

## 2. Praxistransferbericht


# Konzeption und Implementierung von Verhaltensmustern

für virtuelle SchülerInnen im VR-Klassenzimmer

Name, Vorname: Zhang, Yüxi  
Matrikelnummer: 684163  
Ausbildungsbetrieb: Universität Potsdam  
Studienjahrgang: 2020  
Fachbereich: Duales Studium Wirtschaft • Technik  
Studiengang: Informatik  
Modul: 2-2071-TIT20A – Objektorientierte Programmierung (2. Praxistransferbericht)  
Betreuer Hochschule: Prof. Dr. Gerrit Kalkbrenner  
Betreuer Unternehmen: Axel Wiepke  
Anzahl der Wörter: 2924

Vom Ausbildungsleiter zur Kenntnis  
genommen:

23. 08. 2021 F. Zula  
.....  
(Datum/Unterschrift)

Werder (Havel), den 21.08.2021   
.....  
(Datum/Unterschrift der/des Studierenden)

## **Zusammenfassung**

Das Ziel des vorliegenden Praxistransferberichts ist die Konzeption und Implementierung von Verhaltensmustern für virtuelle Schüler/-innen des VR-Klassenzimmer Projekt der Universität Potsdam.

Für die Verständlichkeit der Arbeit werden die theoretischen Grundlagen von der Objektorientierten Programmierung erläutert. Dazu gehören: die Grundkonzepte und -prinzipien der Objektorientierung wie z.B. Klassen, Objekte und Methoden, Datenkapselung und Polymorphie, sowie die grafische Darstellung eines Klassendiagramms (UML-Diagramms).

Im Anschluss findet eine kurze Erklärung des VR-Klassenzimmers statt, gefolgt von einer Anforderungsanalyse, wobei die Voraussetzungen und Ansprüche des zu erstellenden Programmes beschrieben werden. In der Konzeption wird sich mit dem Aufbau des bestehenden Projekts befasst sowie Verhaltensmuster recherchiert. Die Implementierung erfolgt mit der Programmiersprache C# (C-Sharp), bei der in die gegebene Softwareumgebung zwei Verhaltensmuster eingebettet werden. Diese wurden mit einer gewählten Methode getestet und evaluiert. Diese können daraufhin den laufenden Betrieb genommen werden.

## **Abstract**

The goal of this practical transfer report is the conception and implementation of behavior patterns for virtual students of the VR-Classroom Project of the University of Potsdam.

For the comprehensibility of the work, the theoretical basics of object-oriented programming are explained. This includes: the basic concepts and principles of object orientation such as classes, objects and methods, data encapsulation and polymorphism as well as the graphical representation of a class diagram (UML-diagram).

Subsequently, a short explanation of the VR-Classroom takes place, followed by a requirement analysis, whereby the conditions and requests at the program are described. In the conception, the structure of the existing project is dealt with and also the behavioral patterns are researched. The implementation is accomplished with the programming language C# (C-Sharp), where two behavior patterns are embedded into the given software environment. These were tested and evaluated with a chosen method. These can be taken into the running operation.

## **Inhaltsverzeichnis**

<b>1</b>	<b>Einleitung .....</b>	<b>1</b>
<b>2</b>	<b>Theoretische Grundlagen von Objektorientierter Programmierung.....</b>	<b>2</b>
2.1	Objektorientierte Programmierung (OOP) .....	2
2.2	UML-Diagramm (Klassendiagramm) .....	3
<b>3</b>	<b>VR-Klassenzimmer.....</b>	<b>4</b>
<b>4</b>	<b>Anforderungsanalyse .....</b>	<b>6</b>
<b>5</b>	<b>Konzeption .....</b>	<b>7</b>
5.1	Recherche zu exemplarischen Verhaltensmustern .....	7
5.2	Einarbeitung in das VR-Klassenzimmer .....	8
<b>6</b>	<b>Implementierung .....</b>	<b>9</b>
<b>7</b>	<b>Evaluation .....</b>	<b>10</b>
<b>8</b>	<b>Fazit und Ausblick.....</b>	<b>11</b>
<b>9</b>	<b>Literaturverzeichnis .....</b>	<b>12</b>
<b>10</b>	<b>Eidesstattliche Erklärung .....</b>	<b>13</b>
<b>11</b>	<b>Anhang.....</b>	<b>14</b>

## **Abbildungsverzeichnis**

Abbildung 1: Beispiel Klassendiagramm „Kino“ .....	3
Abbildung 2: Screenshot aus dem VR-Klassenzimmer, Die Schüler Harley und Jonas melden sich.....	4
Abbildung 3: vereinfachtes UML-Diagramm (Klassendiagramm) zum VR-Klassenzimmer ...	8
Abbildung 4: : Testversuch bei Veränderung der Variablen bei der Wahrscheinlichkeit .....	10
Abbildung 5: Webanwendung für die Steuerung und die Einstellungen der vSuS und des VR-Klassenzimmers.....	14
Abbildung 6: Unity-Interface mit ChemistryRoom, Play-Button (rot) .....	16
Abbildung 7: Einordnung der Verhaltensweisen in der Webanwendung. Legende: Good behaviours (Mitarbeit/gutes Verhalten), Bad behaviours (Störung, schlechtes Verhalten), Impuls (Interaktion von LuL mit vSoS) (positiv, neutrag, negativ, rufen) Einstufung in grün, gelb und rot.....	16
Abbildung 8: Array der Verhalten für Klassenclown und Streber, Kommentar Partneraktionen (_pairActions) kann Probleme aufrufen .....	19
Abbildung 9: boolesche Variablen: IsclassClown, IsNerd .....	19
Abbildung 10: Seitenpanel, Änderung (rot) nach Einfügen von IsclassClown und IsNerd.....	19
Abbildung 11: NonScriptedClassBehaviour, bereits implementierte Funktion mit kleinen Fehler ("<" muss ">" sein).....	19
Abbildung 12: Schnittstelle isclassClownBehaviour(), while Schleife mit Übergabe der Verhaltensweisen an die Funktion HandleBehaviour() falls wahr (falls Schüler ausgewählt)	20
Abbildung 13: Schnittstelle isNerdBehaviour(), while Schleife mit Übergabe der Verhaltensweisen an die Funktion HandleBehaviour() falls wahr (falls Schüler ausgewählt)	20
Quelle: Abbildung 2: Axel Wiepke	

## **Tabellenverzeichnis**

Tabelle 1: Tabelle für Testversuch bei Veränderung der Variablen bei der Wahrscheinlichkeit .....	10
Tabelle 2: Datenkapselung - Einstellmöglichkeiten für die Zugriffsrechte .....	16
Tabelle 3: Übersicht aller Verhalten (Behaviour) mit Erklärung, Übersetzung, Einordnung und Darstellung .....	18
Quelle: Tabelle 1: [Sil20] S.52 (nachgebaut)	

## **Abkürzungsverzeichnis**

C# - C-Sharp (Programmiersprache).....	1
Git.UP - Git Universität Potsdam .....	14
HWR - Hochschule für Wirtschaft und Recht .....	1
IDE - integrated development environment (integrierte Entwicklungsumgebung) .....	6
LFS - Large File Storage .....	14
LuL - Lehramtsstudentinnen und Lehramtsstudenten .....	4
OOP - Objektorientierte Programmierung .....	1
UML - Unified Modeling Language .....	3
VR - Virtual Reality .....	4
vSoS - virtuelle Schülerin oder Schüler .....	9
vSuS - virtuelle Schülerinnen und Schüler.....	1

