



School of  
Engineering

## **Bachelorarbeit (Informatik)**

Programmierung einer Virtual Reality  
Applikation mit Unity für Oculus Quest 2

---

**Autoren** Yanick Sebastian Senn  
Rhiana Weber

---

**Hauptbetreuung** Dr. Reto Knaack

---

**Datum** 08.06.2022

---

## **Erklärung betreffend das selbstständige Verfassen einer Bachelorarbeit an der School of Engineering**

### **Erklärung betreffend das selbstständige Verfassen einer Bachelorarbeit an der School of Engineering**

Mit der Abgabe dieser Bachelorarbeit versichert der/die Studierende, dass er/sie die Arbeit selbstständig und ohne fremde Hilfe verfasst hat. (Bei Gruppenarbeiten gelten die Leistungen der übrigen Gruppenmitglieder nicht als fremde Hilfe.)

Der/die unterzeichnende Studierende erklärt, dass alle zitierten Quellen (auch Internetseiten) im Text oder Anhang korrekt nachgewiesen sind, d.h. dass die Bachelorarbeit keine Plagiate enthält, also keine Teile, die teilweise oder vollständig aus einem fremden Text oder einer fremden Arbeit unter Vorgabe der eigenen Urheberschaft bzw. ohne Quellenangabe übernommen worden sind.

Bei Verfehlungen aller Art treten die Paragraphen 39 und 40 (Unredlichkeit und Verfahren bei Unredlichkeit) der ZHAW Prüfungsordnung sowie die Bestimmungen der Disziplinarmassnahmen der Hochschulordnung in Kraft.

**Ort, Datum:**

Schaffhausen, 08.06.2022

Schaffhausen, 08.06.2022

**Name Studierende:**

Yanick Sebastian Senn

Rhiana Weber

## Zusammenfassung

Im Rahmen dieser Arbeit wird ein Serious Game mit der Unity Game-Engine für die Oculus Quest 2 entwickelt. Bei dem Serious Game handelt es sich um ein Schaltlogik-Spiel basierend auf Wahrheitstabellen verschiedener Logik-Gatter.

Für das Serious Game wurde in einem ersten Schritt ein didaktisches Konzept entwickelt, da das Serious Game zur Verwendung im Hochschulunterricht angedacht ist. Das Serious Game wurde danach anhand des Konzeptes aufgebaut, um die definierten Lerninhalte in VR-Form wiederzugeben.

Anhand der Recherche bereits existierender Lösungen wurde ersichtlich, dass es bereits ein paar auf Logikgattern basierende Spiele gibt, doch werden diese vermehrt im Web oder lokal auf dem Computer gespielt. Das entwickelte Serious Game hebt sich von den existierenden Lösungen sofern ab, dass es interaktiv mit den Händen sowie mit einer realistischen physikalisch-mathematischen Grundlage gespielt wird und somit auch einen anderen Bezug zum Lerninhalt bietet.

Das fertige Serious Game besteht aus drei Spielmodi; Tutorial, Hauptspiel und Sandbox. Als Einstiegspunkt für den Spieler dient ein Hauptmenü mit Navigation zu allen drei Spielmodi. Im freiwilligen Tutorial erlernt der Spieler, in einer visuell geführten Form alle notwendigen Interaktionen, die fürs Hauptspiel nötig sind.

Der Spieler spielt sich im Hauptspiel Schritt für Schritt durch die verschiedenen Arten von Logikgattern durch, indem er aus den bereits vorhandenen Gattern ein neues Gatter auf einem Spielbrett, welches mit Kabeln, Glühbirnen und Stromschaltern ausgestattet ist, erbaut. Er beginnt mit dem NAND-Gatter, aus diesem baut er ein NOT-, danach folgen AND-, OR-, XOR-Gatter, HALFADDER(Halbaddierer) bis und mit FULLADDER(Volladdierer). Um den Spieler zu motivieren, ist das Serious Game mit einem zeitbasierten Highscore-System ausgestattet. Weißt der Spieler bei einem Schritt nicht mehr weiter, kann er mit Punkten Hinweise und Lösungen beziehen.

In der Sandbox sind alle oben erwähnten Gatter schon vorhanden und der Spieler kann frei eigene Kreationen und Verbindungen, ohne Zeitdruck, auf dem Board erstellen und ausprobieren.

## Abstract

In this thesis, a Serious Game is being developed using the Unity Game Engine for the Oculus Quest 2. The Serious Game is a switch logic game based on truth tables of different logic gates.

As a first step, a didactic concept was developed for this Serious Game since the game is intended for use in university teaching. The Serious Game was then built based on the didactic concept in order to reproduce the defined learning content in VR form.

Based on the research of already existing solutions, it became apparent that a few logic gate-based games already exist, but they are increasingly played on the web or locally on the computer. The developed Serious Game differs from the existing solutions. It is played interactively with the hands and with a realistic physical-mathematical basis and thus offers a different experience regarding the learning content.

The finished Serious Game consists of three game modes: tutorial, main game, and sandbox. The main menu with navigation to all three game modes serves as the entry point for the player.

In the voluntary tutorial, the player learns, in a visually guided form, all the necessary interactions required for the main game.

In the main game, the player plays through the different gates step by step by building a new gate from the existing gates on a board equipped with cables, light bulbs, and power switches. He starts with the NAND gate, from this he builds a NOT, followed by AND, OR, XOR, HALFADDER up to and including FULLADDER. To motivate the player, the serious game is equipped with a time-based high score system. If the player gets stuck, he can use points to obtain hints and solutions.

In the sandbox, all the gates mentioned above are already present and the player can freely create and try out his own creations and connections, without time pressure, on the board.

