinden [Hau10, S.13, 16].

Um eine uneingeschränkte Interaktivität zu gewährleisten, gilt es folgenden drei Faktoren besondere Aufmerksamkeit zu schenken: Zum einen spielt die Geschwindigkeit der Interaktionen und Informationen eine erhebliche Rolle, da Informationen sonst den Nutzer nicht störungsfrei erreichen. Hinzu kommt die Anzahl der verfügbaren Verhaltensweisen und die Abbildungsfunktion dieser, durch welche das System die Fähigkeit erhält, Aktionen vorhersehbar und natürlich an den Anwender anzupassen. [BKP02, S.13]

Um dies weiter auszuführen, folgt eine summarische Erläuterung der wesentlichen Begriffe.

Die Visualisierung beschreibt beispielsweise die plastische Darlegung von Szenen und Objekten hinsichtlich ihrer Lichtspiegelung und Textur sowie der Farbgebung,

Transformation und Geometrie. Durch eine ansprechende Visualisierung vereinfacht sich die Erstellung einer effektiven Simulation. Simulationen bilden die Dynamik der jeweiligen Szene ab; d.h. Prozesse können wirklichkeitsgetreu nachgestellt werden und lassen somit logische Folgerungen auf die Wirklichkeit zu. Der Begriff Präsenz beschreibt die Fähigkeit des Nutzers, sich selbst als Teil der virtuellen Realität und Umgebung wahr zu nehmen. Im Kontrast dazu steht die Immersion, welche die interaktive Einbindung des Anwenders mit Hilfe der Ein- und Ausgabegeräte beschreibt. Hierbei entsteht eine Wechselwirkung zwischen Präsenz und Immersion, da ein hoher Immersionsgrad potenziell zu einem hohen Präsenzgefühl führt. [Rol08, S.132f.]

In heutiger Zeit sinddigitale Welten nicht nur in der Unterhaltungsindustrie sehr beliebt. So wird die virtuelle Realität für Simulationen im Flugzeug- und Automobilbereich sowie in der Schifffahrt und für Lokomotiven eingesetzt [Bri09, S.8].

2.2.1 Allgemeine Funktionsweise

Rund 70 Prozent der menschlichen Wahrnehmung erfolgt über das Auge, welches dadurch der bedeutendste Eingabekanal ist. In der Computergrafik werden Hinweisreize, wie in der virtuellen Realität die räumliche Wahrnehmung, als *Depth Cues* bezeichnet. [Bri09, S.13]

Die virtuelle Realität beschreibt im Allgemeinen eine digitale, künstlich geschaffene Welt, welche in Echtzeit generiert wird. Der Nutzer kann in verschiedenen Formen mit der Umgebung interagieren und diese manipulieren. Eine naturalistische Betrachtungsweise von Objekten ist charakteristisch für die virtuelle Realität.

Die direkte Einbindung des Nutzers und die Fähigkeit des Systems, sich an die Einflussmöglichkeiten während des Versuches anzupassen, lässt eine starke Immersion entstehen. Diese Immersion kann dazu genutzt werden, den Versuchsprozess so real wie möglich zu simulieren und damit einen wiederholbaren Versuchsablauf zu erstellen. Im vorliegenden Beispielfall bedient sich die Steuerung einer VR-Brille und den dazu gehörigen Controllern.

Zu den wohl bedeutsamsten Funktionsweisen gehört die Erweiterung der Sensomotorik mit Hilfe der virtuellen Realität. Dies ermöglicht die Überbrückung der Raum- und Zeitkomponente während der Anwendung und ermöglicht dem Nutzer erweiterte

Handlungsmöglichkeiten innerhalb der Software. Zudem kann mit Hilfe der VR eine Simulation kognitiver Handlungen stattfinden, welche auf den Anwender unterstützend wirken sowie zu einer Entlastung des menschlichen Intellekts führen. Diese beschleunigen außerdem die für die Lösung der Aufgaben erforderlichen Prozesse und steigert das Lernvermögen. Hinzu kommt die Förderung semantischer Strukturen hinsichtlich der Objektivierung und Materialisierung, welche zudem dazu beitragen, vielschichtige, abstrakte Modelle und Datensätze efassen zu können. Animationen und räumliche Darstellungen können überdies die Erkennung von Mustern erleichtern. Hinzu kommt, dass die virtuelle Realität verstärkend auf die Adaptionsleistung des Individuums wirkt, insofern es nicht zu Nebenwirkungen kommt (siehe Kapitel 5.2.5). [BKP02, S.6]

2.2.2 Vergleichbare Applikationen

Im Folgenden findet sich eine Auflistung an vergleichbaren Applikationen, welche unter anderem zur Ideenfindung und als Orientierungshilfe gedient haben. Diese Softwarebeispiele benutzen zum einen ebenfalls die virtuelle Realität und dienen zum anderen, auf unterschiedliche Art und Weise, als Lernhilfen in den Bereichen der Naturwissenschaften.

MEL Chemistry VR

Mel Chemistry VR beschreibt eine Software, die es dem Anwender ermöglicht, mit Hilfe von 150 Lektionen und Tests, einen Einblick in die Chemie zu erhalten. Dabei bezieht sich das Programm nach amerikanischen Bildungskriterien auf eine Bildungsspanne von Kindergartenkindern bis hin zu jungen Erwachsenen, welche ihr 12. Schuljahr (Highschool) abgeschlossen haben. Das Programm verwendet die virtuelle Realität um komplexe chemische Reaktionen zu veranschaulichen. Der User startet in einem virtuellen 3D-Labor und kann hier das gewünschte Experiment heraussuchen. Dabei werden diese nicht nur durch die Interaktionsmöglichkeiten des Users ergänzt, sondern das Gelernte wird durch theoretische Fragen manifestiert. [Sci17]

Labster

Das Unternehmen *Labster* beschäftigt sich mit der Entwicklung eines interaktiven VR-Labors, basierend auf mathematischen Algorithmen. Ziel ist es, wissenschaftliche Bildung und Lernmethoden zu verbessern. Hierbei setzt die Software sowohl auf Gamification-Elemente als auch auf Storytelling, um Lerninhalte den Anwendern effizienter näher zu bringen. Hinzu kommt ein Punktesystem, welches die Lernfortschritte visualisiert und Aufschluss über Ergebnisse vermittelt. Dieses Punktesystem soll die Motivation des Nutzers fördern und einen kontinuierlichen Lernfluss ermöglichen. [Lab17]

Immersive VR Education Ltd.

Immersive VR Education Ltd. bietet mit ihrer kostenlosen Software *ENGAGE Education* eine *Virtual Reality*-Bildungsplattform, die Lehrern und Schülern ein Mehrbenutzersystem bereitstellt, durch welches es möglich ist, in der erstellten, virtuellen Umgebung zu kommunizieren und miteinander zu interagieren. Bis zu 35 Benutzer können sich dadurch gleichzeitig, von verschiedenen Orten aus, vernetzen. Das Tool bietet Funktionen, welche es ermöglichen, den Unterricht spannender zu gestalten und zu ergänzen sowie eine Aufnahmefunktion des abgehaltenen Unterrichts für spätere Zwecke. [Ltd17]

EON Reality

eon Reality bietet Applikationen an, mit denen Schulungen in verschiedenen Wissensgebieten mit Hilfe von *Virtual Reality* und *Augmented Reality* möglich sind. Hierbei sind viele der Applikationen auch auf mobilen Endgeräten verwendbar, was den Nutzen erheblich steigert. Die Schulungen decken vor allem Bereiche wie Medizin, Archäologie und Ingenieurwesen ab und richten sich vorrangig an Studierende und Arbeitskräfte. [Rea17]

Zusammenfassend ergibt sich, dass sich die Grundlagen der hier genannten Applikationen ähneln und alle dasselbe Ziel verfolgen. Die Hauptidee liegt bei der Wissensvermittlung und Erklärung verschiedener biochemischer Reaktionen. Alle Applikationen haben gemein, dass sie für die Veranschaulichung komplexer Daten die virtuelle Realität verwenden und auf schon bekannte Umgebungen zurückgreifen.

So spielt jede Handlung in einem laborähnlichen Umfeld. Die Bedienung wird über eine VR-Brille, Controller sowie kleinere Interaktionsmöglichkeiten realisiert. Jedes der Laboratorien scheint unbeschränkt erweiterbar zu sein. Dabei bezieht sich die in der Arbeit beschriebene Software jedoch nur auf Studierende, die schon eine gewisse Grundlagenkompetenz in dem Fachgebiet besitzen.

2.2.3 HTC Vive

Die *HTC Corporation* entwickelte in Kooperation mit *Valve* eine VR-Plattform, mit der es möglich ist, Inhalte hinsichtlich der virtuellen Realität sinnvoll zu nutzen. Hierbei werden interaktive Welten kreiert und mit Nutzern aus der ganzen Welt geteilt. Somit kann jener Inhalt nicht nur der Spieleindustrie zu Gute kommen, sondern beispielsweise auch Lehrinhalte bieten. [Cor17b]

Die *HTC Vive* besteht aus einem Headset sowie Controller und Basisstationen. Das Headset ist ein *Head-Mounted-Display*, d.h. ein visuelles Ausgabegerät, welches auf dem Kopf getragen wird. Das Sichtfeld hat einen Umfang von 110° und trägt somit einen großen Teil zur Immersion bei. Durch 32 Headset Sensoren wird eine 360°- Bewegungsverfolgung ermöglicht. Die Auflösungsrate beträgt $90\,Hz$ bei einer Bildauflösung von 2160×1200 Pixel. Die Controller besitzen ein Multifunktions-Touchpad sowie einen Trigger mit zwei Druckstufen und werden durch 24 Sensoren präzise geortet. Die zwei Basisstationen werden drahtlos synchronisiert und dienen der Ortung. [Cor17a]

2.3 Design

> „Design itself is an iterative, creative, but controlled process. It needs clear definitions and controlled aims. In all disciplines the role of the designer involves specifying the principles of: need, describing the vision, and producing the result.“ [KLH10, S.8]

Gutes Design verbessert zum einen die Qualität des Produktes, zum anderen fördert es das Verständnis des gegebenen [KLH10, S.17]. So kann sich beispielsweise ein ungenügend konzipiertes Interface nachteilig auf die virtuelle Erfahrung, hinsichtlich Krankheitserscheinungen oder Unbehagen, auswirken [JK05, S.7]. Der entstandene

Kontext erhält erst durch die Verwendung einen Nutzen. So beinhaltet das Design sowohl den Raum und die Objekte des Produktes als auch die damit verbundenen Erzählungen und implementierten Verhaltensweisen. [SZ04, S.41]

Der nun folgende Abschnitt befasst sich mit den grundlegenden 3D- und 2D-Inhalten sowie deren Aufbau und Entstehungsweise. Hierbei werden Begriffe wie beispielsweise Modellierung, Texturierung und Grafiken aufgegriffen und genauer erörtert.

2.3.1 3D-Objekte

Modellierung

Dreidimensionale Modelle bestehen aus Flächen, Kanten und Punkten; auch Polygone, Edges und Vertices genannt. Diese lassen sich mit Hilfe von Modifikationen umformen und ausbauen. Dabei beschreiben Polygone einzelne Flächen des Objektes, welche im geeignetsten Fall drei bzw. vier Ecken besitzen. Edges sind die Kanten der Objekte und Vertices die einzelnen Punkte, welche die Kanten miteinander verbinden. Die hinsichtlich technischer Anwendungen meist verwendeten Programme sind *Autodesk 3ds Max* und *Cinema4D*; hinsichtlich Design wird *Autodesk Maya* empfohlen [Hau10, S. 88].

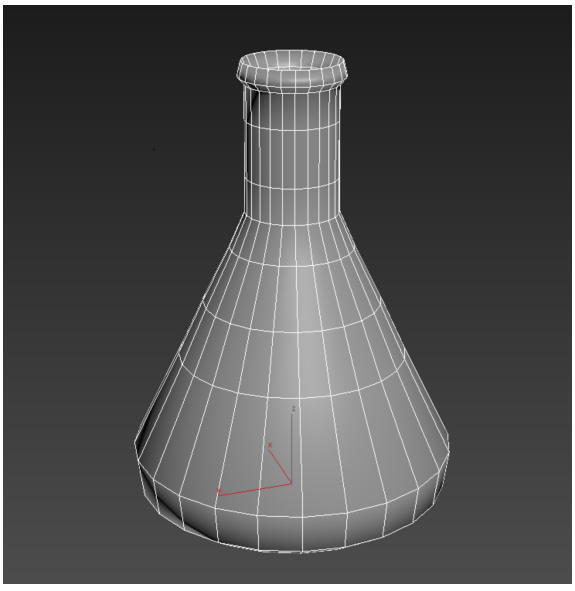

Abbildung 2.2: Sauber modelliertes 3D-Modell eines Erlenmeyerkolbens

Eine ausgezeichnete kostenlose Variante bietet *Blender*. Des Weiteren ist bei High-Poly-Projekten, Modellen mit einer hohen Polygonzahl, und Sculpting die Software *ZBrush* von *Pixologic* als eines der führenden Software-Produkte anzuführen. In der Umsetzung des virtuellen Labors wurden auf die Anwendungen *Autodesk 3ds Max* und *Blender* zurückgegriffen.