## 1 Einleitung

In der menschlichen Entwicklung lernt jeder bewusst oder auch unbewusst kategorisch zu denken, denn es umgibt uns in jeglicher Form. Ob es der Granny Smith Apfel ist, der in der Pause verzehrt wird oder der Audi A3 ist, mit dem der Weg zur Arbeit täglich zurückgelegt wird. Diese Objekte können stets in einer übergeordneten Gruppe zusammengefasst werden. Bei den Granny Smiths gibt es generell die Möglichkeit der Zuordnung zu den Äpfeln oder den Früchten und beim A3 Modell können es generell Audis, Autos oder Fortbewegungsmittel sein. Dieses kategorische Denken findet auch in der Informatik Anwendung und zwar in Form der Objektorientierten Programmierung (OOP). Die Probleme werden dabei auf einer für uns natürlichen Art und Weise modelliert, die der menschlichen Denkweise ähnelt. Da sich das objektorientierte Paradigma in den letzten Jahren in Folge der Digitalisierung, sowohl im Ausbildungsbereich, als auch im industriellen Umfeld durchgesetzt hat, rückt die Thematik der Objektorientierten Programmierung immer mehr in den Vordergrund. [Bol14] (S.2)

Die Objektorientierte Programmierung wird auch im Forschungsprojekt „VR-Klassenzimmer“ in Form der Programmiersprache C-Sharp (C#) genutzt. Das VR-Klassenzimmer bietet Trainingssimulationen für Lehramtsstudierende, wodurch Ressourcen gespart und ethische Bedenken vernachlässigt werden können. Dabei wird das Ziel verfolgt, die Simulationen so realitätsnah wie möglich zu gestalten. Indem menschliche Interaktionen nachgestellt werden, sollen Herausforderungen aber auch Chancen resultieren, mit dem Unterrichtsgeschehen vertraut zu werden (Unterrichten, Gespräche, Umgang mit Unterrichtsstörungen). Die Realitätsnähe wird durch eine Unterscheidung der virtuellen Schüler und Schülerinnen (vSuS) begünstigt, welche in Form von automatisierten Verhaltensmustern bzw. „Charakteren“ teilweise geschaffen werden kann.

Aus diesem Grund besteht der Bedarf einer Implementierung von Verhaltensmustern, der im Rahmen des Praxistransferberichtes der Hochschule für Wirtschaft und Recht (HWR) und des Instituts für Informatik der Universität Potsdam, in den bestehenden Software-Stack des VR-Klassenzimmer Projekts konzipiert und umgesetzt wird.

## **2 Theoretische Grundlagen von Objektorientierter Programmierung**

Die Programmierung im Allgemeinen ist eine Anordnung von Anweisungen, welche in einer Hochsprache verfasst wird. Diese Sprache wird durch einen Compiler bzw. Interpreter in eine Sprache umgewandelt, die der Computer versteht. Dabei können Algorithmen eingebaut werden, welche ein Problem mittels einer Handlungsvorschrift aus endlich vielen Einzelschritten löst. [Fra19] (S.101)

### **2.1 Objektorientierte Programmierung (OOP)**

Der Fokus dieser Arbeit liegt bei der Objektorientierten Programmierung (OOP) und bei der Programmiersprache C# (C-Sharp). Die OOP hat sich in den letzten Jahren in der Industrie aufgrund ihrer Vorteile durchgesetzt. Die Variablen und Methoden (Funktionen) werden zu einer Einheit zusammengefasst, wodurch die Verständlichkeit und Übersicht über das Programm erhöht und die Fehlerquote gesenkt wird. Die Programme können problemlos erweitert und an jeweilige Probleme angepasst werden. Teile der Programme lassen sich einfach und flexibel wiederverwenden. Wichtige Konzepte der OOP sind Klassen, Vererbung und Polymorphie. Die Objekte können verschiedene Formen haben, wie Individuen (Menschen und Tiere), materielle Gegenstände (Auto, Haus) und abstrakte Konzepte (Rechnungen, Plan). Sie haben einen individuellen Charakter, unabhängig davon, ob sie äußerlich differenzierbar sind oder nicht. Des Weiteren besitzen Objekte Eigenschaften und können Verhalten aufweisen. Zum Beispiel ein Fahrrad (Objekt) ist blau (Attribut) und hat zwei (Attribut) Räder und einen Sattel als Sitzmöglichkeit (Methode) und kann eine Person transportieren (Methode). Alle Objekte, die denselben Bauplan haben, können in einer Klasse zusammengeführt werden. Eine Klasse besteht somit aus Attributen (Variablen) und Methoden (Funktionen). [Bew18] (S.30) Aus diesem Grund existieren Superklassen (Oberklassen) und Subklassen (Unterklassen). Diese bilden den Ausgangspunkt für die Vererbung. Die Superklasse kann als Baustein für Subklassen genutzt werden, wobei die Subklassen die Elemente der Superklasse übernehmen und weitere Elemente (Attribute und Methoden) besitzen können. [Rau00] (S.139-140) Die Vererbung bildet die Grundlage für die Polymorphie (Vielgestaltigkeit). Dabei können verschiedene Objekte auf die gleiche Methode unterschiedlich reagieren. Dies wird mittels Überschreibung von Methoden innerhalb einer Klassenhierarchie realisiert, wodurch die Besonderheiten der Subklasse berücksichtigt werden. [Dör16] (S.307) In der OOP können auch Zugriffsrechte auf die jeweiligen Attribute und Methoden verwaltet werden (Datenkapselung). Eine Übersicht ist im Anhang in der Tabelle 1 (S.16) zu sehen. [Sil20] (S.52) Es gibt noch weitere Konzepte und Prinzipien, wie z.B. Interfaces, abstrakte Klassen, Überladen/Überschreiben von Methoden, Abstraktion und Aggregation und Komposition, jedoch würde diese nicht den Rahmen dieser Arbeit entsprechen. Dies gilt auch für die Syntax von C# und dem Vergleich zu anderen Objektorientierten Programmiersprachen wie Java, C++ oder Python, wobei jedoch das Konzept der Objektorientierung bestehen bleibt.

## 2 Theoretische Grundlagen von Objektorientierter Programmierung

### 2.2 UML-Diagramm (Klassendiagramm)

Je größer das Projekt wird, desto größer werden die Interaktionen der Objekte untereinander. Es ist möglich die Beziehungen in Form eines Textes darzulegen. Eine grafische Darstellung stellt eine Alternative dazu da. Bei der OOP ist die Unified Modeling Language (UML) als grafische Beschreibungssprache ein weit verbreiteter Standard. [Mül17] (S. 154-155) Dabei stehen eine Vielzahl von Diagrammtypen zu der Spezifikation von Struktur und Verhalten der zu entwickelnden Programme sowie zur Dokumentation von Aspekten der Implementierung zur Verfügung. [Bol14] (S.495) Klassendiagramme bieten ein architekturelles Konstrukt, das die Systemmodellierung unterstützt. Es wird durch folgende Regeln bestimmt. [Rum04] (S.41)

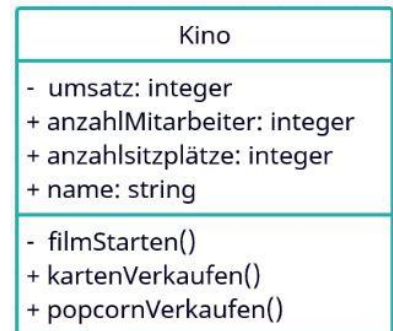


Abbildung 1: Beispiel  
Klassendiagramm „Kino“

Ein beispielhafter Aufbau eines Klassendiagramms ist in der Abbildung 1 (S.3) zu sehen. Dabei ist die Klasse (Rechteck) als Kino abgebildet. Darunter befinden sich in der ersten Zeile alle Attribute der Klasse, wie zum Beispiel die Anzahl der Mitarbeiter im Kino, der Name des Kinos, den Umsatz und die Anzahl der Sitzplätze. Dahinter befinden sich die jeweiligen Datentypen (string oder integer). Darunter befinden sich die Methoden der Klasse. Es kann ein Film gestartet, sowie Karten als auch Popcorn verkauft werden. Die vorher genannte Datenkapselung (siehe 2.1 und Anhang Tabelle ...) kann auch hier schon in Form von Zeichen vorgenommen werden (public +, private -, protected #, package ~). Es existieren noch weitere graphische Mittel, wie z.B. das Unterstreichen von Methoden oder Attributen, wodurch sie zu Klassenmethoden bzw. -attributen werden oder das Festlegen von Beziehungsarten (Pfeile und andere Anmerkungen) zwischen den Klassen und Objekten.