UV-Mapping

Eine *UV Map* stellt ein 2D-Bild dar, welches das entfaltete 3D-Modell repräsentiert. Durch die Verwendung dieser ist es möglich, dem Modell eine akkurate Textur zuzuweisen, ohne dass sich Passagen überschneiden. Die meisten 3D-Programme haben zur Entwicklung der Map ein eingebautes Tool, welches die Erstellung vereinfacht bzw. teilweise komplett übernimmt. Die erstellte, zweidimensionale Textur wird auf einem Koordinatensystem abgebildet, welches die Achsenbezeichnung U und V trägt. [Sei15, S.136]

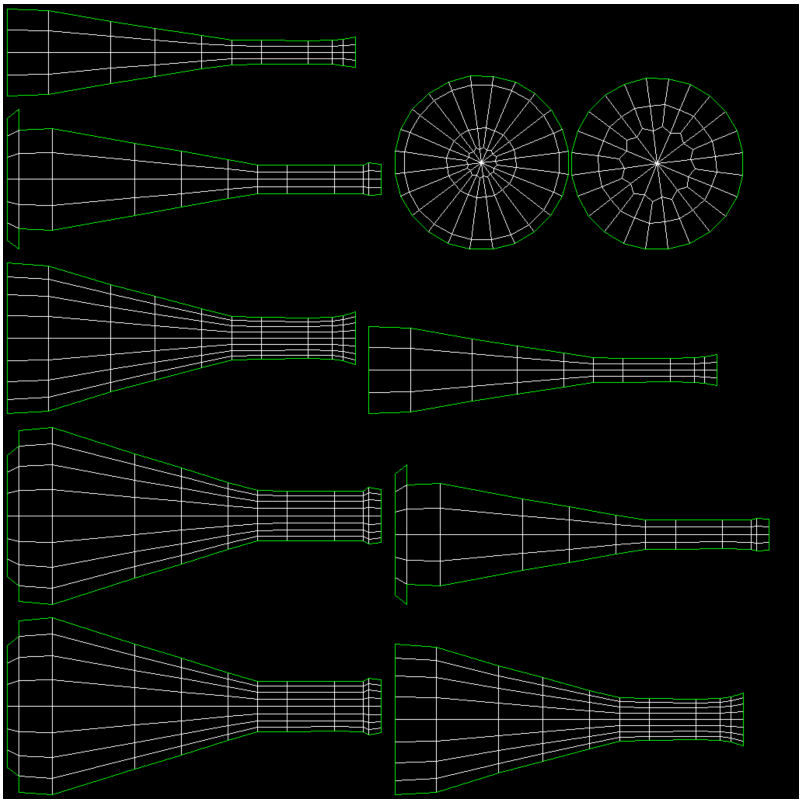

Abbildung 2.3: Beispiel einer UV-Map (Erlenmeyerkolben)

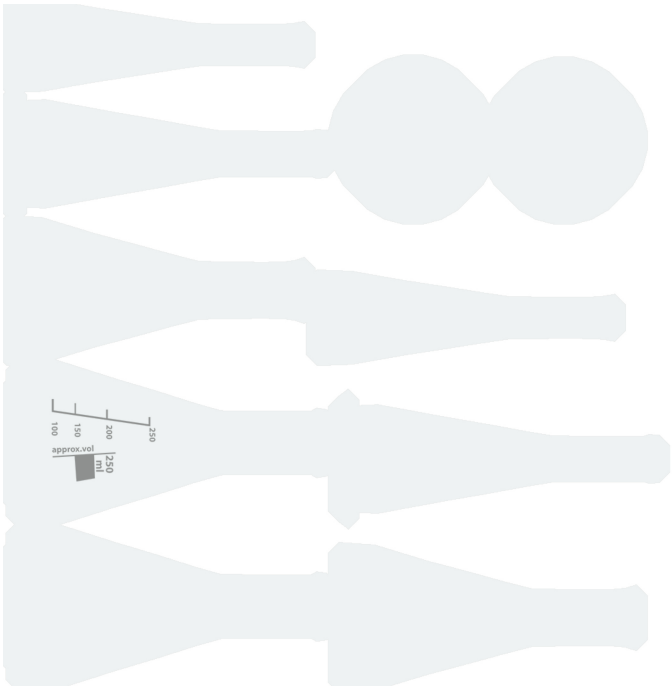

Abbildung 2.4: Beispiel-Textur eines Erlenmeyerkolbens, welches mit der oben beschriebenen UV-Map übereinstimmt

Textur

Texturen bestehen aus Grafiken, welche so an die *UV-Map* eines Objektes angepasst sind, dass sich nach dem Rendern ein stimmiges Bild ergibt. Durch sie erhält das Objekt weitere Details, die durch einfache Low-Poly-Modellierung, Modelle mit einer niedrigen Polygonzahl, nicht darstellbar sind. Der Anwendungsfall enthält hinsichtlich der Texturen einfache Diffuse-Maps, welche lediglich Farbinformationen enthalten.

2.3.2 2D-Elemente

Grafiken

Die Erstellung der Grafiken bezieht sich im Anwendungsfall auf die Benutzeroberfläche sowie die erforderlichen Hintergründe und Buttons. Diese entstanden in *Adobe Illustrator*, da die Entwicklung von Vektorgrafiken mit Hilfe dieser Software

möglich ist. Vektorgrafiken speichern, im Gegensatz zu Rastergrafiken, welche jedem Bildpunkt einen Farbwert zuordnen, die genaue Bildbeschreibung ab. Dadurch sind Vektorgrafiken daher effizienter einsetzbar, da sie verlustfrei skalierbar sind.

Text

Texte sind bei Visualisierungen, wie beispielsweise genaue Ergebniswerte oder dem *Hauptmenü*, unumgänglich. Um eine saubere und anschauliche Textdarstellung zu gewährleisten, sind einige Designschritte zu beachten, die sowohl im *Hauptmenü* als auch in der *Aufgabenstellung* sowie im *Fragenkatalog* beachtet wurden.

Die dazu wichtigsten Regeln sind zum einen die Beachtung der Distanz zwischen den Texten. Hierbei gilt, dass sich nahe stehende Texte als eine Gruppe wahrgenommen werden. Zum anderen gilt, dass diese in der gleichen Art und Weise gestaltet sein müssen. D.h. dass sie sich in Form, Farbe und Kontinuität einteilen lassen. [KLH10, S.132] Durch Einhaltung dieser Regeln entsteht eine saubere und für den Betrachter schlüssige Visualisierung.

2.4 Unity

Unity ist eine Game-Engine, die das Entwickeln von 2D- und 3D-Applikationen mit Hilfe von verschiedenen Tools ermöglicht und vereinfacht. Sowohl Werkzeuge für Partikeleffekte als auch zur Landschaftsgestaltung sowie für Animationen sind in der Software enthalten. Zudem ist es möglich in der Engine *C#*- sowie *JavaScript*-Skripte einzubinden. [Sei15, S.2]

Die *Unity Engine* wird global verwendet, um Spiele und andere Anwendungen umzusetzen und ist für diesen Zweck die meist verwendete Software. Die erstellten Applikationen können für beinahe alle Geräte und jedes Medium bereit gestellt werden, wie beispielsweise Konsolen, PC, mobile Endgeräte oder für Head-Mounted-Displays. Weitere Ressourcen sind mit Hilfe des *Assetstores* und einer Internetverbindung downloadbar. *Unity* steht sowohl für kommerzielle Zwecke, als auch für Hobby-Entwickler zur Verfügung. [Tec17b]

2.4.1 Grundlagen

Oberfläche

Die Oberfläche der *Unity Engine* besteht aus mehreren Fenstern. Diese sind an die Ansprüche des Users individuell anpassbar. Im Standard-Layout sind folgende Fenster eingestellt: *Hierarchie, Szenen-Ansicht, Game-Ansicht, Inspektor, Projekt-Fenster* und die *Console*. In der *Hierarchie* sind alle sich in der Szene befindenden Objekte aufgelistet. Die *Szenen*-Ansicht ermöglicht es, die gerade geöffnete Szene zu verändern und zu gestalten; durch Drücken des Start-Buttons wechselt die Ansicht dieses Fensters automatisch in die *Game*-Ansicht, welche eine Vorschau der Applikation aufzeigt. Der *Inspektor* legt dar, welche Funktionen und Eigenschaften des jeweiligen Objektes verändert werden können und welche anpassbaren Komponenten es besitzt. Am unteren Rand befindet sich das *Projekt*-Fenster, in welchem alle für das Projekt dienlichen *Assets* aufgelistet sind. Dazu gehören beispielsweise 3D-Objekte, Grafiken, Animationen oder Skripte. Im Layer darunter befindet sich die *Console*, welche auch nach betätigen des Start-Buttons sichtbar ist. Alle in der Applikation befindlichen Fehler und Hinweise sind hier für den Nutzer aufgelistet.

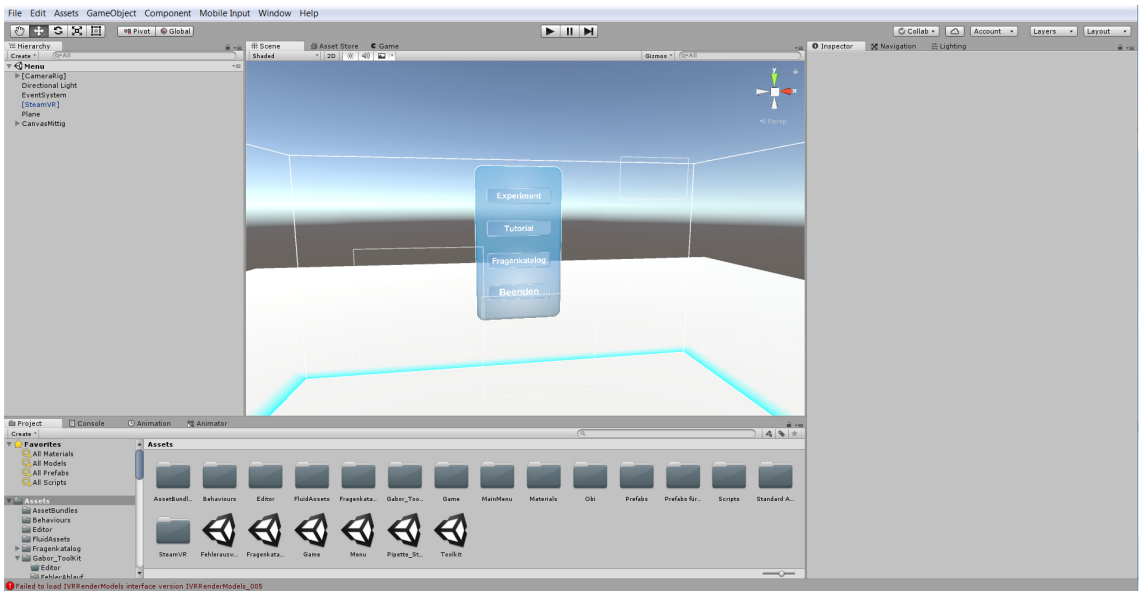

Abbildung 2.5: Layout eines Standard-Unity-Projekts mit allen beschriebenen Fenstern

Szene

Jede *Szene* beschreibt eine neue Umgebung der Applikation. Hier ist es möglich, beispielsweise UI-Elemente, 3D-Objekte und Animationen zu importieren und anzuordnen. Es können beliebig viele neue Szenen instanziiert werden. Um diesen einen Index zuzuordnen, muss die neu erstellte Szene in den *Build Settings* der Applikation hinzugefügt werden. Ein Szenenwechsel erfolgt dann mit der Funktion `SceneManager.LoadScene()`.

Prefab

Ein *Prefab* beschreibt die Vorlage eines Objektes, welche mit allen individuell eingestellten, essentiellen Komponenten abgespeichert und bei Bedarf geladen wird. Jenes Objekt kann beliebig häufig, unabhängig der Szene, geladen werden und besitzt immer dieselben Eigenschaften.

Kamera

Die Kamera zeigt die Sicht des Nutzers auf die Applikation. Sobald eine neue Szene angelegt wird, generiert sich automatisch eine Hauptkamera. Eigenschaften, wie beispielsweise die Sichtweite sowie die Größe des Sichtfensters, sind dabei individuell einstellbar. In der beschriebenen Anwendung, wird kein Gebrauch von der herkömmlichen Hauptkamera gemacht, sondern die in *SteamVR* vorgefertigte Kamera verwendet (Kapitel 2.4.2). Diese bewegt sich immer simultan zum Headset.

Directional Light

In *Unity* sind verschiedene Lichtquellen enthalten: *Directional Light, Spotlight, Point Light* und *Area Light*. Ihre Eigenschaften und Wirkungsweisen auf die Szene kann im *Inspektor* angepasst werden. Dazu zählt beispielsweise die Lichtintensität und der Radius, die Farbe des Lichtes sowie welche Art des Schattens betroffene Objekte werfen. Das im Projekt verwendete *Directional Light* beleuchtet die gesamte Umgebung aus einer bestimmten Richtung.

Rigidbody

Alle mit einem *Rigidbody* versehenen Objekte, lassen sich von der *Unity*-Physik beeinflussen. Hierbei sind den Objekten sowohl eine Kraft als auch ein Drehmoment sowie Gravitation zuzuweisen [Sei15, S.210].

Kollision

Um Kollisionen in der Engine zu simulieren sind sogenannte *Collider* unabdingbar. Hierbei gibt es vier verschiedene Arten *Box Collider, Sphere Collider, Capsule Collider* sowie den *Mesh Collider*; letzterer passt sich der Form des jeweiligen Objektes an. Um Kollisions-Methoden auszuführen, muss ein *Rigidbody* auf einem der kollidierenden Objekte liegen [Sei15, S.219]. Kollisions-Methoden werden entweder am Anfang bzw. Ende einer Kollision aufgerufen sowie wenn zwei Objekte dauerhaft kollidieren (siehe Kapitel 4.3.2).

Graphical User Interface

Das in *Unity* enthaltene UI-System (*uGUI*) ermöglicht es dem Entwickler 2D-Objekte wie normale Game-Objekte zu behandeln und diese dementsprechend verändern zu können. Ein *Canvas* bildet das Hauptobjekt einer jeden Benutzeroberfläche. Dieses passt sich Standardmäßig durch die Einstellung *Screen Space-Overlay* an die verfügbare Bildschirmbreite an. Alle weiteren Objekte wie beispielsweise *Panels*, *Buttons*, Texte etc. werden als Kindobjekte instanziiert.

PlayerPrefs

Die Methode `PlayerPrefs()` ermöglicht es unabhängig vom System Werte zu speichern. Hierbei stehen die Datentypen *string, int* und *float* zur Verfügung; d.h. Zeichenketten aus Ziffern, Wörtern oder Zeichen sowie ganze Zahlen und Fließkommazahlen. Der erste eingegebene Parameter beschreibt die ID des Wertes, der zweite den eigentlichen Wert des Datentyps. [Sei15, S.93f.].

2.4.2 Steam VR

SteamVR ist ein Plugin von *Valve Corporation*, welches für die *Unity Engine* zur Verfügung steht. Die *Unity*-Version 4.7.1 ist Mindestanforderung für das effiziente Einbinden [Cor17c]. Das Plugin funktioniert mit Hilfe des *Steam*-Clients [Mur17, S.25].

Im Allgemeinen beschreibt *SteamVR* eine Schnittstelle zwischen dem *HTC Vive*-System und dem Computer des Anwenders. Diese Schnittstelle bildet die Voraussetzung für Konfigurationen an der Hardware. Dabei werden die Daten und die Position des Headsets und der Controller dauerhaft berechnet und von der Software verarbeitet. [Hä17, S.8].

2.5 Schnittstellen

Schnittstellen (*Interfaces*) definieren Gemeinsamkeiten, wie beispielsweise Methoden, in unterschiedlichen Klassen. Um eine Klasse als Schnittstelle zu definieren, wird das Schlüsselwort `interface` vor dem eigentlichen Klassennamen verwendet. Um jeder Klasse, die von dem *Interface* erbt, Zugriff auf die Komponenten zu erlauben, ist es erforderlich Variablen und Methoden als beispielsweise *public* (öffentlich sichtbar) zu deklarieren. Methoden und Variablen werden dann von den erbenden Klassen nach Belieben verändert. Damit die erbende Klasse die Schnittstelle verwenden kann, ist es zwingend, hinter dem Doppelpunkt des Klassennamens die *Interface*- Klasse zuzufügen. Zusätzlich müssen alle, in der Schnittstelle definierten Methoden auch in den davon erbenden Klassen vorhanden sein.

2.6 XML

Die *eXtensible Markup Language* beschreibt eine Metasprache, welche in dem Austausch und der Speicherung von Daten Verwendung findet. Hierbei ist die Grundidee, Informationen inhaltlich und strukturell zu beschreiben und diese darzustellen. Der Datenaustausch zwischen unterschiedlichen System-Plattformen und Anwendungen ist dabei möglichst einfach und flexibel. Die Speicherung der einzelnen Informationen

gelingt über eine Textdatei, in welcher einzelne Elemente mittels *Tags* gekennzeichnet und begrenzt sind. Da XML eine flexible Sprache ist, gibt es Grundregeln, welche einzuhalten sind. Dies dient der raschen und effizienten Analyse der Dokumente. Dokumente, die diese Grundregeln einhalten, erhalten die Bezeichnung *Wohlgeformte Dokumente*. So beinhalten die Grundregeln, dass das Dokument ein Stammelement besitzen muss, dass Start- und End-Tags kompatibel und richtig geschachtelt sind, dass Sonderzeichen ersetzt wurden, dass Attribute nur einen Wert besitzen sowie in Anführungszeichen gesetzt sind und dass das Dokument keine unzulässigen Zeichenfolgen besitzt. [DG15, S.645f., S.650]

2.7 Lernmethoden

Die virtuelle Realität bietet eine ausgezeichnete Grundlage, Wissen zu vermitteln. Dabei zeichnet sie sich durch eine gute Verfügbarkeit, eine effektive Vermittlung von Basiswissen und die Möglichkeit für den Nutzer selbstständig zu lernen aus [BBB+03, S.12].