### 3 VR-Klassenzimmer

Virtual Reality (Virtuelle Realität oder VR) ist eine dreidimensionale computergenerierte Umgebung. Dabei wird das Ziel verfolgt, eine Situation nachzubilden, mit welcher der Nutzer (User) interagiert. Mittels gezielten Feedbacks und Reflexion lässt sich dadurch Wissen aneignen. [Lac20] (S.76) Diese Umgebung kann über ein Head-Mounted-Display (sogenannte VR-Brille) oder über die Software Unity<sup>1</sup> dargestellt werden. Das VR-Klassenzimmer ist ein Projekt von Axel Wiepke, Raphael Zender, sowie Eric und Dirk Richter [Wie19] und weiteren Mitarbeitern der Universität Potsdam<sup>2</sup>. Das Projekt dient dem Verhaltenstraining von Lehramtsstudentinnen und Lehramtsstudenten (LuL) und soll ihnen die Möglichkeit bieten sich auf den späteren realen Unterricht vorzubereiten. Dabei üben sie, praxisnah und zielgerichtet auf Unterrichtsstörungen einzugehen, um dadurch ihre Klassenmanagementkompetenzen unter Beweis zu stellen und gegebenenfalls zu verbessern. Es wird ein Klassenzimmer nachgebildet, indem sich virtuelle SchülerInnen und Schüler (vSuS) befinden (siehe Abbildung 2 (S.4)).



Abbildung 2: Screenshot aus dem VR-Klassenzimmer, Die Schüler Harley und Jonas melden sich

Die vSuS sind optisch in der zweiten Sekundarstufe und ethnisch divers. Im Klassenzimmer sind verschiedene Anordnungen der Tische und verschiedene Umgebungen, wie z.B. ein Klassenraum für Chemie möglich. Der LuL kann real im 3x3 Meter großen (Play Area) Raum laufen, wodurch sich der Charakter im VR-Klassenzimmer bewegt. Jedoch ist der Raum begrenzt, weshalb Teleportationen mittels eines Controllers genutzt werden. Ein Coach/eine Coachin kann über eine Weboberfläche (siehe Anhang Abbildung 5 (S.14)) viele Aspekte des VR-Klassenzimmers kontrollieren. In der Webanwendung ist es möglich das Wetter (Sonne, Regen, Schnee) und Skripte einzustellen und auch Aufnahmen (auch aus Perspektive der Schüler) für eine bessere Reflexion zu erstellen. Die Skripte sind voreingestellte Aktionen, die von den Schülern ausgelöst werden, wie z.B. Mitarbeit (z.B. mitschreiben oder Hand heben) oder Störverhalten (z.B. Papierball werfen). Es gibt auch die Möglichkeit die vSuS gezielt auszuwählen, wobei der/die Coach/in sie manuell ansteuert. Die Aktionen der vSuS basieren auf realen Unterrichtssituationen in einer Klasse und werden nach dem Buch „Techniken der Klassenführung“ [Kou06] kategorisiert. Einige dieser Aktionen (Übersicht siehe Anhang Tabelle 2 (S.17-18)) können zu Verhaltensmuster in Form eines Skripts zusammengefasst werden, um einen „Charakter“ widerzuspiegeln. Diese helfen realitätsnahe Interaktionen zu ermöglichen. [Wie19]

<sup>1</sup><https://unity3d.com/de> Abrufdatum: 29.06.2021

<sup>2</sup>[https://gitup.uni-potsdam.de/mm\\_vr/vr-klassenzimmer/wikis/Kernteam](https://gitup.uni-potsdam.de/mm_vr/vr-klassenzimmer/wikis/Kernteam) Abrufdatum: 29.06.2021



## 4 Anforderungsanalyse

Es gelten die allgemeinen Grundkonzepte der OOP, die oben genannt worden sind (siehe 2.1). Dazu gehört weiterhin die Einarbeitung in die aktuelle Softwarestruktur des VR-Klassenzimmers. Die Wahl der Objektorientierten Programmiersprache fiel dabei auf C#, da Unity die gewählte 3D Laufzeit- und Entwicklungsumgebung (bzw. Spiel-Engine) ist. Alle entwickelten Skripte werden in der Programmiersprache geschrieben<sup>3</sup>, weswegen auch die Syntax der Sprache ein Teil der Einarbeitung ist. Aus diesem Grund kommt auch die integrierte Entwicklungsumgebung (integrated development environment, kurz IDE) Visual Studio<sup>4</sup> zum Einsatz, da sie wie C# auf Unity abgestimmt ist. Zudem liegt durch ihre Bekanntheit eine gute Dokumentation von Informationen vor. Jede IDE hat je nach Entwicklung eigene Funktionen, wobei meistens ein Code-Editor, Compiler und weitere Funktionen gegeben sind. Des Weiteren müssen die vorgegebenen Normen des Projekts<sup>5</sup>, die sogenannten „Standards“ eingehalten werden. Dazu gehören: Namenskonventionen, Vorgaben für Kommentare und weitere Richtlinien für die Programme bzw. Skripte, die geschrieben werden. Aus der Recherche über die Verhaltensmuster sollen quantifizierbare Merkmale hervorgehen, die evaluiert und gegenübergestellt werden. Dafür soll eine geeignete Evaluationsmethode gewählt werden. Diese Verhaltensmuster sollten bestenfalls zu den bereits implementierten Verhalten (z.B. Hand heben oder sich schlagen) und allgemein ins Projekt passen, da andernfalls neue Verhalten implementiert werden müssen, die nicht dem Hauptziel dieser Arbeit entsprechen.

---

<sup>3</sup><https://unity.com/de/how-to/beginner-game-coding-resources> Abrufdatum: 14.08.2021

<sup>4</sup><https://visualstudio.microsoft.com/de/vs/> Abrufdatum: 15.08.2021

<sup>5</sup>[https://gitup.uni-potsdam.de/mm\\_vr/vr-klassenzimmer/wikis/Standards](https://gitup.uni-potsdam.de/mm_vr/vr-klassenzimmer/wikis/Standards) Abrufdatum: 13.08.2021

## 5 Konzeption

Es existiert (zum Zeitpunkt der Erstellung dieser Arbeit) keine UML-Darstellung oder andere grafische Darstellungen über das VR-Klassenzimmer Projekt. Da das Projekt sehr umfangreich ist, wird sich hierbei nur auf die wesentliche Aspekte beschränkt, die für die gegebene Problemstellung relevant sind.

### 5.1 Recherche zu exemplarischen Verhaltensmustern

Die Recherche nach den Verhaltensmustern erweist sich als recht komplex. Es ist schwer festzulegen, ob ein Kind einen „guten“ oder „schlechten“ Charakter hat, da diese Sichtweise sehr subjektiv ist. Zuerst muss sich mit dem Begriff: „Charakter“ psychologisch und soziologisch auseinandergesetzt werden.