Hierbei werden Inhalte stärker versinnlicht, d.h. die erfahrenen Reize können auf unterschiedliche Sinneskanäle ausgeweitet werden. Dahingehend ist es möglich, Lerninhalte sowohl mit haptischen als auch mit auditiven Rückmeldungen greifbarer zu gestalten. Im Anwendungsfall ist die Veranschaulichung abbildungsgetreu gestaltet; Darstellungen sind möglichst realistisch und illusionistisch. Der erhöhte Realismus steigert die Wahrscheinlichkeit, Lernerfahrungen auf das reale Umfeld anzuwenden und zu transferieren. [BKP02, S.110ff.]

Durch eine gezielte Ausübung der Anwendungsaufgaben kommt es zu einem erfolgreichen Fähigkeitserwerb. Mit Hilfe von didaktischer Reduktion, Orientierung an den Lerngruppen und geschickt eingesetzter Transparenz kann dieser Fähigkeitserwerb verbessert und ausgebaut werden. Zudem sind Lernzeiten flexibel sowie individuell und begangene Fehler entsprechend visualisierbar. [JH09, S.6ff.]

Ein Nachteil am Wissenserwerb mit Hilfe der virtuellen Realität ist jedoch, dass dem Nutzer eine Diversifikation von Informationen präsentiert wird, welche grundlegend nicht lernrelevant erscheinen und dieser dadurch den Fokus auf die eigentliche Aufgabe verlieren kann [BKP02, S.112]. Zudem kann es bei geringen Vorkenntnissen zu einer Überlastung kommen, da der Lernende nicht nur durch den Lernprozess

an sich, sondern zusätzlich durch die Anwendung der Software beansprucht wird [JH09, S.8].

3 Konzeptionierung des virtuellen Labors

Um das vorherige Kapitel hinsichtlich der Grundlagen der Thematik und der elementaren Funktionsweisen des virtuellen Labors zu ergänzen, beschäftigt sich das folgende Kapitel mit der Konzeptionierung des Labors. Hierbei wird das Konzept der Erstellung der essenziellen Objekte und den dazu gehörigen Szenen in der *Unity Engine* erläutert. Die Darlegung der Einbindung der VR-Brille und den dazugehörigen Controllern bildet den Schluss des Kapitels, gefolgt von einer kurzen Zusammenfassung.

3.1 Fehlerquellen

Um einen klaren Versuchsablauf zu garantieren, ist zunächst zu entscheiden, welche Fehler innerhalb der Software theoretisch umsetzbar sind.

Simulierbar sind beispielsweise die richtige Auswahl der Pipette und die Stellung und Bewegung dieser, da dies simple analysierbare Parameter sind. Für ein erfolgreiches Pipettieren ist die Pipette stets senkrecht zu halten. Sollte es während des Versuchs zu einer Schräglage kommen, sind Abweichungen in der pipettierten Menge problemlos abbildbar. Die korrekte Wahl der Pipetten-Spitze ist essentiell und problemlos nachprüfbar, da jede Spitze nur mit einem Pipetten-Volumen übereinstimmt. Die Volumeneinstellungen können mit Hilfe der Anzeige implementiert werden. Auch hier ist es möglich einen Maximalwert zu definieren und damit das Überdrehen des Volumens zu simulieren. Des Weiteren beschränkt sich die Arbeit vorerst auf das Vorwärts-Pipettieren.

Fehlerquellen, wie das Herauslaufen der Flüssigkeit bei einer defekten oder falsch gewählten Spitze, sind vorerst nicht möglich, da dies den Umfang der Arbeit übersteigen würde.

3.2 Objekterstellung

Die designtechnischen Aspekte, die im Folgenden beschrieben werden, beinhalten die allgemeinen Gestaltungskonzepte sowie die Farbgebung der für das virtuelle Labor relevanten Objekte. Die Designanforderungen sind dahingehen umgesetzt, dass sie möglichst die Funktionalität und Interaktivität fördern und dadurch eine gute Bedienbarkeit gewährleistet ist. Eine zweckmäßige Gestaltung der Objekte bringt dem Nutzer essentielles Wissen und Informationen verständlicher näher.

In der virtuellen Realität, in Kombination mit einer realistischen Simulation, ist es essenziell, dass der Nutzer eine, dem Vorbild entsprechende, Perspektive auf die einzelnen Objekte erhält, um den essentiellen Realismus zu bewahren. Dies ist nur dann gegeben, wenn die Formen und Farben dem normalen Umfeld des Nutzers entsprechen und in jeglicher Hinsicht nachvollziehbar erscheinen. Somit ist es durchaus erforderlich, in der Konzeption der Objekte auf eine passende Größe und zweckmäßige Textur zu bestehen. Um die einzelnen Komponenten sinnvoll zu verbinden, ist zudem ein möglichst hoher Detailgrad von Nutzen. Dieser ist allerdings im gegebenen Fall durch einen niedrigen Polycount, eingeschränkt, da dieser sonst zu Performanz-Problemen in der Software führt.

Vorarbeit

Die Entwicklung der für das virtuelle Labor benötigten 3D-Modelle fand in dem Programm *Autodesk 3ds Max 2015* statt. Die Autorin entschied sich auf Grund von umfangreichen Vorerfahrungen für diese Software. Um einen Eindruck der erforderlichen Modelle zu erhalten und eine strukturierte Konzeptionsphase zu gewährleisten, war die Erstellung einer Assetliste unumgänglich. Die Assetliste ist dahingehend gegliedert, dass alle verwendbaren Objekte mit den dazu gehörigen Einzelteilen aufgelistet sind. Auf dieses Konzept greift der Modellierer während der eigentlichen Erstellung zurück und entscheidet, welche der Objekte essentiell sind und welche nur als Zusatz dienen. Des Weiteren erfolgte die Suche nach geeigneten Referenzbildern, da die späteren Modelle relativ realitätsnah gestaltet sein sollen. Um dies zu gewährleisten, werden in der Praxis verwendete Pipetten der Marke *Eppendorf* betrachtet. Diese verfügen über alle, für die Anwendung benötigten Funktionen.

Modellierung

Das virtuelle Labor sollte in seiner Gestaltung echten Laboratorien in nichts nachstehen. Um dies zu gewährleisten, wurden zunächst reale Laboratorien einer Analyse unterzogen. Des Weiteren entstand auf dieser Grundlage basierend die unter *Vorarbeit* beschriebene Assetliste. Vorgefertigte, online verfügbare Objekte sind entweder mit Kosten verbunden oder nicht ausreichend ausgestaltet. Daher erfolgte eine von Grund auf eigenständige Modellierung dieser. Die Modelle sind hierbei so konzipiert, dass nach Vollendung des Modells jedes Element nach Belieben animierbar bzw. im Nachhinein veränderlich ist. Dies war eine Hauptanforderung, da die Anwendung zu einem späteren Zeitpunkt erweitert und verbessert werden soll.

UV-Mapping

Um vollendete Modelle texturieren zu können, ist in den meisten Fällen eine *UV-Map* erforderlich. Diese muss so konzeptioniert sein, dass sie nahtlos auf das 3D-Objekt passt. Die Konzeption sieht daher vor, dass das Modell schon nach der Modellierung in essentielle Einzelteile aufgeteilt ist. Die *UV-Map* beschreibt ein Abbild des vollkommen aufgeklappten Modells, mit all seinen Polygonen. Ohne ein sinnvolles *UV-Mapping* ist eine Texturierung nicht oder nur schwer möglich, da das 2D-Abbild der Polygone übereinander liegt oder sich überschneidet und sich somit auch die Texturen überlagern.

Texturierung

Zunächst fand eine Recherche hinsichtlich der benutzbaren Texturen statt. Diese müssen sowohl frei von Lizenzen als auch zur Wiederverwendung und Veränderung gekennzeichnet sein. Ist dies gegeben, sind sie für das Projekt zulässig. Die Texturen müssen zudem eine angemessene Auflösung besitzen. Dies ist von großer Relevanz, da sie bei einer zu starken Vergrößerung verpixeln. Bei einigen Texturen ist es möglich, sie hinsichtlich ihrer Ränder so zu gestalten, dass sie unbegrenzt, als schlüssiges Muster aneinander passen. Diese Eigenschaft ist vor allem bei größeren Flächen, wie beispielsweise Wänden oder Böden von großem Nutzen. Hierbei ist zu beachten, dass die Doppelung des verwendeten Musters für den Anwender nicht erkennbar wird, da die Anwendung unterdessen an Immersion verliert.

Export

Für den Export der benötigten Dateien bieten sich zwei Dateitypen, *.obj* und *.fbx*, an. Der Unterschied zwischen *.obj* und *.fbx* besteht darin, dass *.obj* Dateien zwar von den meisten Programmen importiert werden können, jedoch *.fbx* Dateien zusätzlich beispielsweise Animationen enthalten können. Daher fiel die Wahl bei der Umsetzung auf das Format *.fbx*.

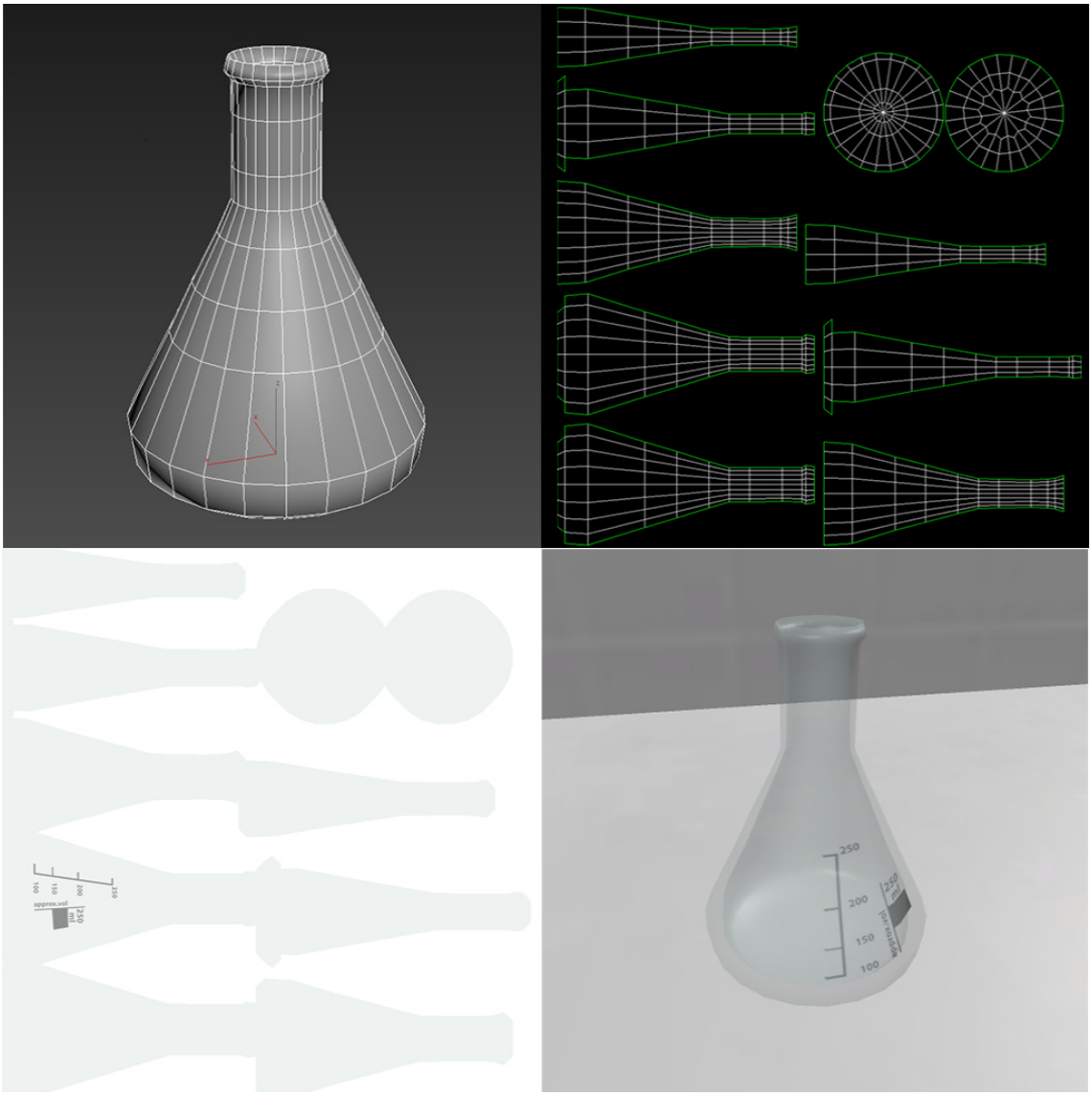

Abbildung 3.1: 3DModell-Entstehungs-Zyklus eines Erlenmeyerkolbens

2D-Inhalte

Das Konzept der 2D-Inhalte beschränkt sich im Anwendungsfall auf die Elemente des *Hauptmenüs* und der Gesamtheit der damit in Verbindung stehenden Inhalte. Die Grafikinhalte des *Hauptmenüs* bestehen aus den Schaltflächen und dem eigentlichen Hintergrundobjekt. Diese sollen visuell voneinander abhängig sein und sich designtechnisch nicht widersprechen. Dafür sind passende Farben und Formen auszusuchen. Die darauf aufbauenden Menüs, wie das *Tutorial*, die *Auswertung* oder die *Auswahl* des Experiments, sollen in ihrem Design an dem des *Hauptmenüs* orientiert sein.

3.3 Szenenbau

Um die Konzeption der einzelnen Szenen zu beginnen, war zunächst eine Entscheidung zu treffen, welche Umgebungen für die Software relevant sind. Neue Szenen dienen der Strukturierung der Applikation und umgehen Performanz-Probleme, da nur für die jeweilige Szene relevante Elemente geladen werden müssen. Ein funktionierendes, gut strukturiertes *Hauptmenü*, welches dem Nutzer eine einfache Führung durch das Programm erlaubt, war daher unumgänglich. Des Weiteren wurde in Kooperation mit der *BIOTACMI* entschieden, dass ein vorangehender *Fragenkatalog* vor Experimentbeginn, eine gute Wiederholung der theoretischen Inhalte darstellt. Dieser muss allerdings auch vom *Hauptmenü* aus aufrufbar sein. Nach erfolgreicher Beendigung des *Fragenkatalogs* soll der Nutzer Zugang zum eigentlichen Experiment erhalten, welches im virtuellen Labor stattfindet. Hierbei erhält dieser eine klare Aufgabenstellung in Bezug zum Versuch und abschließend eine eindeutige Analyse, die Auskunft darüber gibt, welche Fehler während des Versuchs begangen worden sind. Dies gewährleistet einen klar verständlichen Ablauf des Experiments. Um Ordnung zu schaffen und spätere Anpassungen zu erleichtern, ist es ratsam, für jede Thematik einen Ordner zu erstellen. Zudem ist hinsichtlich der Ausleuchtung der Szenen darauf zu achten, dass die Objekte keinen störenden Schatten werfen und damit die Handhabung erschweren.