Laut dem Oxford Languages<sup>6</sup> (Dictionary) ist der Charakter ein „individuelles Gepräge eines Menschen durch ererbte und erworbene Eigenschaften, wie es in seinem Wollen und Handeln zum Ausdruck kommt.“ Ein Charakter ist ein Gebilde aus vielen einzelnen Facetten, welche sich im Laufe des Lebens stetig ändern kann. Wissenschaftlich gesehen, wird danach gestrebt danach für die Charakterzüge biologische Anhaltspunkte zu finden, um diese Abläufen zuzuordnen. [Ewa24] Es können, jedoch alle möglichen Faktoren Einfluss auf die Charakterbildung haben. Dies kann bereits bei der Geburt in eine bestimmte soziale Schicht passieren, wobei in Untersuchungen festgestellt wurde, dass Kinder mit höherem sozialen Status in Schulen im Vergleich zu anderen, wesentlich bessere Ergebnisse erbringen und generell kompetenter sind. Mit einem niedrigen sozialen Status wurden in einer Untersuchung mit Versuchspersonen einige Adjektive, wie z.B. „faul“, „ungebildet“, „drogenabhängig“ und „nicht intelligent“ assoziiert. Zudem können Stereotypen in bestimmten Kontexten die Urteile und das Verhalten beeinflussen. So kann der soziale Status z.B. in Bezug auf Leistung auch den Stereotyp fleißig hervorrufen. „Aus diesem Grund ist anzunehmen, dass Lehrkräfte und Lehramtsstudierende zwar auch Einstellungen und Stereotypen über niedrige und hohe soziale Schichten im Allgemeinen haben, im schulischen Kontext aber hauptsächlich leistungsspezifische Stereotypen aktiviert werden.“ [Glo20] (S.133-158) Die zu implementierenden Verhaltensweisen im VR-Klassenzimmer fallen auch in diesen Leistungskontext. Da ein Charakter einer Person, aufgrund der Komplexität und der möglichen Änderung im Laufe des Lebens, sich schwer empirisch zusammenfassen und sich als ein Begriff darstellen lässt, wird für das VR-Klassenzimmer auf leistungsbezogene Stereotypen zurückgegriffen. Die ausgewählten Stereotypen sind der „Klassenclown“ und der bekannte „Streber“. Ein Klassenclown versucht Aufmerksamkeit auf sich zu ziehen, spielt und arbeitet wenig bis gar nicht mit. Ihm werden die Verhalten: "Eating", "Drinking", "Staring", "Playing", "Tapping", "Hitting", "Chatting" und "Throwing" zugewiesen (bereits implementiert). Ein Streber hingegen ist fleißig, passt auf und arbeitet viel und gerne mit. Er erhält die Verhalten: "Writing", "RaiseArm", "PlakatPartner", "AskQuestion" (siehe Anhang Abbildung 7 (S.16)). Diese Verhaltensmuster können mit wenig Aufwand überarbeitet und somit an unterschiedliche Anforderungen angepasst werden.

---

<sup>6</sup><https://languages.oup.com/google-dictionary-de/> Abrufdatum: 19.08.2021

## 5 Konzeption

### 5.2 Einarbeitung in das VR-Klassenzimmer

Das VR-Klassenzimmer besteht aus vielen verschiedenen Komponenten. Um einen groben Überblick über die Software zu geben, bietet sich eine UML-Darstellung (Klassendiagramm) an (siehe Abbildung 3 (S.8)). Diese wird jedoch stark vereinfacht und aufs wesentliche reduziert, wodurch die Komplexität abnimmt (Schnittstelle IEnumerator als Methode dargestellt). Unity besitzt sogenannte Unity-Assets, also Objekte, welche in der Entwicklungsumgebung genutzt werden können. Assets können außerhalb, wie z.B. eine Audio-Datei, ein Bild und ein 3D-Modell oder innerhalb von Unity erstellt werden, wie z.B. ein Animator-Controller und Audio-Mixer.<sup>7</sup> So gibt es auch beim VR-Klassenzimmer Assets, welche im sogenannten Asset-Store heruntergeladen wurden und andere die vom Team des VR-Klassenzimmers selbst entwickelt wurden (diese beginnen mit own z.B. ownScripts und ownAnimations).<sup>8</sup> In den Verzeichnissen „Assets\ownScripts\Students“ und „Assets\ownScripts\Students\OneStudent“ gibt es zwei Dateien, die hauptsächlich für die Implementierung relevant sind (SpecialBehaviours und BehaviourController). In der Klasse SpecialBehaviours werden spezielle Verhalten in Form eines Arrays eingebaut. Hier werden die Verhalten für die Stereotypen „Klassenclown“ (\_classClown) und „Streber“ (\_nerd) implementiert. In der Klasse BehaviourController können den vSuS mithilfe einer Bool die Stereotypen „Streber“ oder „Klassenclown“ zugeordnet werden. In der jeweiligen Methode isClassclownBehaviour oder isNerdBehaviour wird auf SpecialBehaviours zugegriffen. Zufällig wird eine Stelle eines der beiden Arrays an die Methode HandleBehaviour übergeben, wodurch die vSuS die vorgegebenen Verhalten aus dem Array ausführen. Der StudentController (nicht relevant für Implementierung) übergibt den vSuS ihre Attribute, wie z.B. die Rotation, Name, ID usw. So kann man sagen, dass ein Klassenclown oder Streber ein Objekt von vSuS ist. Eine detailliertere Erklärung der Abläufe folgt in der Implementierung (6.).

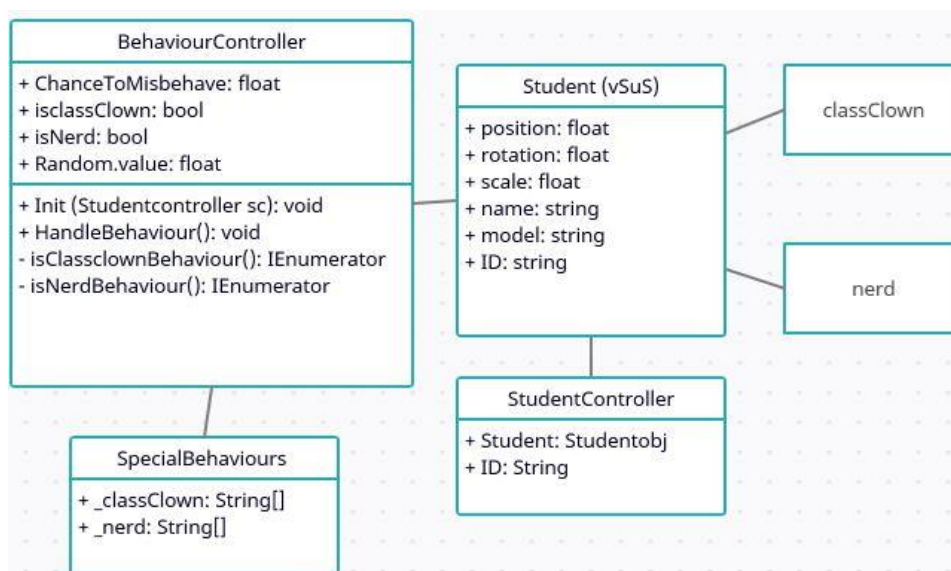


Abbildung 3: vereinfachtes UML-Diagramm (Klassendiagramm) zum VR-Klassenzimmer

<sup>7</sup><https://unity3d.com/de/quick-guide-to-unity-asset-store> Abrufdatum: 16.08.2021

<sup>8</sup>[https://gitup.uni-potsdam.de/mm\\_vr/vr-klassenzimmer/wikis/DeveloperEinf%C3%BChrung](https://gitup.uni-potsdam.de/mm_vr/vr-klassenzimmer/wikis/DeveloperEinf%C3%BChrung)  
16.08.2021

Abrufdatum:

## 6 Implementierung

Das Ziel der Implementierung ist es, die genannten Stereotypen in den aktuellen Stand des Projekts einzubringen. Zuerst muss die Entwicklungsumgebung eingerichtet werden, damit der Code getestet werden kann. Eine Erklärung für die Einrichtung der Entwicklungs- und Testumgebung (Unity) ist im Anhang unter der Abbildung 5 (S.14) zu finden.

In `SpecialBehaviours` werden die Arrays mit den genannten Verhalten zugewiesen und eingebaut (siehe Abbildung 8 (S.19)). In der Datei `BehaviourController` werden boolesche Variablen eingefügt (siehe Anhang Abbildung 9 (S.19)), wodurch im Seitenpanel von Unity ein Kästchen entsteht (siehe Anhang Abbildung 10 (S.19)). Somit kann in der Entwicklungsumgebung direkt eine virtuelle Schülerin oder ein Schüler (vSoS) als Klassenclown oder Streber ausgewählt werden. Solange die boolesche Variable wahr ist, soll eine vSoS ein Verhalten aus dem Array durchführen. Um dies zu realisieren, existiert bereits in der Klasse `BehaviourController` eine Schnittstelle (`IEnumerator`)<sup>9</sup> mit dem Namen `NonScriptedClassBehaviour()`. Diese wird kopiert und an die Anforderungen angepasst (siehe Anhang Abbildung 11 (S.19)). Es gibt eine Float (Zahl mit Nachkommastellen) namens `ChanceToMisbehave`. Diese wird in der Schnittstelle alle 10 Sekunden um eine Float von 0,004f (entspricht 0,4%) erhöht. Zudem wird auch stetig eine neue zufällige `Random.value`<sup>10</sup> (Float zwischen 0 und 1) generiert. Falls der Wert von `ChanceToMisbehave` größer (in `NonScriptedBehaviour()` kleiner als - Fehler) als `Random.value` ist, dann wird eine zufällige Stelle des Arrays, also ein Verhalten, an die Funktion `HandleBehaviour()` übergeben (nicht dargestellt wegen Komplexität). Die Funktion lässt eine vSoS ein Verhalten ausführen (siehe Anhang Abbildung 12 (S.20)). Entsprechend wurde dies auch für den Streber (`isNerdBehaviour()`) durchgeführt (siehe Anhang Abbildung 13 (S.20)). Es kann derzeit aber noch zu Komplikationen kommen. Falls eine vSoS beide Verhaltensweisen zugewiesen bekommt, dann führt sie/er nur Verhalten aus einem Array aus. Es ist ratsam eine Exception (Ausnahme) einzubauen, welche diesen Fehler auffängt. Außerdem kann gesagt werden, dass ein Fehler ausgegeben werden soll oder der vSoS auf „Idle“ verweilt. Weiterhin zu beachten ist, dass zwei vSoS nicht direkt nebeneinandersitzen sollten. Dies ist durch die Partnerinteraktionen (siehe Anhang Abbildung 7 (S.19)) bedingt, da eine Aktion während der Ausführung einer anderen diese überschreiben könnte (ein Fehler wird ausgegeben). Diese Interaktion könnte behoben werden, indem die Partnerinteraktionen aus den Arrays ausgelassen werden oder eine Exception eingebaut wird. Des Weiteren wird das Verhalten nicht gestoppt, da z.B. eine vSoS etwas trinkt und solange das Verhalten ausführt, bis ein neues Verhalten zugewiesen wird. Zuletzt werden die Dateien auf einen separaten Branch (Zweig) ins Git-Verzeichnis (Versionsverwaltungssoftware) gepusht (hochgeladen), damit diese weiter eingesehen und entwickelt werden können.

---

<sup>9</sup><https://docs.microsoft.com/de-de/dotnet/api/system.collections.ienumerator?view=net-5.0>  
20.08.2021

<sup>10</sup><https://docs.unity3d.com/ScriptReference/Random-value.html> Abrufdatum: 20.08.2021

## 7 Evaluation

Die Evaluation der programmierten Stereotypen ist als sehr variabel zu betrachten. Nach Ausführung in Unity wird in einer bestimmten Zeit die Wahrscheinlichkeit auf ein Verhaltensweise erhöht. Voreingestellt bei der implementierten Methode `NonScriptedClassBehaviour()` ist eine Wahrscheinlichkeitserhöhung um 0,4% alle 10 Sekunden. Durch die Veränderung dieser Variablen, kann die Wahrscheinlichkeit des Auftretens eines Verhaltens aus den Arrays angepasst werden. Um eine ungefähre Vorstellung davon zu erhalten, was geschieht, wenn diese Variablen verändert werden, wurde eine Übersicht erstellt (siehe Abbildung 4 (S.10)). Dieser Testversuch wurde pro Einstellung drei Mal in jeweils vier Minuten durchgeführt und der Mittelwert ermittelt. Nach vier Minuten wurden beim Ausgangswert (1.) 0,67 Verhalten ausgeführt. Wenn die Wahrscheinlichkeit verdoppelt wird, (2.), dann steigt das Verhalten auf 1,33. Falls stattdessen jedoch das Zeitintervall um die Hälfte verringert wird, dann steigt die Anzahl auf 2. Ein ähnliches Verhältnis in einem extremeren Ausmaß bietet die 10-fache Steigerung der Wahrscheinlichkeit (6.), wodurch die Verhalten auf 3,67 steigen, aber beim Zehntel des Zeitintervalls auf 11,67 Verhalten steigt. Dies liegt daran, dass bei niedrigen Zeitintervallen auch die `Random.value` neu generiert wird und somit den Vergleich von `ChanceToMisbehaviour` mit `Random.value` öfter ausführt. Ein genaueres Ergebnis würden mehr Tests und eine Implementierung des Stopps der Verhalten liefern, da diese kontinuierlich hintereinander ausgeführt werden und so die Zählung bei Wahl des gleichen Verhaltens verfälscht.

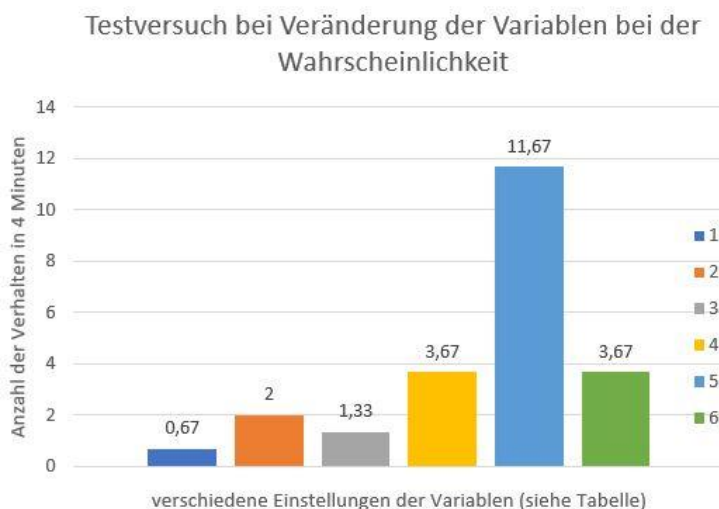


Abbildung 4: : Testversuch bei Veränderung der Variablen bei der Wahrscheinlichkeit

	Anzahl der Verhalten	Zeitintervall in Sekunden	Wahrscheinlichkeit in %	Kommentar
1.	0,67	10	0,4	Ausgangswert
2.	2	5	0,4	halbes Zeitintervall
3.	1,33	10	0,8	doppelte Steigerung der Wahr.
4.	3,67	5	0,8	doppelte Steigerung der Wahr. und halbes Zeitintervall
5.	11,67	1	0,4	1/10 des Zeitintervalls
6.	3,67	10	4	10fache Steigerung der Wahr.

Tabelle 1: Tabelle für Testversuch bei Veränderung der Variablen bei der Wahrscheinlichkeit

## **8 Fazit und Ausblick**

Dieser Praxistransferbericht hat das Thema der Objektorientierten Programmierung nur grob angeschnitten. Dennoch ist zu erkennen, dass die Objektorientierte Programmierung eine erhebliche Relevanz in der heutigen digitalisierenden Welt hat. Es ist möglich sich durch gute Einarbeitung in die Thematik einzufinden, wodurch die Konzeption und Implementierung der Verhaltensmuster in die vorhandene Software gelungen sind. Dies ist auch auf die Programmiersprache C# und die IDE Visual Studio zurückzuführen, da diese sehr bekannt sind und dadurch eine gute Dokumentation von Informationen vorliegt.

Während der Implementierung sind einige Probleme mit der Version des VR-Klassenzimmers und Unity aufgetreten, da das VR-Klassenzimmer stetig erweitert und verändert wird. Die entwickelten Verhaltensweisen sind weiterhin noch ausbaufähig (wie in 6. erläutert) und müssen auch für den/die Coach/in über die Weboberfläche (eigenes Feld bzw. Taste) implementiert werden. Jedoch können die Verhaltensweisen einfach an gewünschte Anforderungen angepasst werden. Zukünftig kann die Performance bzw. Laufzeit des Implementierten Codes und die Wahrscheinlichkeiten für das Auftreten von Verhaltensweisen angepasst werden (7.), um eine realitätsnahe Testumgebung darzustellen.