3.3.1 Menü

Das *Hauptmenü* wird benötigt, um den User durch die Applikation zu führen. Hierbei ist es ratsam, sich zunächst Gedanken über das Aussehen und die benötigten Unterpunkte zu machen. Um die Bedienung zu erleichtern, sollten Elemente gewählt werden, welche dem Nutzer bereits bekannt sind. Die dafür am besten geeigneten Objekte sind in diesem Fall herkömmliche *Buttons*, da die Handhabung intuitiv und leicht erkennbar erscheint. Jene *Buttons* benötigen, für ein klareres Verständnis, bei Berührung ein visuelles Feedback. Das *Hauptmenü* besteht aus den Unterpunkten: *Experiment*, *Tutorial*, *Fragenkatalog* und *Beenden*, welche sich gegliedert untereinander befinden. Die Auswahl der *Buttons* sollte über bloße Berührung und der darauf folgenden Bestätigung möglich sein.

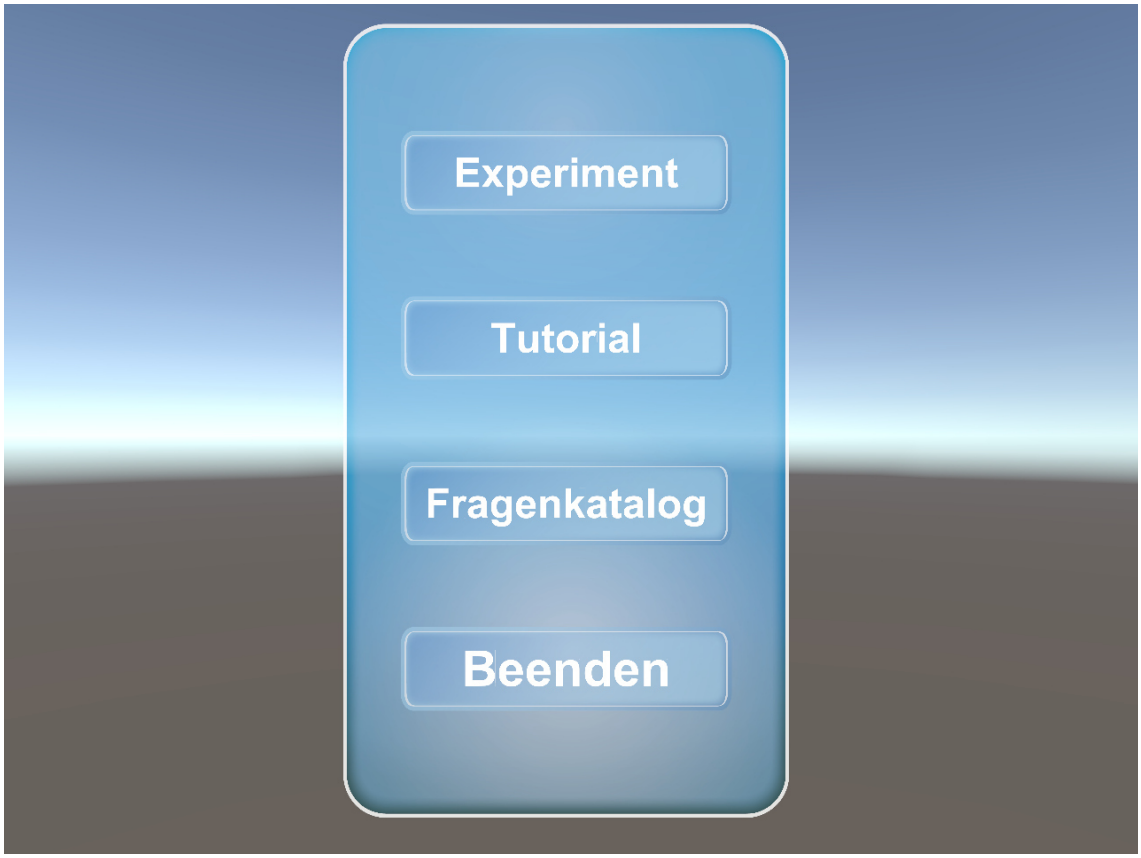

Abbildung 3.2: Frontale Ansicht des Hauptmenüs in der Unity-Engine

Experiment

Durch die Auswahl des Unterpunktes *Experiment* erhält der User Zugriff auf ein weiteres *Panel*. Mit Hilfe dieses *Panels* soll es möglich sein, das gewünschte Experiment auszuwählen. Hierbei ist das *Panel* so gegliedert, dass sich in der linken unteren Ecke der *Zurück*-Button befindet. Dieser befindet sich auf jedem *Panel* an der gleichen Stelle, um das Zurechtfinden zu erleichtern. Darüber findet sich, nach Auswahl des gewünschten Experiments, eine Experimentbeschreibung, damit dem User seine Auswahl erneut bewusst wird. Rechts neben der Beschreibung gibt es *Buttons* inklusive passender Visualisierungen zu den verschiedenen Versuchen zur Auswahl. Die Visualisierung dient hierbei erneut dem Verständnis. Erst nach dieser Auswahl lässt sich der *Start*-Button, welcher sich in der rechten, unteren Ecke befindet, betätigen, welcher den Nutzer zum *Fragenkatalog* weiter leitet. (Kapitel 3.2.3)

Tutorial

Der *Tutorial*-Button führt den User zu einer Grafik, welche die Handhabung der Controller beschreibt. Diese ist vor allem für neue Nutzer essentiell, damit der Einstieg in den Versuch erleichtert ist. Hierbei werden die unterschiedlichen Eingabemethoden visualisiert und der Anwender bekommt einen Überblick über die Aktionsmöglichkeiten mittels der Controller. Auch hier ist es erforderlich, dass sich ein *Zurück*-Button auf dem *Panel* befindet.

Fragenkatalog

Der *Fragenkatalog*-Button soll den User in eine neue Szene weiterleiten, in welcher die, aus den Lernmodulen des Studiengangs hervorgehenden, theoretischen Grundlagen des Pipettierens abzufragen sind. (Kapitel 3.2.2)

Beenden

Durch die Auswahl des Unterpunktes *Beenden*, soll die Anwendung komplett abgebrochen werden und der Anwender zurück zu seinem Desktop gelangen.

3.3.2 Fragenkatalog

Der *Fragenkatalog* fragt die, aus den Modulen hervorgehenden, theoretischen Grundlagen des Pipettierens ab. Um eine gute Übersicht zu gewährleisten, ist dieser wie ein ganz normaler Test aufgebaut und findet sich auf einem Klemmbrett wieder. Es ist wichtig, den Nutzer nicht mit einer zu großen Menge an Aufgaben zu überfordern, da dieser sonst schnell die Motivation oder den Überblick verlieren kann. Um ein schnelles weiterkommen zu gewährleisten und dennoch das theoretische Wissen zu manifestieren, fiel die Wahl auf Multiple-Choice-Fragen. Hierbei hat der Anwender bei jeder Frage drei Auswahlmöglichkeiten, von denen allerdings nur eine korrekt ist. Die Fragen sind untereinander angeordnet und das Feld zum Ankreuzen der Antwort befindet sich auf der linken Seite der Antwortmöglichkeit. Dabei sind sie die Fragen visuell als eine Einheit zu vernehmen, da die Texte gruppenweise angeordnet sind.

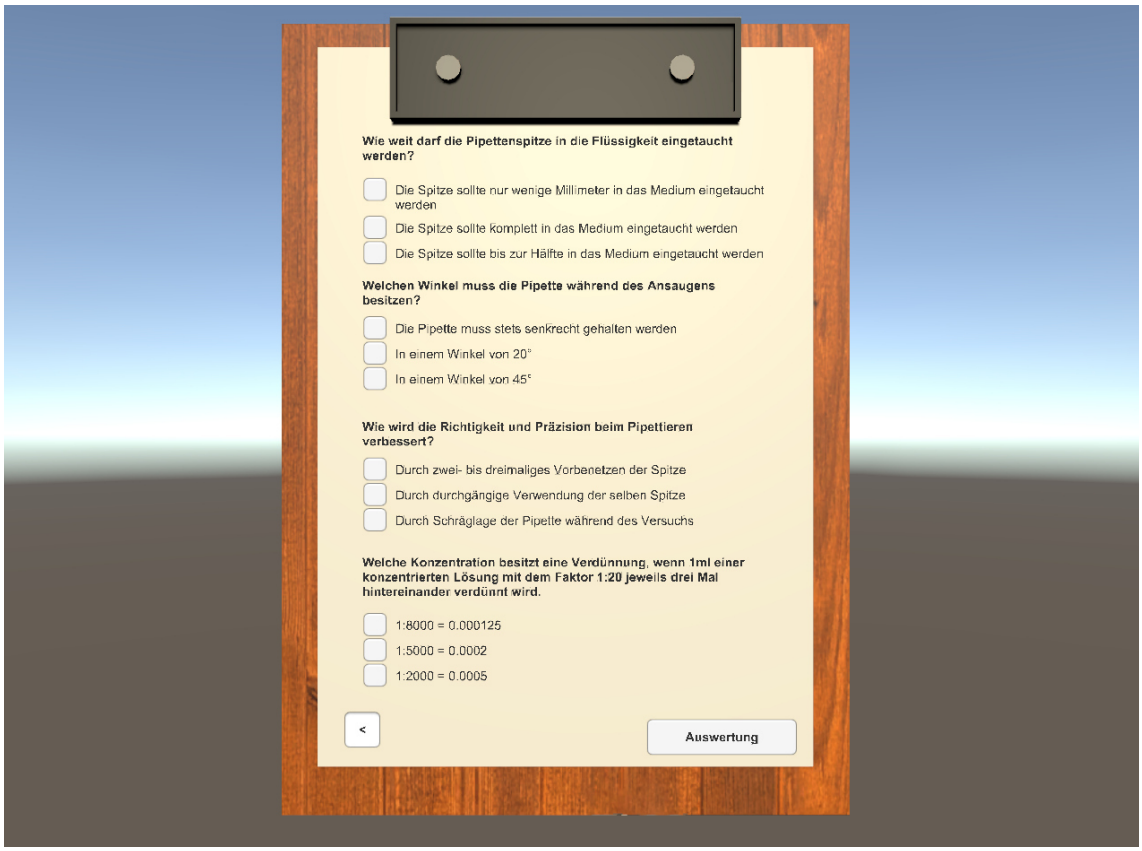

Abbildung 3.3: Frontale Ansicht der Szene Fragenkatalog

Nach Vervollständigung des *Fragenkatalogs* und dem Betätigen des *Auswertungs-Buttons* kommt es zu einer Darstellung der *Auswertung*. Diese zeigt die Anzahl der Fehler, sowie den prozentualen Anteil der richtigen Antworten an. Sollten nicht alle Fragen zu 100 Prozent richtig beantwortet sein, wird der User mit Hilfe des *Weiter-Buttons* am unteren Rand des Klemmbretts zurück zum *Hauptmenü* geführt. Hier kann dieser den Test von neuem starten. Erst nach zutreffender Beantwortung aller Aufgaben, kommt es zu einer Weiterleitung zum eigentlichen Experiment, welches im besagten virtuellen Labor stattfindet.

Den Inhalt der Fragen bilden die zusammengefassten Informationen aus der Recherche und dem dazu passenden Vorlesungsskript. Dabei bestehen die Aufgaben unter anderem aus den Kerngedanken, welche hinter dem Versuch liegen und aus weiterführenden Überlegungen auf rechnerischer Basis. Somit bieten die Aufgaben nicht nur eine Abfrage der zum Experiment zugehörigen Theorie, sondern bilden zusätzlich eine Stütze für den darauf folgenden Versuch.

3.3.3 Virtuelles Labor

Das virtuelle Labor bildet das Kernstück dieser Arbeit. Hier finden die bislang implementierten Versuche statt. Um einen sinnvollen Versuchsablauf zu gewährleisten, ist es erforderlich, dass alle integrierten Objekte sinnvoll und passend zum Experiment aufgebaut sind. Das erleichtert die Handhabung des Experimentes.

Um den Nutzer direkt mit der Fragenstellung des entsprechenden Versuches zu konfrontieren, ist diese als ein *Panel*, welches dem Blick des Users folgen soll, angelegt. Das Übersehen der Aufgabenstellung ist damit nicht möglich. Die Fragestellung ist an das Sichtfeld angepasst und findet sich nach Bestätigung auf einem Klemmbrett im Raum wieder. Dies soll gewährleisten, dass sich der Nutzer zu jedem Zeitpunkt erneut mit der Aufgabenstellung auseinander setzen kann. Das Labor ist schlicht mit den extra dafür ausgesuchten Objekten eingerichtet, damit der Nutzer nicht mit unnötigen Reizen konfrontiert wird. Um den Versuchsablauf für den User so einfach wie möglich zu gestalten, befinden sich auf der linken Seite des Raumes alle essentiellen Informationen und auf der rechten Seite der eigentliche Versuch mit den dafür relevanten Objekten.

Auf der rechten Seite ist der Versuchsaufbau vorzufinden. Hierzu zählen alle erforderlichen Objekte, wie beispielsweise die verwendbaren Spitzen, Pipetten mit ver-

Abbildung 3.4: Ansicht des virtuellen Labors inklusive der eingebauten SteamVR-Kamera

schiedenen Volumen, ein Abwurfbehälter für bereits benutzte Spitzen, Reagenzgläser sowie Schaltflächen zur Bestätigung des Versuchs, zum Neustart und um zurück zum *Hauptmenü* zu gelangen. Jene Schaltflächen besitzen im Design die für ihre Zwecke üblichen Piktogramme. Das wichtigste Versuchsobjekt ist die Pipette. Diese beinhaltet alle benötigten Animationen, wie den Spitzenabwurf, die Auf- und Abnahme von Flüssigkeiten und das Einstellen der Volumenanzeige. Da sich die Zuordnung der verschiedenen Pipetten schwierig gestaltet, befindet sich auf der linken Seite des Raumes ein Plakat mit den Volumengrößen der einzelnen Pipetten. Zudem wird auch die Visualisierung der Volumenanzeige sowie die Handhabung der Controller hier erneut aufgeführt. Die Pipette und die damit verbundene Volumenanzeige lassen sich vollständig mit den Controllern bedienen. Die Auf- und Abgabe ist mittels des *Touchpads* realisiert. Dieses wirkt sich in Kombination mit der *Grip*-Taste zudem auf die Anzeige aus. Bei positiven Werten dreht sich sowohl das Drehrad als auch die untere Anzeige. Da jedes Rad der Anzeige von 0 bis 9 nummeriert ist, ist diese pro Schritt um 36° zu drehen. Simultan dazu besitzt jedes Anzeigerad einen im Skript definierten Zähler (*Counter*), der hochgezählt wird. So dreht sich die erste Anzeige

bei einem Schritt um 36° und der *Counter* steht auf 1. Erreicht der erste *Counter* die Zahl 10 wird dieser auf 0 zurückgesetzt und die zweite Anzeige wird um 36° gedreht und der *Counter* der zweiten Anzeige steht auf 1. So wird mit allen 4 Anzeigerädern verfahren. Bei negativen Werten funktioniert die Anzeige gleichermaßen, nur zählt der *Counter* entgegengesetzt und die Räder drehen sich um -36°.

Die restlichen beweglichen Objekte können mittels des einfach bedienbaren *Trigger*-Buttons versetzt werden.

3.3.4 Auswertung

Das Design des *Auswertungspanel* ist schlicht gehalten und beinhaltet ausschließlich, die für das Ergebnis benötigten Texte, da nur dies für den Nutzer von Relevanz ist. Hierbei werden alle im Versuch begangenen Fehler systematisch aufgelistet, damit der User einen Überblick über diese erhält. Die Fehler werden innerhalb eines Skripts von Gabor Schulze definiert [Sch17]. Dieses Skript speichert die Fehler als *PlayerPrefs*. Das *Auswertungs-* Skript besitzt nun die Aufgabe, dort eingegebene Fehler auszulesen und je nach Inhalt das *Auswertungs-Panel* zu gestalten. Zudem beinhaltet das *Panel* in der unteren linken Ecke eine Schaltfläche die zurück zum *Hauptmenü* führt.

Abbildung 3.5: Ansicht des Auswertungs-Panels

3.4 Einbindung der VR

Die Einbindung der *HTC Vive* erfolgt beinahe automatisch durch das Herunterladen des entsprechenden Plugins, welches im *Unity Assetstore* kostenlos zur Verfügung steht. Nach erfolgreicher Einbindung befindet sich das erforderliche *CameraRig* im *Projekt-* Fenster unter *Assets/SteamVR/Prefabs*. Es beinhaltet sowohl das Headset als auch beide Controller. Die Standardverfolgung geschieht automatisch beim Starten der Applikation.

Die Ortung des Headsets und der Controller erfolgt automatisch durch das in dem Projekt integrierten *SteamVR*. Um dieses effizient und exakt wie möglich verwenden zu können, ist es ratsam, zunächst eine Raumvermessung mit Hilfe der *HTC Vive*-Software durchzuführen.

3.5 Zusammenfassung

Dieses Kapitel beinhaltet sowohl die Beschreibung der Entstehung der Objekte und der essentiellen *Unity*-Szenen als auch die Einbindung der virtuellen Realität in Form von einem Headset und den dazu gehörigen Controllern. Die relevanten Objekte liegen nun in ihrer Gesamtheit vor und können problemlos in die Engine eingebunden und animiert werden. Dabei sind diese zu einem späteren Zeitpunkt problemlos anpassungsfähig und erweiterbar. Die erstellten Szenen beinhalten alle bedeutsamen Elemente, welche im folgenden Schritt ihre Interaktivität und Steuerung erhalten. Dabei hat jede einzelne Szene einen Mehrwert für das Gesamtprojekt. Zudem ist es möglich, mit Hilfe der nun integrierten *SteamVR*-Plugins, jene Szenen auch im virtuellen Raum zu begutachten.

4 Implementierung

Das zuvor behandelte Kapitel befasste sich mit der Konzeptionierung der Objekte und den entstandenen Szenen. An dieser Stelle folgt entsprechend deren Implementierung in der *Unity Engine*. Dementsprechend wird die Objekterstellung, die Objekteinbindung sowie die Umsetzung der Steuerung und der Interaktionen mit den gegebenen Gegenständen erörtert.

Um die Implementierung der einzelnen Szenen zu vereinfachen, entstanden im Laufe der Arbeit *UML*-Diagramme (*Unified Modeling Language*), welche die einzelnen Phasen des Projektes auffächern und konzeptionell erläutern. Hierbei beschreibt das aufgeführte *Use-Case*-Diagramm (*Anwendungsfalldiagramm*) die einzelnen Anwendungsfälle mit den dafür erforderlichen Akteuren, Abhängigkeiten und Beziehungen. Um dieses übersichtlicher zu gestalten, sind die implementierten Szenen einzeln aufgelistet.

Somit stellt die oberste Abbildung (Abbildung 4.1) das *Hauptmenü* dar, in welchem der Nutzer verschiedene Schaltflächen zur Auswahl hat. Je nach Entscheidung führt dieses weiter zum *Fragenkatalog* (zweite Abbildung). Ein erfolgreiches Ausfüllen des *Fragenkatalogs* und die darauf folgende Bestätigung generiert automatisch eine Bewertung und eine dementsprechende Weiterleitung zum *Hauptmenü* oder zum eigentlichen *Experiment*. Das dritte Abbild zeigt die verschiedenen Handlungsmöglichkeiten des Versuchs auf. Hierbei sind die Grundinteraktionen mit den Objekten, wie beispielsweise die Manipulation der Pipette und den Schaltflächen aufgelistet sowie die damit zusammenhängenden Beziehungen und Reaktionen. Ein vollzogenes und bestätigtes *Experiment* führt schließlich zu Abbildung vier, welche den Ablauf der *Auswertung* darstellt.

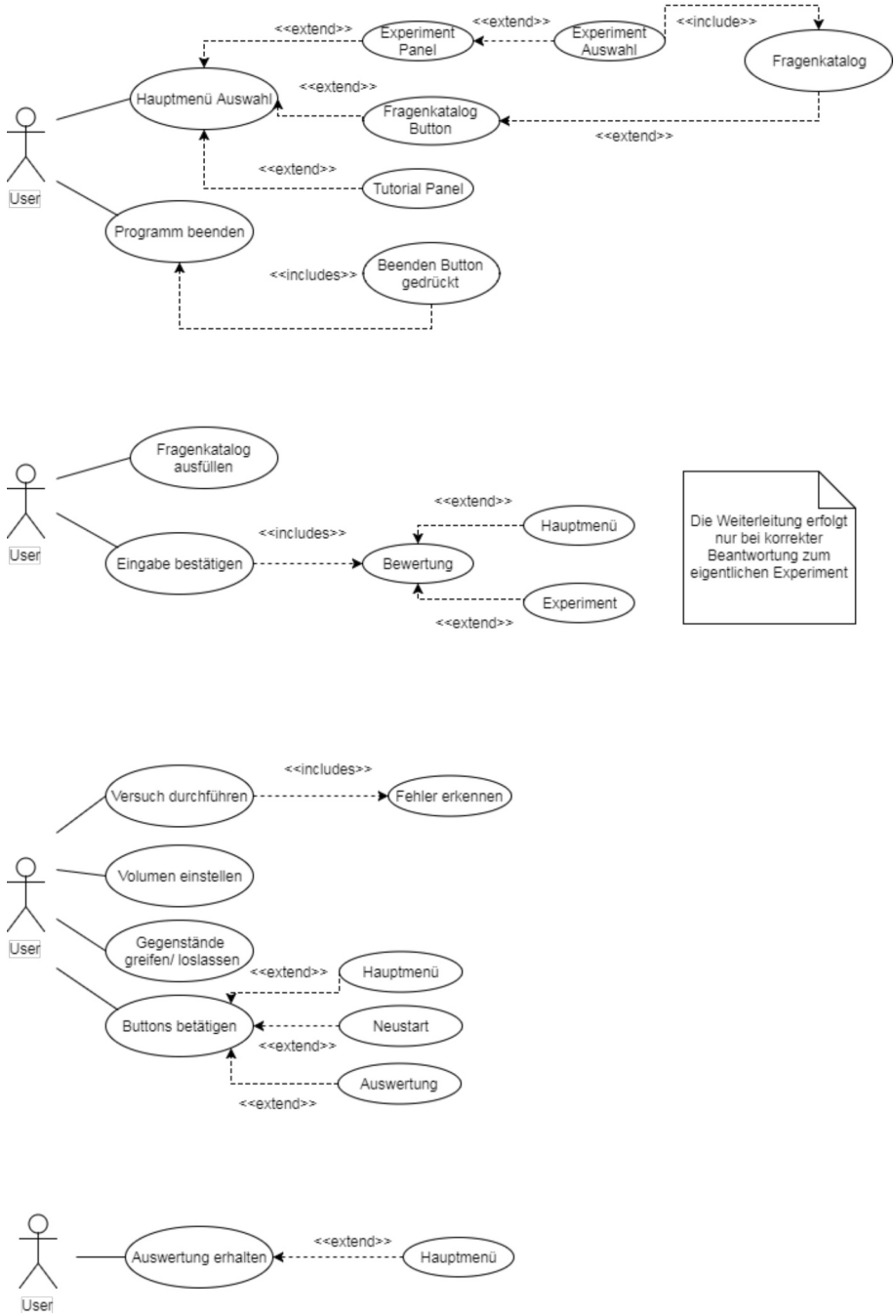

Abbildung 4.1: Use-Case-Diagramm mit allen in der Software enthaltenen Interaktionsmöglichkeiten

4.1 Vorarbeit

Die Vorarbeit zur Implementierung der Szenen in der *Unity Engine* bestand darin, wichtige Informationen, wie beispielsweise die Kompatibilität verschiedener Dateiformate, zu recherchieren. Zudem war eine erneute Einarbeitung in die Programmierung mit `C#` unerlässlich.

Zunächst musste eine neue *Unity*-Szene erstellt werden. Diese sollte dreidimensional und mit der *HTC Vive* kompatibel sein. Um Kompatibilität zu gewährleisten, muss unter *Edit/ProjektSettings/Player/OtherSettings* die Komponente *Virtual Reality Supported* verwendet und *OpenVR* hinzugefügt werden. Dadurch erkennt *Unity* die Hardware der *HTC Vive* Mit Hilfe der *Console* kann für jede erforderliche Umgebung eine neue Szene hinzugefügt werden. Die Entwicklung der VR-Umgebung ist durch das Plugin *SteamVR* erleichtert.

4.2 Objekterstellung

Um Komplikationen jeglicher Art zu vermeiden, ist es unumgänglich gewesen, die 3D-Modelle sauber hinsichtlich ihrer Struktur zu modellieren. Denn nur so ist eine akkurate Einbindung in die verwendete *Unity Engine* gewährleistet. Dies beinhaltet die Vermeidung von *Polstellen* oder sich überlappenden Polygonen. *Polstellen* beschreiben Stellen in der Modellierung von Objekten, an welchen ein Vertex (Punkt) mehr als vier Edges (Kanten) miteinander verbindet. Dies kann dazu führen, dass bei der Unterteilung des Objektes ein unsauberes Ergebnis entsteht. Zudem kann es bei der Konvertierung in andere Formate oder bei der Einführung in die Engines zu Problemen und Fehlern kommen. Sich überlappende Polygone sind zudem zu vermeiden, da es in der weiterführenden Entwicklung des Objektes zu Komplikationen bei der Texturierung kommen kann. Der *Polycount*, Anzahl der Polygone bzw. Vertices aus denen das Modell besteht, ist zudem möglichst gering zu halten, da dies sonst zu Performanz-Problemen in der Engine führt.

Modellierung

Alle entstandenen Objekte wurden aus einem einzelnen Quader bzw. einem Zylinder geformt. Hierfür werden zunächst die Höhe, Breite und Länge sowie die Anzahl der

zu Beginn gewünschten Unterteilungen bestimmt. Ein *Edit Poly* wird zu Beginn als Modifizierung auf das Modell gelegt. Mit Hilfe jener Modifizierung ist es möglich, das Modell nach Belieben zu Formen und zu verändern. Sollten weitere Rundungen von Nutzen sein, wird als weitere Modifikation ein *TurboSmooth* genutzt. Hierbei werden als Neuerung weitere Edges zwischen den schon Bestehenden eingesetzt und das Modell erhält abgerundete Kanten. Dies ist vor allem bei Objekten wie beispielsweise Bechergläsern oder Erlenmeyerkolben von Nutzen. Abschließend ist es ratsam, alle Vertices auszuwählen und im *Edit Poly* der Einstellung *Weld*, einen möglichst niedrig gewählten Wert zuzuordnen. Dies führt alle sich überlappenden Punkte zusammen und bereinigt damit das entstandene Objekt.

UV-Mapping

Das Programm *Autodesk 3ds Max* ermöglicht es, eine Modifizierung auf das Objekt zu legen, welche eine *UV-Map* erstellt. Diese ist meist akkurat genug, damit sie direkt Verwendung finden kann. Sollte dies nicht der Fall sein, besteht des Weiteren die Möglichkeit, einzelne Passagen der *UV-Map* eigens zu gestalten.

Texturierung

Die Umsetzung der Texturierung wurde in dem Programm *Blender* vollzogen, da das Programm hinsichtlich der äußerlichen Gestaltung der Modelle wesentlich übersichtlicher und leichter bedienbar ist als *Autodesk 3ds Max 2015*. Durch das Format *.obj* konnte die in *Autodesk 3ds Max* erstellte *UV-Map* ohne Komplikationen importiert und weiter verwendet werden. Durch Veränderungen in *Adobe Photoshop* und der Erstellung eigener Texturen gelang die benötigte Varianz der Darstellung des Labors. Zusätzlich ergab sich eine Anpassung der Texturen hinsichtlich des divergierten Volumens der Pipetten, der Bechergläser und der Erlenmeyerkolben.

Die Zuordnung der jeweiligen Texturen zu den angemessenen Polygonen gelang in *Blender* problemlos. Zudem konnten Materialien passend arrangiert und das finale 2D-Abbild als *Portable Network Graphics (.png)* gespeichert werden.

Export

Da die Zusammenstellung der erforderlichen Texturen in *Blender* wesentlich übersichtlicher gestaltet ist und *Autodesk 3ds Max* zudem an einigen Stellen Probleme bei dem Export von *.fbx* Dateien aufweist, stellte es sich als sinnvoll heraus, zunächst *.obj* Dateien für *Blender* zu exportieren und diese nach der Erstellung der Textur als *.fbx* erneut abzuspeichern und dann in *Unity* zu importieren.

2D-Inhalte

Die Entwicklung der grafischen Elemente wurde in *Adobe Illustrator CS6* realisiert. Die Hintergründe für die grafische Benutzeroberfläche entstanden aus einem Rechteck mit abgerundeten Ecken. Diese sind stilistisch aneinander angepasst und in allen fundamental relevanten Größen vorhanden. Der verwendete Farbton ist dabei nicht zu grell gewählt und lenkt den Nutzer dadurch nicht unnötig von seinen Aufgaben ab. Belangvolle Texte entstanden erst in der Engine, um etwaige Stilunterschiede zu vermeiden. Die entstandenen Vektorgrafiken sind verlustfrei skalierbar und lassen sich problemlos in die verwendete Engine einbinden.

Jegliche Schaltflächen und *Panels* erhielten die gleiche Art von Farbgestaltung und Design. Dies führt zu einem stimmigen, gut verständlichen Gesamtbild. Die gewählten Farbkombinationen sollten als nicht zu aufdringlich empfunden werden und sich gut miteinander kombinieren lassen. Die Schriftgröße muss für eine Darstellung mit der VR-Brille groß genug und die Schriftart deutlich sein.

4.3 Objekteinbindung

Im nächsten Schritt erfolgt die Einbettung der konzipierten 3D-Modelle und 2D-Grafiken mit den dazugehörigen Einstellungen für die Weiterverarbeitung. Durch Hereinziehen in das Projekt oder durch *Import New Assets* erhält das Projekt alle relevanten Objekte.

Um 2D-Elemente als diese zu erkennen, ist es erforderlich im *Inspektor* unter `Texture Type` `Sprite (2D and UI)` einzustellen und den Namespace `UnityEngine.UI` in den verwendeten Skripten einzubauen. 3D-Objekte werden automatisch als diese

erkannt. Hier ist es unerlässlich, regelmäßig verwendete Objekte als *Prefab* abzuspeichern. Dies wird durch das Hinzufügen des Objektes zum *Prefab*-Ordner realisiert. Texturen sind den Modellen zuzuweisen. Hierbei muss das gewünschte Modell in der Hierarchie ausgewählt sein. Jedes Modell enthält ein Material, in welchem unter `Main Maps/Albedo` die zugehörige Textur zuzuweisen ist.