## 9 Literaturverzeichnis

- [Bew18] Jörg Bewersdorff: *Objektorientierte Programmierung mit JavaScript*. Wiesbaden: Springer Vieweg, 2018.
- [Bol14] Cornelia Boles und Dietrich Boles: *Objektorientierte Programmierung spielend gelernt mit dem Java-Hamster-Modell*. Wiesbaden: Springer Vieweg, 2014.
- [Dör16] Sebastian Dörn: *Programmieren für Ingenieure und Naturwissenschaftler*. Springer Vieweg, 2016.
- [Ewa24] G. Ewald: *Temperament und Charakter*. Berlin, Heidelberg: Springer-Verlag, 1924.
- [Fra19] Claudio Franzetti: *Essenz der Informatik*. Springer Vieweg, 2019.
- [Glo20] Sabine Glock und Hannah Kleen: *Stereotype in der Schule*. Wiesbaden: Springer Fachmedien, 2020.
- [Kai20] Richard Kaiser: *C++ mit Visual Studio 2019 und Windows Forms-Anwendungen: C++17 für Studierende und erfahrene Programmierer – Windows Programme mit C++ entwickeln*. Springer Vieweg, 2020.
- [Kou06] Jacob S. Kounin: *Techniken der Klassenführung. Standardwerke aus Psychologie und Pädagogik. Reprints..* Münster: Waxmann, 2006.
- [Lac20] Maximilian Lackner und Horst Orsolits: *Virtual Reality und Augmented Reality in der Digitalen Produktion*. Wiesbaden: Springer Gabler, 2020.
- [Mül17] Heinrich Müller und Frank Weichert: *Vorkurs Informatik: Der Einstieg ins Informatikstudium*. Wiesbaden: Springer Vieweg, 2017.
- [Rau00] Otto Rauh: *Objektorientierte Programmierung in JAVA: Eine leicht verständliche Einführung*. Wiesbaden: Vieweg+Teubner Verlag, 2000.
- [Rum04] Bernhard Rumpe: *Modellierung mit UML*. Berlin Heidelberg: Springer-Verlag, 2004.
- [Sil20] Christian Silberbauer: *Einstieg in Java und OOP: Grundelemente, Objektorientierung, Design-Patterns und Aspektorientierung*. Springer Vieweg, 2020.
- [Wag16] Christian Wagenknecht: *Programmierparadigmen: Eine Einführung auf der Grundlage von Racket*. Wiesbaden: Springer Vieweg, 2016.
- [Wie19] Axel Wiepke, Eric und Dirk Richter und Raphael Zender: *Einsatz von Virtual Reality zum Aufbau von Klassenmanagement-Kompetenzen im Lehramtsstudium*. N. Pinkwart und J. Konert (Hrsg.). DELFI 2019. Bonn: Gesellschaft für Informatik e.V.: 2019, S. 133-144. Abgerufen 12. Jan. 2021.  
[online] <https://dl.gi.de/handle/20.500.12116/24390>

10 Eidesstattliche Erklärung

---

**10 Eidesstattliche Erklärung**

Ich erkläre ehrenwörtlich:

1. dass ich meinen Praxistransferbericht selbständig verfasst habe,
2. dass ich die Übernahme wörtlicher Zitate aus der Literatur sowie die Verwendung der Gedanken anderer Autoren an den entsprechenden Stellen innerhalb der Arbeit gekennzeichnet habe,
3. dass ich meinen Praxistransferbericht bei keiner anderen Prüfung vorgelegt habe.

Ich bin mir bewusst, dass eine falsche Erklärung rechtliche Folgen haben wird.

Werder (Havel), den 23.08.2021

---

Ort, Datum

Yüxi Zhang

## 11 Anhang

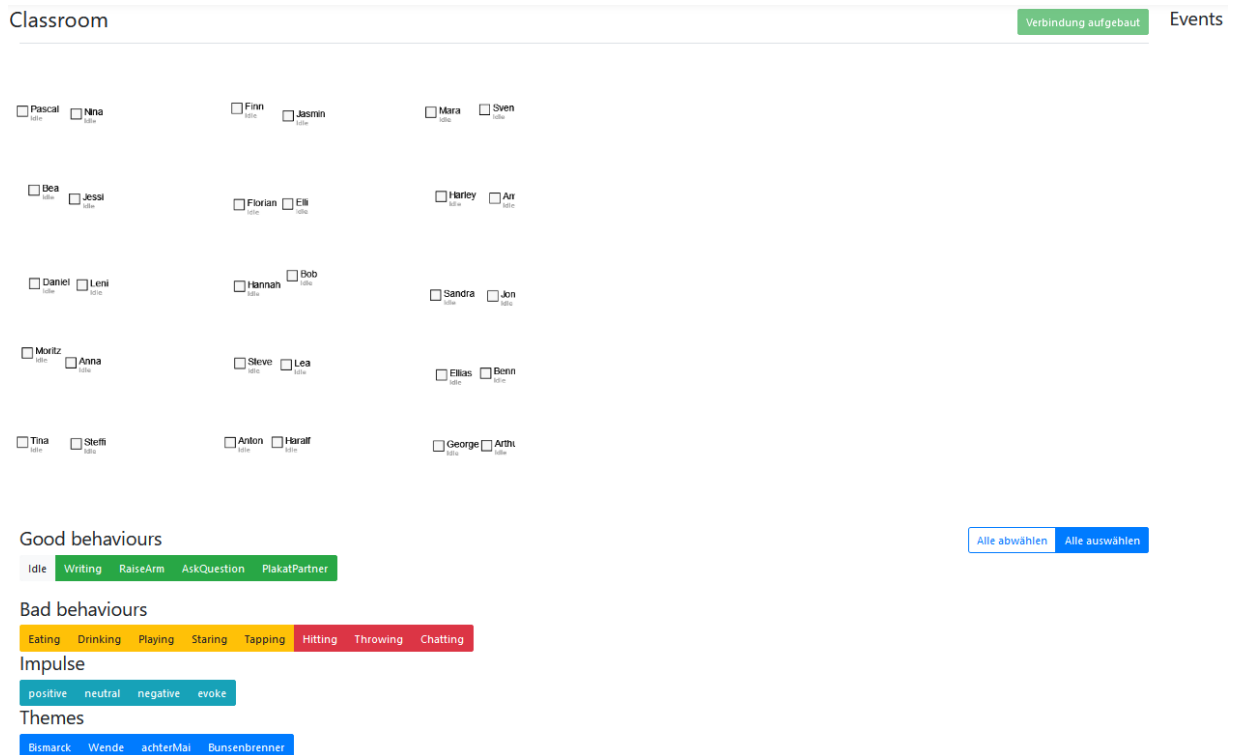


Abbildung 5: Webanwendung für die Steuerung und die Einstellungen der vSuS und des VR-Klassenzimmers

Bevor die Verhaltensmuster implementiert werden können, muss die Entwicklungs- und Testumgebung eingerichtet werden. Die Implementierung erfolgt in das GitLab bzw. Git.UP (Git Universität Potsdam) Verzeichnis für das VR-Klassenzimmer-Projekt<sup>11</sup>. Zuerst wurde sich in das bestehende Softwareprojekt gearbeitet, mittels eines Wikis und dessen Developereinführung welches sich im GitLab befindet, eingearbeitet<sup>12</sup>. In der Einführung werden alle Programme aufgelistet, um das VR-Klassenzimmer bzw. die Testumgebung starten zu können. Besonders wichtig sind hierbei Git und Git LFS (Large File Storage), Unity und Node.js. Git<sup>13</sup> ist ein kostenloses Versionsverwaltungsprogramm, welches genutzt wurde, um die aktuellen Dateien bzw. die aktuelle Version vom VR-Klassenzimmer herunterzuladen. Git LFS<sup>14</sup> dient dazu, größere Dateien herunterzuladen, welche nicht direkt im Repository vorhanden sind (nur über einen Pointer). Unity (Version 2021.1.5f1)<sup>15</sup> ist eine Laufzeit- und Entwicklungsumgebung für Spiele und weiteren Anwendungen. Es wird dafür genutzt, um das VR-Klassenzimmer zu simulieren und darzustellen, falls keine VR-Brille vorhanden ist. Node.js<sup>16</sup> ist eine Laufzeitumgebung für Java-Script, wodurch der Code außerhalb eines Webbrowsers ausgeführt werden kann und für die Darstellung der Webanwendung wichtig ist.