4.4 Steuerung

Im Nachfolgenden wird auf die Implementierung des VR-Systems und der Interaktionen eingegangen. Hierbei ist zu beachten, dass die zur Verfügung stehenden Skripte auf die in der Applikation interagierenden Objekte angepasst sind. Somit sind spätere Anpassungen und zukünftige Erweiterungen des Projekts möglich.

4.4.1 Virtual Reality-System

Das Virtual Reality-System wird dank dem *SteamVR*-Plugin automatisch von *Unity* erkannt. Um die Controller effizient nutzen und diese in Interaktionen mit anderen Objekten einbinden zu können, ist es zunächst erforderlich, beide Controller sowohl mit einem *Rigidbody* als auch mit einem *BoxCollider* zu versehen. Die Komponente *Rigidbody* mit dem eingeschalteten Parameter `Is Kinematic` ermöglicht es, dass das Objekt hinsichtlich der Berechnung weiterhin von der Physik-Engine berücksichtigt wird, jedoch keine direkte Beeinflussung mehr durch diese, sondern nur noch durch die `Transform`-Komponente, erfährt [Sei15, S.210]. Der *BoxCollider* erhält den Parameter `Is Trigger`, damit beide Controller durch den *Collider* als Signalgeber dienen, falls sie mit einem interagierenden Objekt kollidieren. Als vorangehendes Beispiel für die Aufnahme und die Abgabe von Objekten mit Hilfe der Controller, diente das *HTC Vive*-Tutorial von Eric Van de Kerckhove [VdK16].

Zunächst wird durch das auf den Controllern liegende Skript ausgelesen, ob durch die Schaltflächen eine Eingabe erfolgt. Sollte der *Trigger* des kollidierenden Controllers gedrückt sein und es zu einer Kollision mit einem anderen Objekt kommen, auf welchem ein *Rigidbody* liegt, kann dieses Objekt vom Controller aufgehoben werden. Dies passiert durch die Methode `AddFixedJoint()`, welche die Bewegung des Objektes auf die des Controllers beschränkt [Tec17a]. Hierbei ist zusätzlich die

Kraft, die beide Objekte zusammenhält, einstellbar. Wird der *Trigger* losgelassen, löst sich die Kraft des Bandes zwischen dem Objekt und dem Controller.

Zusätzlich zu den beweglichen Objekten, gibt es Elemente, die sich je nach eingelesenem Index verändern lassen. Hierbei läuft die Interaktion mittels einer Schnittstelle (*Interface*) ab. Inwieweit die Controller-Eingabe die entsprechenden Elemente beeinflusst, wird in Kapitel 4.3.2 beschrieben.

4.4.2 Interaktionen

Jedes Objekt, mit dem der Nutzer interagieren kann, besitzt sein eigenes angepasstes Skript. Unterscheidungen gibt es dabei zwischen den Elementen der grafischen Benutzeroberfläche und den benutzbaren Objekten während des Versuchs.

Die wohl wichtigsten Methoden sind die `Trigger-Methoden`. Unterschiede sind zwischen `OnTriggerEnter`, `OnTriggerStay` und `OnTriggerExit` zu verzeichnen. Sie werden jeweils aufgerufen, wenn die Kollision gerade angefangen hat, über einen längeren Zeitpunkt stattfindet oder sich die Kollision löst. Alle drei Methoden werden von den auf den Controllern befindlichen *Collidern* ausgelöst.

Die Interaktionen mit den Elementen der grafischen Benutzeroberfläche und den im Versuch integrierten Objekten funktioniert über eine sogenannte Schnittstelle (*Interface*), von welchem alle steuerbaren Objekte erben. Die *Controller-Klasse* prüft demnach zuerst, ob das kollidierende Objekt von diesem Skript erbt. Sollte dies der Fall sein, können Änderungen mittels der *Controller-Buttons* am Objekt bzw. in der Szene vorgenommen werden.

GUI-Elemente

Die Handhabung des *Hauptmenüs* wird im Nachfolgenden durch ein dafür angelegtes Klassendiagramm beschrieben. Jede verfügbare Schaltfläche erhält ein eigenes Skript, welches mit der Schnittstelle gekoppelt ist.

Die Elemente der grafischen Benutzeroberfläche, welche in der Anwendung durch Schaltflächen vertreten sind, reagieren auf eine bleibende Berührung mit dem Controller immer auf dieselbe Art und Weise. Hierbei verfärbt sich durch `GetComponent<Image>().color = Color.black;` die Hintergrundfarbe der berührten Schalt-

fläche schwarz. Sobald der Controller und die jeweilige Schaltfläche nicht mehr in Berührung stehen, erscheint diese wieder in der herkömmlichen weißen Hintergrundfarbe. Dieses Feedback hilft dem Nutzer sich in der Applikation zurecht zu finden. Sollte der *Trigger* während einer Kollision mit einer Schaltfläche gedrückt werden, so wird je nach Schaltfläche eine andere Methode aufgerufen. Der *Zurück-Button* führt auf jedem *Panel* zurück zum *Hauptmenü*. Durch den *Tutorial* und den *Experiment-Button* wird jeweils ein entsprechendes *Panel* eingeblendet und das *Hauptmenü* ausgeblendet. Der *Beenden-Button* schließt durch `Application.Quit();` die Applikation.

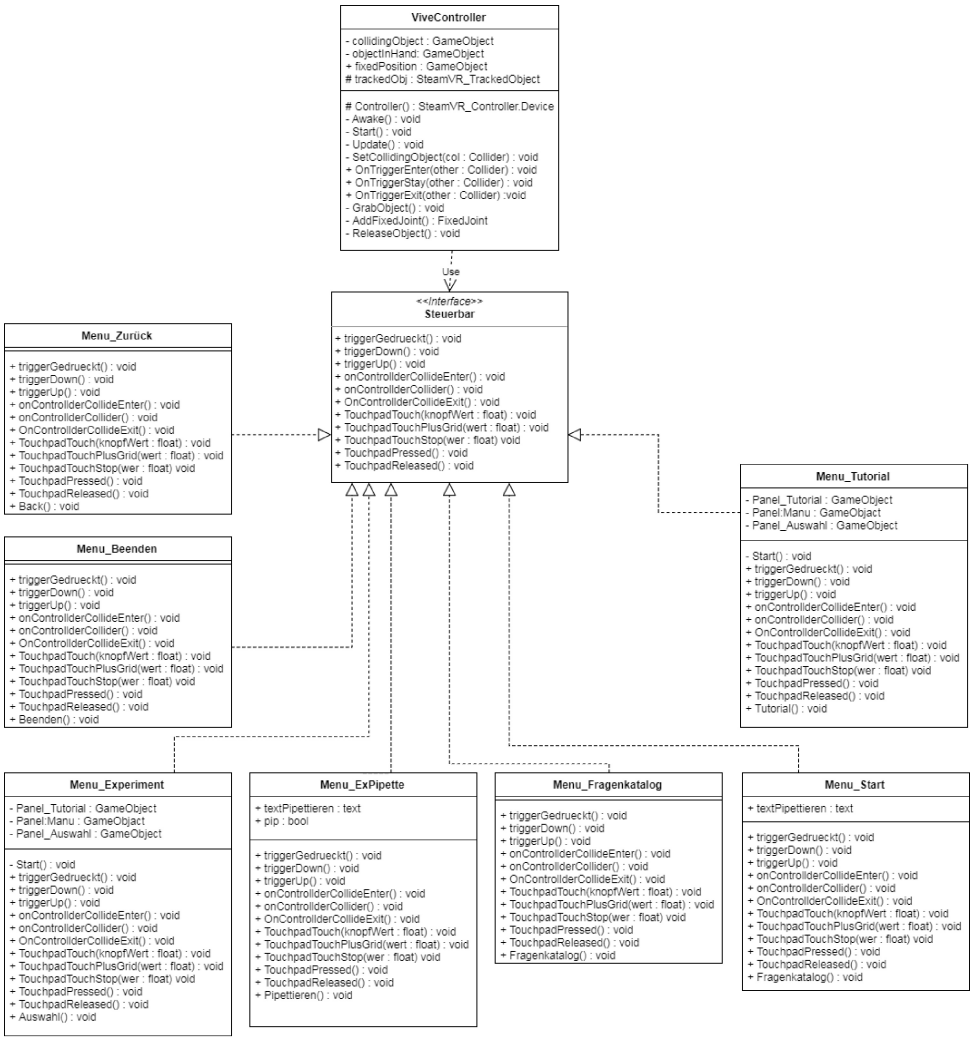

Abbildung 4.2: Klassendiagramm Menü

Die Szene, welche den *Fragenkatalog* enthält wird entweder durch den *Fragenkatalog-Button* geladen oder durch eine Versuchsauswahl im *Experiment-Panel*. Durch die Auswahl eines Versuchs wird eine *Boolsche Variable* im *Menu_ExPipette*-Skript auf den Wert `true` gesetzt und es erscheint ein dazu gehöriger Text in der Anwendung. Das *Menu_Start*-Skript prüft nun, ob jene Variable auf `true` gesetzt ist. Ist dies der Fall, lädt der *Start-Button* den Fragenkatalog.

Fragenkatalog

Das implementierte *Fragenkatalog-Panel* enthält als Kind-Elemente des *Canvas* so genannte *Toggles*. Diese ermöglichen es, mehrere Antwortmöglichkeiten in Form von *Checkboxen* zu jeder Frage hinzufügen zu können. Sobald einer der *Toggles* aktiviert wird, setzt *Unity* automatisch einen Haken in das Kästchen. Auch in dieser Szene sind alle Klassen mit der Schnittstelle verbunden. Das entwickelte *Fragenkatalog_Toggle*-Skript regelt die Antwortmöglichkeit dahingehend, dass nur eine Antwort pro Frage ausgewählt werden kann.

Dies wurde erneut durch eine *Boolesche Variable* geregelt. Die Schaltfläche mit der Aufschrift *Auswertung* liest nach betätigen aus, welche *Checkboxen* aktiviert wurden. Die Auswertung bezieht sich sowohl auf die richtigen als auch auf die falschen Antworten. Das Ergebnis wird in einem neuen *Panel* mit Hilfe von Textfeldern angezeigt. Die Anzeige beschränkt sich auf die Anzahl der korrekten und falschen Antworten, auf die Prozentzahl der korrekt beantworteten Fragen in Relation zu ihrer Gesamtanzahl und einen Hinweis zur Weiterleitung. Der *Weiter-Button* führt den Nutzer entweder zurück zum *Hauptmenü* oder weiter zum Versuch. Welche Weiterleitung verwendet wird, entscheidet das *Fragenkatalog_Weiter*-Skript, welches die Antworten auswertet. Erst wenn alle Fragen richtig beantwortet und damit 100 Prozent erreicht sind, erhält der Nutzer Zugriff auf den Versuch.

Die Auswahl wird durch die Controller durchgeführt, welche durch das *SteamVR*-Skript automatisch implementiert sind. Zusätzlich wird der Controller durch einen Stift ergänzt, der die Auswahl des gewünschten *Toggles* vereinfacht. Dieser ist am rechten Controller fixiert und lässt sich somit simultan zu diesem bewegen.

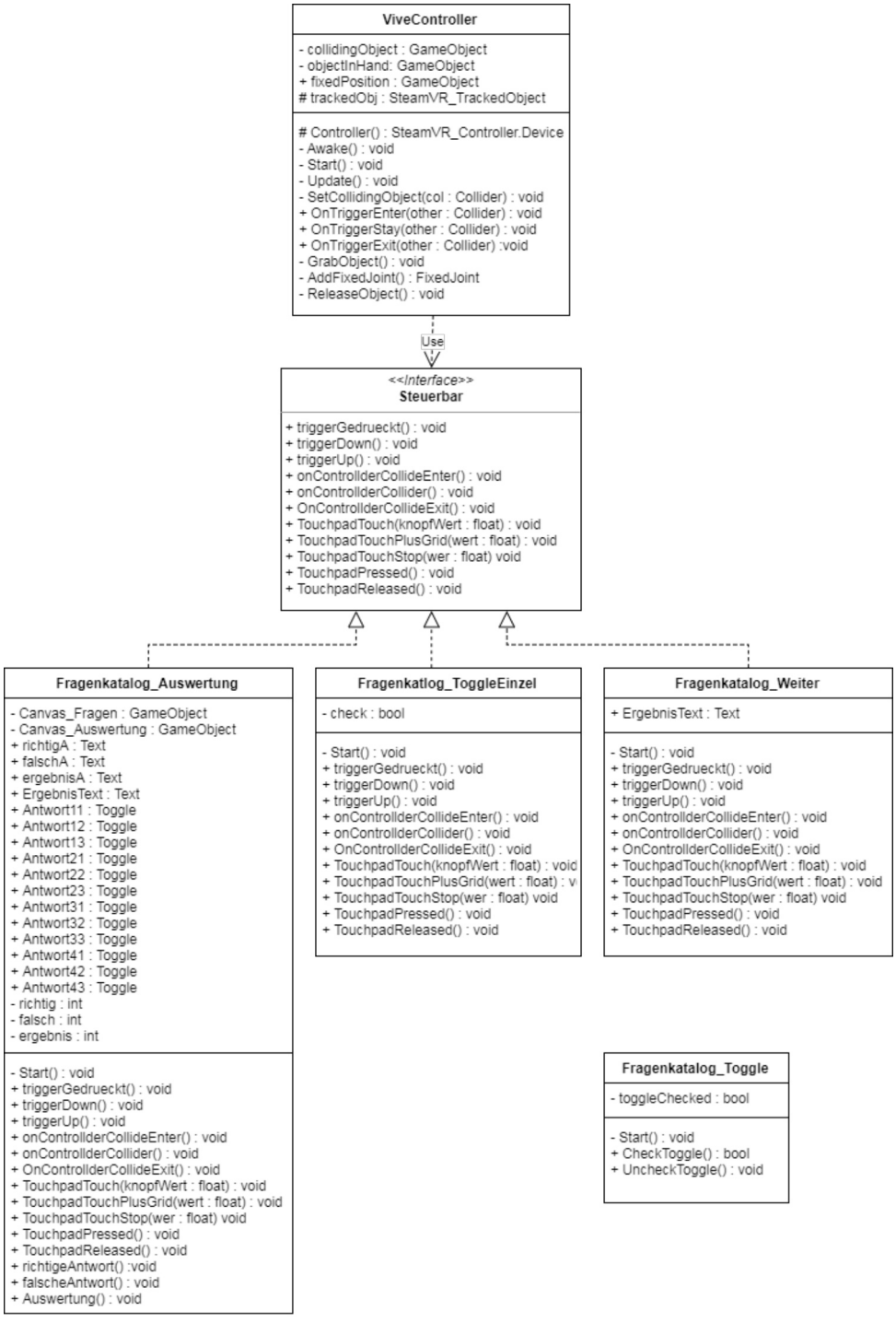

Abbildung 4.3: Klassendiagramm Fragenkatalog

Versuch

Sobald der User den *Fragenkatalog* korrekt ausgefüllt und den *Weiter-Button* bestätigt hat, lädt das Skript die eigentliche Versuchs-Szene. Der Abschnitt in dem sich der Nutzer bewegen kann, ist so ausgelegt, dass dieser keine großen Passagen zu bewältigen hat, um den Versuch absolvieren zu können. Zunächst wird dem Nutzer die Aufgabenstellung in Form eines *Panels* angezeigt. Dieses ist am Headset fixiert, d.h. es folgt den Kopfbewegungen. Um das *Panel* zu schließen, ist ein einfaches Drücken des *Triggers* erforderlich, welches durch `Destroy(gameObject);` das `Panel`-Objekt, als in der Szene enthaltenes Objekt, zerstört.

Alle Objekte, die zur Interaktion zur Verfügung stehen, enthalten sowohl ein *Rigidbody* als auch einen *BoxCollider* und sind ebenso mit der Schnittstelle verbunden. Auf dem Versuchstisch befinden sich auf der rechten Seite drei 3D-Modelle, die als Schaltfläche fungieren. Hierbei handelt es sich um einen *Neustart-*, einen *Menü-* und einen *Auswertungs-Button.* Sobald die *Collider* der Schaltflächen und die des Controllers in Berührung kommen und der *Trigger* gedrückt wird, lädt die aufgerufene Methode die jeweilige Szene. Der *Neustart-Button* lädt dieselbe Szene neu, der *Menü-Button* führt zum Hauptmenü und der *Auswertungs-Button* zu einer detaillierten Versuchsauswertung.

Ein besonderes Objekt im Versuch ist die Pipette. Sie besitzt alle relevanten Funktionen und die dazu gehörigen Animationen, welche einen Nutzen für den Ablauf einer Flüssigkeitsaufnahme und -abgabe mit sich bringt. Das Skript bedient sich, genau wie die anderen Objekte, der Schnittstellen-Methoden und Variablen. Sobald der Controller mit der Pipette kollidiert, kann mit dieser der Versuch vollzogen werden. Die Animationen beziehen sich auf die oberen Bedienknöpfe und dem Drehrad, inklusive der Volumenanzeige. Der obere Bedienknopf, welcher für die Aufnahme und Abgabe der Flüssigkeit zuständig ist, lässt sich mit Hilfe des integrierten *Touchpads* bedienen. Hierbei wird der Y-Achsen-Wert des *Touchpads* zunächst ausgelesen. Um eine Weiterverarbeitung zu vereinfachen, rundet sich die Fließkommazahl durch `Mathf.Round` auf die nächst gelegene, ganze Zahl auf. Die Animation lässt sich einfach gestalten, wenn der oberste Punkt des *Touchpads* auf null gesetzt ist. Die lokale Position des Knopfes verändert sich nun simultan zum eingelesenen Y-Achsen Wert des Touchpads. Das Drehrad rotiert nach Betätigen des *Grip-Buttons* und je nach eingegebenem Y-Wert des *Touchpads* um seine eigene Y-Achse, parallel zu den

Anzeigen der Pipette. Der Abwurfknopf wird durch Drücken des *Touchpads* bedient, welcher diesen durch Druck auf der X-Achse rotiert.

Zusätzlich besteht mit Hilfe der Pipette die Möglichkeit, das Flüssigkeitsvolumen einzustellen. Um dies zu realisieren wird erneut das *Touchpad* in Kombination mit dem *Grip-Button* verwendet. Das Drücken des *Grip-Buttons* ermöglicht es sowohl positive als auch negative Y-Achsen-Werte des *Touchpads* auszulesen, mit welchen sich die Anzeige steuern lässt. In diesem System ergaben sich zwei Ausnahmen. Der Sprung von 9999 auf 0000 und der Sprung von 0000 auf 9999 mussten einzeln programmiert werden, da es sich hierbei um das Maximum und Minimum der Anzeige handelt. Hierbei erhalten die Zähler automatisch den zur Anzeige passenden Wert.

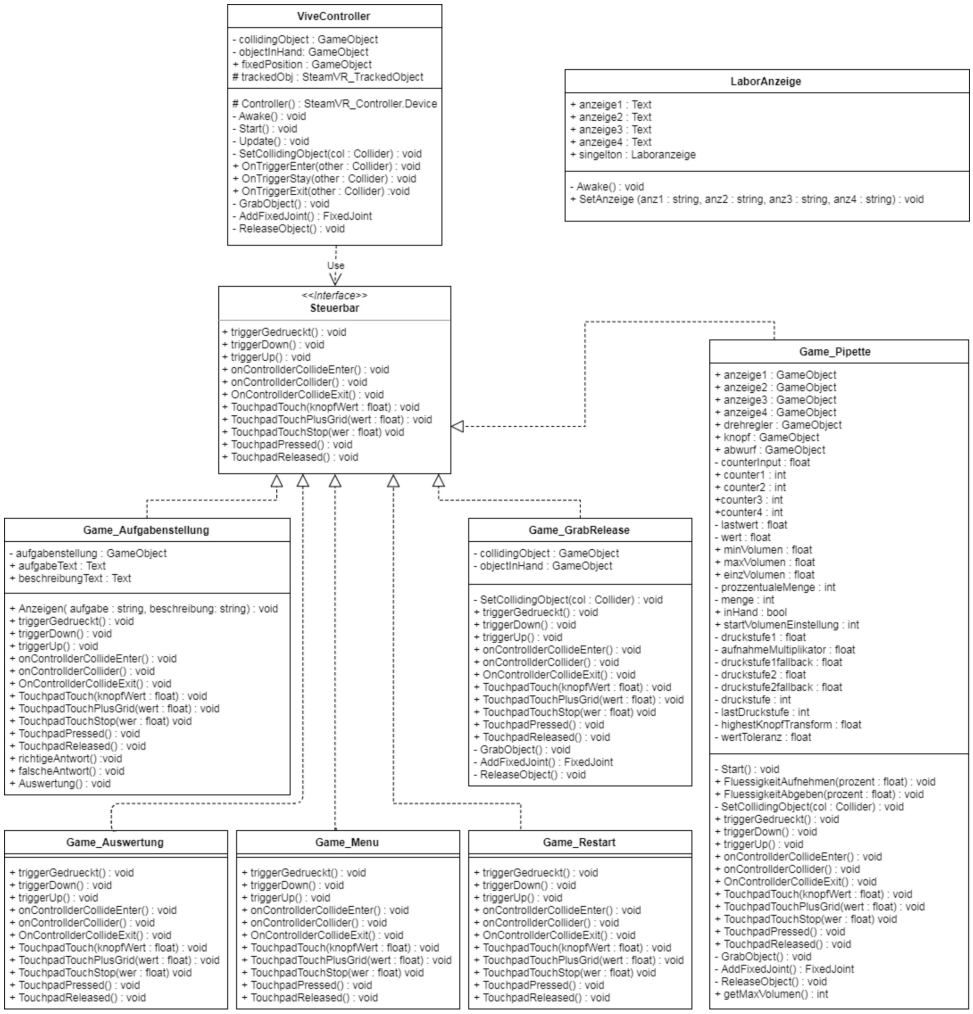

Abbildung 4.4: Klassendiagramm Game

4.5 Auswertung

Die während des Versuchs entstandenen Fehler werden von einem `C#`-Skript von Gabor Schulze mit Hilfe von *PlayerPrefs*-Methoden gespeichert [Sch17].

In der Methode `PlayerPrefs.SetString(ID, wert);` gibt der vordere Parameter den *Key* bzw. die *ID* an und der hintere den Wert dieser ID. Mit Hilfe von `PlayerPrefs.GetString()` können die Parameter im *Auswertungs*-Skript ausgelesen werden. [Sei15, S.93]

Das Skript analysiert dabei zunächst, ob eine `PlayerPrefs` mit der entsprechenden ID existiert. Sollte dies der Fall sein, wird der für diese ID festgelegte Wert ausgegeben. Im umgekehrten Fall wird eine vorgegebene Meldung ausgegeben. Je nach vorgefundenem Inhalt ändert sich der Text des *Auswertungs-Panel* und zeigt die Fehler an.

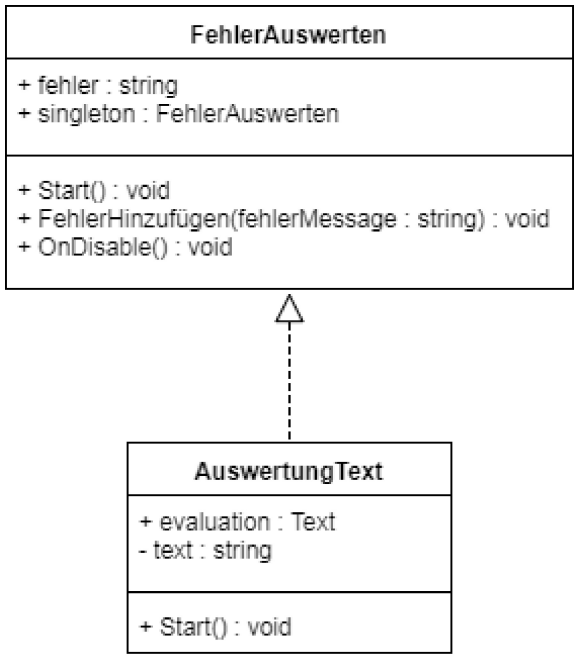

Abbildung 4.5: Klassendiagramm Auswertung

5 Evaluation

Das nun folgende Kapitel setzt sich kritisch mit der entstandenen Software auseinander. Dabei wird der wissenschaftliche Nutzen des Programms eruiert sowie positive und negative Aspekte hinsichtlich der Benutzerfreundlichkeit und der sich daraus ergebenden Lernhilfe erläutert. Zudem sind während der Implementierung Probleme entstanden, welche in diesem Kapitel eine Auflistung finden.

5.1 Aufbau des Experiments

Der Aufbau der virtuellen Testszene orientiert sich stark am realen Experiment. Unterschiede bezüglich der Handhabung der Objekte sind durch die virtuelle Umgebung und der dazu gehörigen Hardware beinahe selbstverständlich. Im Allgemeinen bieten die verwendeten Programme gute Gestaltungs- und Anwendungsmöglichkeiten, vor allem da *Unity* eine enorme Variation an Formaten unterstützt. Zudem ist Feedback für den Nutzer in jeglicher Hinsicht gut einzubinden. Alle essentiellen Objekte sind vorhanden und der Nutzer kann mit ihnen interagieren. Durch eine strukturierte Implementierung können sowohl alle Skripte überarbeitet sowie neue Skripte und Objekte ergänzt werden. Eine komplette Neugestaltung des Labors ist möglich und für spätere neu implementierte Experimente relevant. Alle für den Versuch essentiellen Objekte sind in Reichweite des Nutzers aufzufinden. Dahingehend agiert der Nutzer nur in einem kleinen Umfeld.

Die im Folgenden aufgeführte Tabelle gibt einen Einblick in die Systeminformationen, unter welchen die Arbeit realisiert wurde.

Systeminformationen	
Betriebssystem	Windows 10 Enterprise 2016 LTSB 64-Bit-Version (10.0, Build 14393)
Prozessor	Intel(R) Core(TM) i7-6700 CPU @ 3.40GHz (8 CPUs), 3.4GHz
Speicher	32768MB RAM
DirectX-Version	DirectX 12
Grafikkarte	NVIDIA GeForce GTX 1080
Speichergröße	8 GB GDDR5X
Anzeigemodus	1920×1200 (32 bit) (59 Hz)

Tabelle 5.1: Systeminformationen

5.1.1 Versuchsablauf

Um einen genauen Ablauf des Versuches vor Augen zu haben, war es zunächst erforderlich, die Aufgabenstellung sinnvoll in das Projekt einzubinden. Durch ein an das Headset gebundenes *Panel* kann diese nicht unbemerkt bleiben. Des Weiteren befinden sich die Controller-Handhabung und das jeweilige Pipetten-Volumen als virtuelles Poster an der Wand der Nutzerumgebung. Da es bei Verdünnungsreihen üblich ist, kleine Mengen an Flüssigkeit zu pipettieren, dies allerdings nur schwer kompatibel mit den gegebenen Umständen ist, musste eine Kompromisslösung gefunden werden. So bezieht sich der Versuch auf Flüssigkeitsvolumen von 1-10 mL. Die Flüssigkeiten werden nicht, wie üblich, in Reaktionsgefäße pipettiert, sondern es wird die Verwendung von Reagenzgläsern bevorzugt, da diese hinsichtlich der Hardware leichter zu handhaben sind. Nach Beendigung des Versuchs ist eine Schaltfläche zu betätigen, die das ausgeführte Experiment auswertet. Hierbei sind auf einzelne Fehler zu achten, welche in Kapitel 2.1.2 aufgeführt sind.

5.1.2 Versuchsauswertung

Die in der Auswertung gesammelten Fehler beziehen sich auf eine Auswahl der in Kapitel 3.1 aufgelisteten Fehlerquellen. Diese hat Gabor Schulze implementiert und erweitert [Sch17]. Derzeit ist noch keine exakte Analyse der Fehler möglich. So sind Fehlerquellen, wie die Lage der Pipette oder das Überdrehen der Anzeige, noch nicht

abgeschlossen simulierbar. Die ermittelten Fehler werden in der Testszene mit Hilfe der Methode `PlayerPrefs()` ausgelesen und in Form von Textfeldern auf einem `Panel` ausgegeben. Dies ist die bislang simpelste und anschaulichste Methode um den Nutzer die begangenen Fehler vor Augen aufzuzeigen.

5.2 Identifizierte Probleme

Während der Implementierung sowie der Testphasen der Steuerung und der Interaktionen sind einige Probleme aufgetreten. Manche Probleme sind weitestgehend gelöst oder umgangen, bei einigen war dies nicht bzw. nur unzureichend möglich. Die dadurch gewonnen Erkenntnisse sind in diesem Kapitel aufgelistet und kritisch bewertet.

5.2.1 Objekte

Die Entwicklung der Objekte brachte keine größeren Schwierigkeiten mit sich, da die Modellierung meist mit Hilfe von Referenz-Objekten stattfand. Hinsichtlich des Exports war jedoch darauf zu achten, dass kein Teilobjekt der Modelle nach dem Export verschoben oder gespiegelt ist. Dies kam vor allem beim Exportieren in das Format *.fbx* häufiger vor. Des Weiteren gab es in einigen Fällen spätere Abänderungen bezüglich des *Pivot-Punktes*. Dieser zeigt an, an welchem Punkt im Raum (Nullpunkt) das Objekt verschoben, gedreht und skaliert werden kann. Bei den zu den Pipetten gehörigen Spitzen ist es unumgänglich, dass dieser am oberen Rand im Mittelpunkt der Spitze liegt. Nur so ist gewährleistet, dass diese ohne weitere Probleme durch Berührung am unteren Teil der Pipette haften.

5.2.2 Controller

Die Ortung der Controller funktioniert mit Hilfe der Basisstationen meist zufriedenstellend. Sollte dies doch nicht korrekt verlaufen, ist eine neue Raumvermessung zu vollziehen. Die Aufnahme von Objekten und die Interaktion mit diesen und der grafischen Benutzeroberfläche konnte schnell und einfach realisiert werden. Hierbei kann der Controller jedoch eine menschliche Hand schlecht darstellen. So gelang

die Interaktion mit Hilfe der Schaltflächen zwar tadellos, trug allerdings nur wenig zur eigentlichen Immersion bei. Die implementierte Pipette, die das Kernstück des Experiments bildet, bedient sich einer weitestgehend sinnvollen Handhabung. Jedoch ist eine genaue Bedienung mittels den Controllern nicht möglich. So können genaue Bewegungsabläufe, die in der realen Versuchsreihe von großer Bedeutung sind, nicht ausreichend simuliert werden. Zudem erfordert die korrekte Handhabung der Controller etwas Übung.