<sup>11</sup>[https://gitup.uni-potsdam.de/mm\\_vr/vr-klassenzimmer/](https://gitup.uni-potsdam.de/mm_vr/vr-klassenzimmer/) Abrufdatum: 29.06.2021

<sup>12</sup>[https://gitup.uni-potsdam.de/mm\\_vr/vr-klassenzimmer/wikis/DeveloperEinf%C3%BChrung](https://gitup.uni-potsdam.de/mm_vr/vr-klassenzimmer/wikis/DeveloperEinf%C3%BChrung) Abrufdatum: 29.06.2021

<sup>13</sup><https://git-scm.com/> Abrufdatum: 29.06.2021

<sup>14</sup><https://git-lfs.github.com/> Abrufdatum: 29.06.2021

<sup>15</sup><https://unity.com/de> Abrufdatum: 30.06.2021

<sup>16</sup><https://nodejs.org/de/> Abrufdatum: 01.07.2021

## 11 Anhang

---

Nach dem Herunterladen des Repositories (Archiv, in dem die aktuellen Daten des VR-Klassenzimmers gespeichert werden) aus dem GitLab kann Unity gestartet werden. Es kann eine Szene ausgewählt werden, welche vom Team des Projekts erstellt wurde. Für diese Arbeit wurde der ChemistryRoom (Chemie Klassenzimmer) ausgewählt (siehe Abbildung 6 (S.16)). Eine genauere Erklärung der Einrichtung von Unity befindet sich im folgenden Abschnitt.

Diese Erklärung ist für die Betriebssysteme: Windows 10 und für Linux (Ubuntu 20.04) erstellt (zuerst Windows / dann Linux, sonst beides).

Bei Windows kann die Software über die Webseiten heruntergeladen werden. Git, Git LFS und Node.js können bei den gegebenen Webseiten heruntergeladen werden (siehe Fußzeile) / Bei Linux können Installationen über das Terminal (Kommandozeile oder Konsole) geregelt werden, da Installationspakete vorliegen. Git: „sudo apt install git“, Git LFS: „sudo apt install git-lfs“ und Node.js „sudo apt install nodejs“. Für die richtige Unity Version muss UnityHub<sup>17</sup> (Versionsverwaltung für Unity) heruntergeladen werden und installiert werden / bei Linux wird eine Datei namens „UnityHub.AppImage“ heruntergeladen, welche mit dem Befehl „sudo chmod -x UnityHub.AppImage“ ausführbar wird. Danach ist die Version 2021.1.5.f1<sup>18</sup> von Unity zu installieren. Bei Windows reicht es, wenn auf den Link „Install this version with Unity Hub“ geklickt wird / bei Linux funktioniert der Link nicht und es muss im Terminal der Pfad zu der UnityHub.AppImage Datei und der Link angegeben werden z.B. „Pfad/Downloads/UnityHub.AppImage unityhub://2021.1.5f1/3737af19df53“. Nachdem die 2021.1.5.f1 Version installiert wurde kann ein neues 3D Projekt erstellt werden. Für die richtige Szene muss oben unter File - Open Scene (oder Strg + O) eine Szene ausgewählt werden. Die verschiedenen Szenen befinden sich im heruntergeladenen/geklonten Git Verzeichnis „Pfad\vr-klassenzimmer\Assets\Scenes\ChemistryRoom.unity“. In Unity kann anschließend auf den Play-Knopf gedrückt werden und die Szene beginnt (siehe Abbildung 7 (S.16)).

Für die Webanwendung muss in den geklonten Ordner navigiert werden: „Pfad\vr-klassenzimmer\website-control~“. Dort muss die Datei „package-lock.json“ entfernt oder umbenannt werden (entweder fehlerhaft oder gerade nur bei meiner Version, auch bei Linux). Nun muss in diesem Verzeichnis bei Windows in cmd (Eingabeaufforderung) und bei Linux im Terminal der Befehl „npm i“ ausgeführt werden (nur bei erstmaliger Nutzung). Dieser liest und installiert alle benötigten Abhängigkeiten bzw. Software, aus der „package.json“ Datei. Wenn bei Unity der Play-Knopf gedrückt wurde, wird ein Localhost erstellt (Netzwerkschnittstelle zwischen Unity und Webanwendung, ). Danach kann in der cmd bzw. Terminal der Befehl „npm start“ oder die „startCoach.sh“ ausgeführt werden (bei Linux Datei erneut ausführbar machen mit chmod). Dadurch wird ein weiterer Localhost erstellt, welcher mit dem ersten Localhost kommuniziert. Es wird auch automatisch eine Webseite geöffnet, wo durch Anklicken die Studenten oder Szenarios ausgewählt werden können.

---

<sup>17</sup><https://unity3d.com/de/get-unity/download>

<sup>18</sup><https://unity3d.com/de/unity/whats-new/2021.1.5> Abrufdatum: 02.07.2021

## 11 Anhang

	In derselben Klasse	In der Unterklasse	Im selben Package	Überall
<b>private</b>	Sichtbar			
<b>protected</b>	Sichtbar	Sichtbar	Sichtbar	
<b>public</b>	Sichtbar	Sichtbar	Sichtbar	Sichtbar
<b>Nichts angeben</b>	Sichtbar		Sichtbar	

Tabelle 2: Datenkapselung - Einstellmöglichkeiten für die Zugriffsrechte

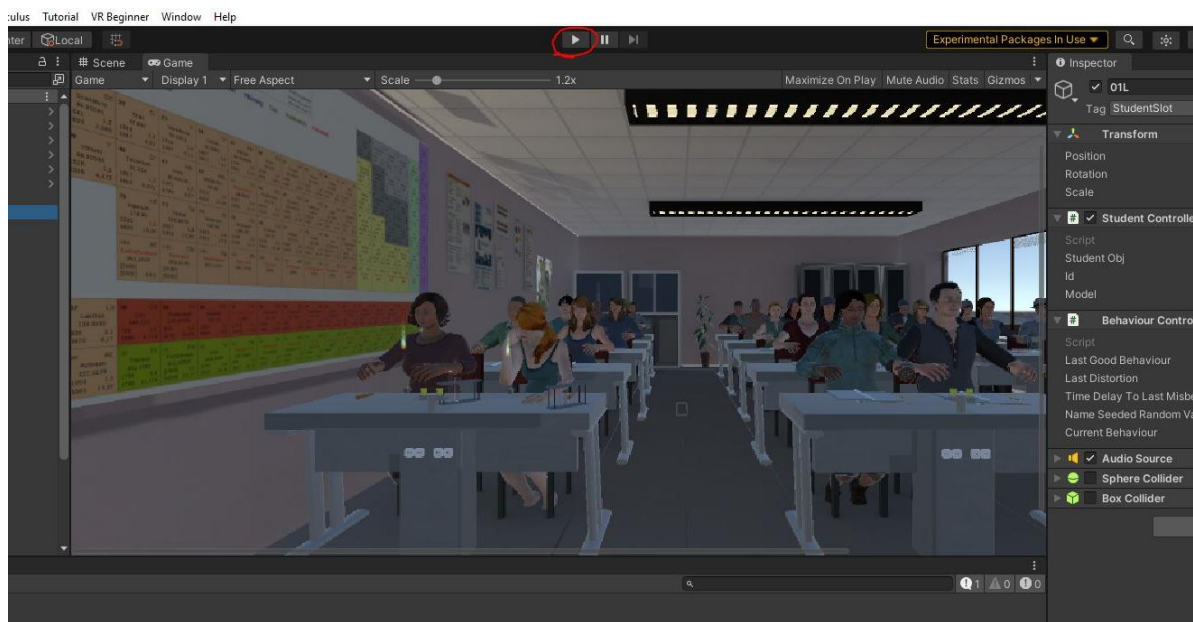


Abbildung 6: Unity-Interface mit ChemistryRoom, Play-Button (rot)

### Good behaviours

Alle abwählen

Alle auswählen

Idle Writing RaiseArm AskQuestion PlakatPartner

### Bad behaviours






Eating Drinking Playing Staring Tapping Hitting Throwing Chatting

### Impulse

positive neutral negative evoke

Abbildung 7: Einordnung der Verhaltensweisen in der Webanwendung. Legende: Good behaviours (Mitarbeit/gutes Verhalten), Bad behaviours (Störung, schlechtes Verhalten), Impuls (Interaktion von LuL mit vSoS) (positiv, neutrag, negativ, rufen) Einstufung in grün, gelb und rot