5.2.3 Format

Auf ein gut erkennbares Format wurde stets geachtet. Jedoch ist dies in einigen Fällen nicht bzw. nur schwer möglich. So sind kleinere Details, wie beispielsweise die Volumenanzeige bzw. die Anzeigeräder an der Pipette, mit Hilfe der *HTC Vive* nur schlecht lesbar. Diese ungenügende Lesbarkeit hatte zur Folge, dass jene Objekte in anderer Form zusätzlich in das Projekt eingebaut wurden. So findet sich eine Darstellung der zu den Pipetten zugehörigen Volumina zusätzlich als Poster im Raum. Auch die Anzeigeräder der Volumenanzeige erhielten ein genaueres Auftreten in Form von einem Zähler an der Wand des Labors. Zudem kam es zu einer Anpassung der Schriftgröße der vorangehenden Fragen an die Sicht des Nutzers. Jedoch wurde damit der Umfang der Fragen auf dem *Panel* auf eine vergleichsweise geringe Anzahl begrenzt. Auch die Länge der Fragen und Antworten mussten auf das Notwendigste gekürzt werden.

5.2.4 Performanz

Um eine flüssige Wiedergabe der Anwendung zu gewährleisten, ist die Applikation in mehrere Szenen unterteilt, wobei die eigentliche Versuchsszene am aufwendigsten gestaltet ist. Auffallend in diesem Punkt ist jedoch, dass im Labor größere zusammengehörige Objekte nicht gleichzeitig bewegt werden können. So ist es nicht möglich, beispielsweise mit der Spitzenbox mit allen enthalten Spitzen zu interagieren, da die Bewegung nicht mehr flüssig dargestellt wird, sondern nur stockend abläuft. Das Bewegen der einzelnen Spitzen stellte allerdings kein Problem dar. Somit sind die Interaktionen darauf beschränkt, dass einzelne oder nur kleine Gruppen in der Szene beweglich sind.

5.2.5 Allgemeine Probleme

Die allgemeinen Probleme beziehen sich zum einen auf die Handhabung und zum anderen auf die Benutzerfreundlichkeit des Projektes. Bei der Bedienung des *Hauptmenüs* kommt es beispielsweise vor, dass durch das schnelle *Panelwechsel*, Schaltflächen aus Versehen gedrückt werden oder der Controller mehrere Schaltflächen auf verschiedenen *Panels* gleichzeitig bestätigt, wenn diese übereinander liegen. Dies ist beispielsweise bei den *Hauptmenü-Panel* und der danach aufgerufenen *Experiment-Panel* der Fall. Hierbei ist eine Korrektur dieses Fehlers nur teilweise möglich gewesen. Des Weiteren bietet das Menü keine gute Grundlage für eine gelungene Immersion eines realen Umfelds, da die Interaktion immer in einem künstlich gestalteten Wirkungsbereich stattfindet. Eine weitere Auffälligkeit liegt in der virtuellen Realität schlechthin. Um diese angemessen verwenden zu können, ist ein grundlegendes Verständnis für Technik erforderlich. Zudem muss das erforderliche Equipment vor Ort sein. Ein weiterer Malus ist die Tatsache, dass die Applikation bislang nur für einzelne Nutzer zugänglich ist. Somit ist kein Austausch mit weiteren Personen in der Applikation möglich.

Zudem können bei manchen Menschen durch das Verwenden der virtuellen Applikation Schwindel oder ähnliche Begleiterscheinungen wie Kopfschmerzen aufkommen. Diese Störung trägt die Bezeichnung *Simulator Sickness* und führt bei betroffenen Nutzern zu Beeinträchtigungen, die sowohl gesundheitliche als auch funktionale Areale betreffen. Diese Art von Begleiterscheinung kann sich auf die spätere Verhaltensweise in der realen Umgebung ungünstig auswirken. [JH09, S.13]

5.3 Fazit

Zusammenfassend ist festzuhalten, dass die Applikation alle wünschenswerten Eigenschaften besitzt und komplett ausgestaltet werden konnte. Das Feedback für den Nutzer ist effizient und sinnvoll eingebaut und die verwendeten Objekte erweiterbar und dynamisch gestaltet. So können nicht nur neue Modelle und Grafiken, sondern auch dazugehörige Skripte, problemlos bearbeitet und eingebunden werden. Jedoch ist unter anderem zu bemerken, dass in seltenen Fällen Verbesserungen angebracht sind. So ist die Controller-Handhabung bzw. die gegebene Controller-Struktur nicht optimal für eine Verdünnungsreihe mittels Pipettieren gestaltet, da die essentielle

Feinmotorik nicht trainiert werden kann. Des Weiteren sind kleinere Objekte nur schwer zu erkennen, was die Arbeit mit Mikroliter-Pipetten erheblich erschwert. Hinzu kommen kleinere Performanz-Probleme, die zwar umgangen, aber nicht gelöst sind sowie die Tatsache, dass das Projekt nicht auf mehrere Personen ausgelegt ist.

6 Zusammenfassung und Ausblick

Das letzte Kapitel dient dazu, die entstandene Arbeit sinnvoll zusammen zu fassen und einen Ausblick über die bereitliegenden Erweiterungsmöglichkeiten des Projektes zu geben. Hierbei wird erneut auf die Machart des virtuellen Labors eingegangen und positive und negative Aspekte beschrieben.

Im Allgemeinen ist die Erarbeitung, der als Titel dieser Arbeit formulierten Aufgabe, in allen Punkten erfolgt. So legen die in Kapitel 1 formulierte Motivation und die darauf folgende Aufgabenstellung den Grundstein für die Arbeit. Die in Kapitel 2 dargelegten Grundlagen führen den Leser geschickt in die Thematik ein und zeigen die Voraussetzungen für das Entstehen der Arbeit auf. Darauf aufbauend schließt sich das Kapitel der Konzeption an, in welchem die Autorin die genaue Darstellung der Applikation darlegt. Die Implementierung des Systems in Kapitel 4 setzt sich detailliert mit dem genauen Aufbau der *Unity Szenen* und Skripten auseinander. Kapitel 5 rundet mit der Evaluation die Arbeit ab. Hierbei wird erneut auf die entstandenen Aspekte eingegangen und deren Nutzen, vor allem in Bezug auf die Handhabung und der Benutzerfreundlichkeit, eruiert.

Das Hauptmerkmal der Arbeit ist dabei, ein Konzept für eine Applikation zu erschaffen, welche eine Verdünnungsreihe mittels Pipettieren in Kooperation mit der *HTC Vive*-Hardware sinnvoll realisiert und diese hinsichtlich User Interface und Steuerung umsetzt. Diese soll in Zukunft beispielsweise hinsichtlich weiterer Experimente problemlos erweiterbar sein. Ob diese Eigenschaft gegeben ist, wird im Nachfolgenden kurz erörtert.

6.1 Anpassungs- und Verbesserungsmöglichkeiten

Hinsichtlich der in Kapitel 5 ausgeführten Evaluation gibt es, nach kritischer Betrachtung, Anpassungs- und Verbesserungsmöglichkeiten. Diese betreffen sowohl die

Steuerung, als auch die Performanz. Die Steuerung kann beispielsweise durch eine veränderte Hardware-Komponente verbessert werden. So ist es möglich, statt den herkömmlichen Controllern, geortete Handschuhe zu verwenden. Hierbei erkennt die Software nicht nur die Position der Hand, sondern die Aktionen der einzelnen Finger sind mit inbegriffen. Dadurch kann der bislang fehlende Aspekt der Feinmotorik in den Versuch mit eingebracht und die Simulation des Pipettierens wesentlich realistischer gestaltet werden. Aufgetretene Performanz-Probleme sind wohl auch zukünftig nur schwer zu beheben, da zusammengehörige Objekte einen zu hohen *Polycount* haben, als dass sie als Gruppe verwendet werden können. Der *Polycount* der Modelle wurde bereits so gering wie möglich gehalten, daher kann dieser, ohne kompletten Detailverlust, nicht weiter angepasst werden. Auch die Anpassung an kleinere Formate, bezüglich kleinerer Schrift und Details, ist eine komplizierte Thematik, da dies mit der Auflösung der Hardware-Komponenten zusammenhängt.

In der Zukunft wäre es sinnvoll die Software dahingehend zu erweitern, dass mehrere Nutzer miteinander interagieren und den Versuch miteinander durchführen können. Zudem ist es praktikabel, weitere Versuche sowie Modelle mit entsprechenden Funktionalitäten in das bislang bestehende Programm einzubinden. Des Weiteren kann der Fragenkatalog modifiziert werden. So könnten beliebig viele Fragen, in zufälliger Reihenfolge, mit Hilfe einer Textdatei oder XML eingebunden werden. Ein genaueres Feedback während des Versuchs, in Form von Sound-Einspielungen oder aufblinkenden Fehlersymbolen, könnten die Simulation interessanter gestalten. Zudem wäre es sinnvoll, die begangenen Fehler abzuspeichern, um bei einer erneuten Durchführung einen Vergleich der Ergebnisse zu erhalten.

6.2 Fazit

Die Applikation stützt sich auf die Grundideen des Designkonzepts, welche sich während der Implementierung als sinnvoll herausstellten. Zudem ist die Software mit den gewünschten Funktionen versehen, welche mit der Erweiterung, durch die Arbeit von Gabor Schulze, kompatibel sind [Sch17]. Des Weiteren gibt es ausgiebige Erweiterungsmöglichkeiten und Abschnitte, die während dem Ausbau der Applikation verbessert werden können. Dazu zählen zum einen weitere Objekte und zum anderen damit zusammenhängende Skripte, die den Funktionsumfang ausbauen. Alle

erfolgten Schritte sind in der Ausarbeitung beschrieben und geben Rückschlüsse auf die Grundidee und die Hintergründe der Arbeit.

Literaturverzeichnis

[BBB+03] Ingeborg Baumer, Tanja Beck, Uwe Berger, Claudia Brinks, Anneliese Fearns, Michael Friedrich, Gerhard Gruhler, Uwe Jäger, Regina Kammer de Orozco, Wolfgang Küchlin, Richard Leiner, Jean-Francois Levesque, Bernhard Möule, Helmut Malz, Robert Massen, Qinghao Meng, Gerd Nusser, Dorin Popescu, Tobias Rapp, Hubert Roth, Klaus Schiling, Dietmar Schmid, Christina Spilca und Marianne Weber: *Telelaboratorien, neue Elemente moderner Hochschullehre*, Schmid,Dietmar AND Druhler,Gerhard AND Fearns,Anneliese, Aalen, 2003.

[BKP02] Gary Bente, Nicole Krämer und Anita Petersen: *Virtuelle Realitäten*, Bd. 5, Hogrefe, Göttingen, 2002.

[Bri09] Manfred Brill: *Virtuelle Realität*, Springer, Berlin, 2009.

[Cor17a] HTC Corporation: *Produkt Hardware*, 2017, URL: https://www.vive.com/de/product/, besucht am 14.11.2017.

[Cor17b] HTC Corporation: *WIR STELLEN VOR: VIVEPORT FÜR ENTWICKLER*, 2017, URL: https://developer.viveport.com/de/develop_portal/, besucht am 14.11.2017.

[Cor17c] Valve Corporation: *SteamVR Plugin*, Juni 2017, URL: https://www.assetstore.unity3d.com/en/#!/content/32647, besucht am 25.11.2017.

[DG15] Walter Doberenz und Thomas Gewinnus: *Visual C# 2015*, Hanser, München, 2015.

[Epp13] Eppendorf AG: *Eppendorf SOP, Standardanweisung für Pipetten*, 2013, URL: www.eppendorf.com/script/binres.php?RID=35500&DOW=1&pb=1daff712b6696ff9, besucht am 08.12.2017.

[Ewa17] Kornelia Ewald: *Der Einfluß von Dosiertechniken auf das Analysenergebnis*, 2017, URL: https://online-shop.eppendorf.de/eshopdownload/downloadbykey/112928_Userguide_1, besucht am 08.12.2017.

[Hau10] Uwe Hausstädtler: *Der Einsatz von Virtual Reality in der Praxis*, Rhombos, Berlin, 2 Aufl., 2010.

[Hei17] Anna Heinrich: *Entwicklung und Implementierung eines Animationstools zur Visualisierung katalytischer Vorgänge am Beispiel der elektrochemischen CO_2 -Reduktion*, Bachelorarbeit, Fakultät für Angewandte Computer- und Biowissenschaften, Hochschule Mittweida, 2017.

[Hä17] Timm Häsemeyer: *Kollaboratives Erkunden von Software mithilfe virtueller Realität in ExplorViz*, Bachelorarbeit, Arbeitsgruppe Software Engineering, Christian-Albrechts-Universität zu Kiel, 2017.

[JH09] Klaus Jenewein und Danica Hundt: *Wahrnehmung und Lernen in virtueller Realität – Psychologische Korrelate und exemplarisches Forschungsdesign*, Techn. Ber., Otto-von-Guericke-Universität Magdeburg, 2009.

[JK05] Gerard Jounghyun Kim: *Designing Virtual Reality Systems*, Springer Verlag, London, 2005.

[KLH10] Silke Konsorski-Lang und Michael Hampe: *The Design of Material, Organism, and Mind*, Springer, Berlin, 2010.

[Lab17] Labster: *About Labster*, 2017, URL: https://www.labster.com/about/, besucht am 14.11.2017.

[Ltd17] Immersive VR Education Ltd.: *Engage Education Platform*, 2017, URL: www.immersivevreducation.com/engage-education-platform/, besucht am 08.12.2017.

[MMSS12] Ralf Martens-Menzel, Gerhard Schulze und Jürgen Simon: *Maßanalyse, Theorie und Praxis der Titrationen mit chemischen und physikalischen Indikationen*, Walter de Gruyter, Berlin, 2012.

[Mur17] Jeff Murray: *Building Virtual Reality with Unity and Steam VR*, CRC Press, Boca Raton, 2017.

[Rea17] EON Reality: *AUGMENTED AND VIRTUAL REALITY EDUCATION*, 2017, URL: https://www.eonreality.com/applications/augmented-virtual-reality-education/, besucht am 14.11.2017.

[Rol08] Ingo Rollwagen: *Zeit und Innovation*, Bd. 3, transcript, Bielefeld, 2008.

[Sch17] Gabor Schulze: *Entwurf und Implementierung einer dynamischen Softwareplattform für konfigurier- und erweiterbare Simulationen auf Basis der Unity Engine*, Bachelorarbeit, Fakultät für Angewandte Computer- und Biowissenschaften, Hochschule Mittweida, 2017.

[Sci17] MEL Science: *MEL Chemistry VR*, 2017, URL: https://melscience.com/vr/, besucht am 14.11.2017.

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

[SZ04] Katie Salen und Eric Zimmermann: *Rules of Play, Game Design Fundamentals*, The MIT Press, Massachusetts, 2004.

[Tec17a] Unity Technologies: *Unity User Manual (2017.2)*, Okt. 2017, URL: https://docs.unity3d.com/Manual/index.html, besucht am 18.11.2017.

[Tec17b] Unity Technologies: *Unternehmensfakten*, 2017, URL: https://unity3d.com/de/public-relations, besucht am 15.11.2017.

[VdK16] Eric Van de Kerckhove: *HTC Vive Tutorial for Unity, Using The Controllers With Physics Objects*, Dez. 2016, URL: https://www.raywenderlich.com/149239/htc-vive-tutorial-unity, besucht am 18.11.2017.

Selbstständigkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig angefertigt, nicht anderweitig zu Prüfungszwecken vorgelegt und keine anderen als die angegebenen Hilfsmittel verwendet habe. Sämtliche wissentlich verwendete Textausschnitte, Zitate oder Inhalte anderer Verfasser wurden ausdrücklich als solche gekennzeichnet.

Mittweida, den 11. Dezember 2017

Kerstin Knura