11 Anhang

Name	Übersetzung	Erklärung	Einordnung	Darstellung der Verhaltensweisen durch virtuellen Schüler (Haralf)
Idle	Untätig/ inaktiv	vSoS macht keine aktiven Handlungen	keine	
Writing	Schreiben	vSoS schreibt in ihr Heft	Good behaviours (grün)	
AskQuestion	Eine Frage stellen	vSoS stellt eine Frage mit Audio: „Können wir heute früher Pause machen?“	Good behaviours (grün)	
RaiseArm	Arm heben	vSoS hebt den Arm	Good behaviours (grün)	
PlakatPartner	Zusammenarbeit	Hier in Chemieraum mit Partner etwas in Reagenzgläser (fehlen) gießen	Good behaviours (grün)	
Impulse	Impuls	Interaktion von LuL und vSoS z.B. bei Störung wird das Verhalten wieder auf Idle gesetzt.	keine	kein



11 Anhang








Eating	Essen	vSoS isst etwas z.B. einen Apfel	Bad behaviours (gelb)		
Drinking	Etwas Trinken	vSoS trinkt etwas aus einer Dose	Bad behaviours (gelb)		
Staring	Starren	vSoS starrt auf den Boden	Bad behaviours (gelb)		
Playing	Spielen	vSoS spielt mit Stift und Lineal	Bad behaviours (gelb)		
Tapping	Antippen?	vSoS tippt Mitschüler an?	Bad behaviours (gelb)	Keine Handlung nach Ausführung	
Hitting	Schlagen	Zwei vSoS schlagen sich (hier mit Ellbogen)	Bad behaviours (rot)		
Chatting	Plaudern/quatschen	Zwei vSoS reden miteinander über Freizeithemen	Bad behaviours (rot)		
Throwing	Werfen	vSoS wirft einen Papierball in Richtung der LuL	Bad behaviours (rot)	Anzeige fehler (Apfel noch in der Hand)	

Tabelle 3: Übersicht aller Verhalten (Behaviour) mit Erklärung, Übersetzung, Einordnung und Darstellung

## 11 Anhang

```
//_pairActions = { "Hitting", "Chatting", "PlakatPartner" }; can lead to logic problems (do not select two students next to each other)
public static readonly string[] _classClown = { "Eating", "Drinking", "Staring", "Playing", "Tapping", "Hitting", "Chatting", "Throwing" };
public static readonly string[] _nerd = { "Writing", "RaiseArm", "PlakatPartner", "AskQuestion" };
```

Abbildung 8: Array der Verhalten für Klassenclown und Streber, Kommentar Partneraktionen (\_pairActions) kann Probleme aufrufen

```
//select behaviour
public bool IsclassClown;
public bool IsNerd;
```

Abbildung 9: boolesche Variablen: IsclassClown, IsNerd

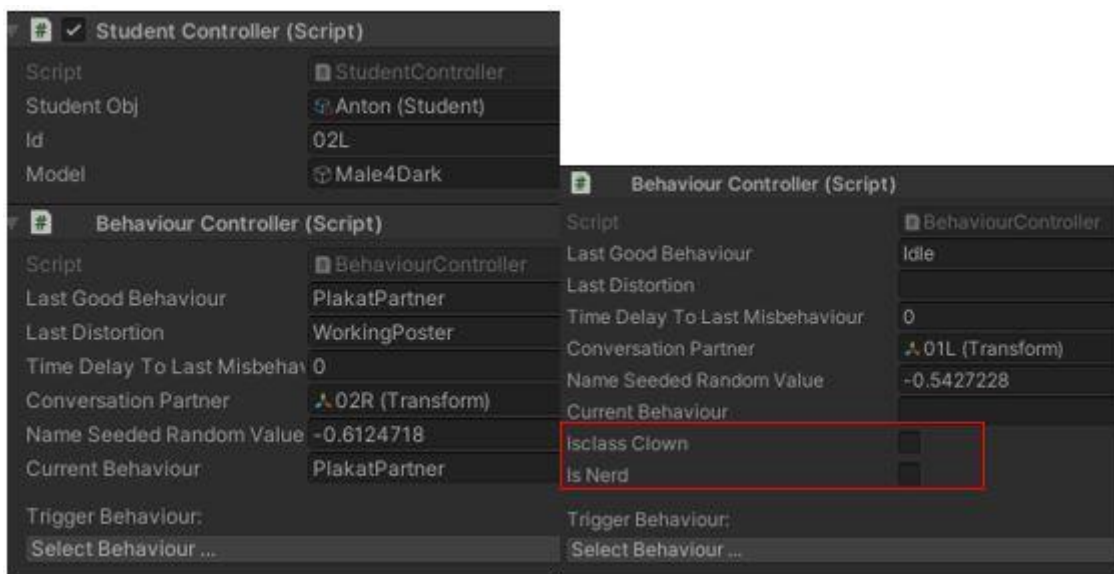


Abbildung 10: Seitenpanel, Änderung (rot) nach Einfügen von IsclassClown und IsNerd

```
private IEnumerator NonScriptedClassBehaviour()
{
    int levelOfDisturbance = 0;

    while (true)
    {
        if (ReplayController.isLoading)
        {
            // if a replay is being loaded, do not trigger random disturbance but only replay these that were recorded
        }
        else if (ChanceToMisbehave < Random.value)
        {
            ClassController.SetRandomDisturbance(gameObject, levelOfDisturbance);
            levelOfDisturbance++;
        }
        else
        {
            ChanceToMisbehave += 0.004f;
        }

        yield return new WaitForSeconds(10);
    }
}
```

Abbildung 11: NonScriptedClassBehaviour, bereits implementierte Funktion mit kleinen Fehler ("<" muss ">" sein)

## 11 Anhang

```
private IEnumerator isclassClownBehaviour()
{
    int levelOfDisturbance = 0;

    while (true)
    {
        if (ReplayController.isLoading)
        {
            // if a replay is being loaded, do not trigger random disturbance but only replay these that were recorded
        }
        else if (ChanceToMisbehave > Random.value)
        {
            System.Random rnd = new System.Random();
            int randClown = rnd.Next(0, SpecialBehaviours._classClown.Length);
            var behaviourClassclown = SpecialBehaviours._classClown[randClown];
            HandleBehaviour(behaviourClassclown);
            //Console.WriteLine("Setting behaviour to: {0}" + behaviourKlassenc clown);
            levelOfDisturbance++;
            ChanceToMisbehave = 0;
        }
        else
        {
            ChanceToMisbehave += 0.004f;
        }

        yield return new WaitForSeconds(10);
    }
}
```

Abbildung 12: Schnittstelle isclassClownBehaviour(), while Schleife mit Übergabe der Verhaltensweisen an die Funktion HandleBehaviour() falls wahr (falls Schüler ausgewählt)

```
private IEnumerator isNerdBehaviour()
{
    int levelOfDisturbance = 0;

    while (true)
    {
        if (ReplayController.isLoading)
        {
            // if a replay is being loaded, do not trigger random disturbance but only replay these that were recorded
        }
        else if (ChanceToMisbehave > Random.value)
        {
            System.Random rnd = new System.Random();
            int randNerd = rnd.Next(0, SpecialBehaviours._nerd.Length);
            var behaviourNerd = SpecialBehaviours._nerd[randNerd];
            HandleBehaviour(behaviourNerd);
            //Console.WriteLine("Setting behaviour to: {0}" + behaviourNerd);
            levelOfDisturbance++;
            ChanceToMisbehave = 0;
        }
        else
        {
            ChanceToMisbehave += 0.004f;
        }

        yield return new WaitForSeconds(10);
    }
}
```

Abbildung 13: Schnittstelle isNerdBehaviour(), while Schleife mit Übergabe der Verhaltensweisen an die Funktion HandleBehaviour() falls wahr (falls Schüler ausgewählt)