## Abkürzungen

3D	Drei-Dimensional
API	Application Programming Interface (Programmierschnittstelle)
CTIT	Computertechnik für Informatik (Modul an der ZHAW)
LED	Light-emitting diode
SDK	Software Development Kit
VR	Virtual Reality
ZHAW	Zürcher Hochschule für Angewandte Wissenschaften

Tabelle 1: Tabelle mit Abkürzungen

## Symbole

.	NAND Operator
$\neg$	NOT Operator
$\wedge$	AND Operator
$\vee$	OR Operator
$>=$	Grösser-oder-gleich-Zeichen

Tabelle 2: Tabelle mit Symbolen

## Glossar

Agile	Ein Softwareentwicklungsprozess welcher den Fokus auf transparente und agile Veränderungen an Software legt. [1]
C#, C Sharp	Eine Programmiersprache welche von Microsoft entwickelt wird [2].
Crunch	Entstehen wenn Arbeit nicht korrekt geschätzt wurde und dadurch mehr Arbeit in knapper Zeit erledigt, werden muss. Tritt normalerweise gegen Ende eines Sprints auf.
Daily-Standup	Ein tägliches Meeting in dem Team, um die anstehende Arbeit zu besprechen. Es ermöglicht dem Team Probleme oder Crunches frühzeitig zu erkennen und zu behandeln.
Epic	Eine thematische Zuordnung für Stories und Tasks. Kann auch als Meilenstein bezeichnet werden.
Game-Engine	Ein Software-Framework das dem Entwickler Möglichkeiten zur Programmierung, Interaktion und Gestaltung von Spielen ermöglicht. [3]
IEC 60617-12	Die Norm der Abhängigkeitsnotation der graphischen Symbole für Schaltpläne. [4]
JIRA	Unterstützende Software von Atlassian zur Durchführung von Agilen Softwareprojekten. [5]
SCRUM	Ist ein weit verbreitetes Projekt Management Framework für die Agile Softwareentwicklung.
Spawnen	Zur Laufzeit eines Spiels ein bestimmtes Objekt erstellen. Beispielsweise wenn eine Spielfigur ein Projektil aus einer Waffe schießt.
Sprint	Repräsentiert eine Zeit Box mit klar festgelegter Dauer (Normalerweise 2 Wochen) um bestimmte Task oder Stories zu erledigen. Es sollte generell nicht mehr Arbeit in einen Sprint gepackt werden als auch vom Team realisierbar ist.
Story	Eine Gruppierung von einem oder mehreren Tasks.
Task	Eine atomare Aufgabe. Es ist möglich Tasks eigenständig in einem Meilenstein gruppiert in Stories zu erfassen.
Unity	Eine Game-Engine von «Unity Technologies» welche die C# Scripting API

	verwendet. Die Engine verfügt über grosse Beliebtheit aufgrund der einfachen Bedienung und der breiten Kompatibilität zu PCs und Konsolen. [6] [7]
Virtual Reality	Die Virtual Reality beschreibt eine Simulation der realen oder einer willkürlichen Welt. Diese Simulationen können durch das Tragen eines Virtual Reality Headset erlebt werden. [8] Alternativ gibt es auch Räume, welche die Virtual Reality projizieren können. [9]
Virtual Reality Headset	Ein Virtual Reality Headset ist ein Gerät, ähnlich einer Brille, welche über den Kopf gezogen werden kann. Die Linsen in dem Headset projizieren den aktuellen Blickwinkel des Benutzers. Zusätzlich verfügen die meisten Headsets über zwei Controller (Ein Controller pro Hand) damit Interaktionen mit dem Benutzer möglich sind. [10]

*Tabelle 3: Tabelle mit Glossar.*

# Inhaltsverzeichnis

<i>Zusammenfassung</i> .....	2
<i>Abstract</i> .....	3
<i>Abkürzungen</i> .....	4
<i>Symbole</i> .....	4
<i>Glossar</i> .....	5
<i>Inhaltsverzeichnis</i> .....	7
<b>1 Einleitung</b> .....	<b>13</b>
<b>1.1 Virtual Reality</b> .....	<b>13</b>
<b>1.2 Definition Virtual Reality</b> .....	<b>13</b>
<b>1.3 Serious Game</b> .....	<b>13</b>
<b>1.4 Spielidee</b> .....	<b>14</b>
1.4.1 Inspiration .....	14
1.4.2 Idee .....	14
1.4.2.1 Design Ideen .....	15
1.4.3 Warum mit VR? .....	16
<b>1.5 Zielsetzung</b> .....	<b>16</b>
<b>1.6 Oculus Quest 2</b> .....	<b>17</b>
1.6.1 Spezifikationen der Oculus Quest 2 .....	17
1.6.2 Controller Mapping der Oculus Quest 2 Controller .....	18
<b>2 Ausgangslage</b> .....	<b>19</b>
<b>2.1 Existierende Lösungen</b> .....	<b>19</b>
2.1.1 Please, Don't Touch Anything 3D .....	19
2.1.2 Nandgame .....	20
2.1.3 Logic World .....	21
2.1.4 Minecraft .....	22
2.1.5 Human Resource Machine .....	23
<b>3 Theoretische Grundlagen</b> .....	<b>24</b>
<b>3.1 Logische Verknüpfungen</b> .....	<b>24</b>
3.1.1 NAND .....	24
3.1.2 NOT .....	24
3.1.3 AND .....	25
3.1.4 OR .....	25
3.1.5 XOR .....	25
3.1.6 Binäre Addition .....	26
3.1.6.1 HALFADDER .....	26
3.1.6.2 FULLADDER .....	27
<b>4 Didaktisches Konzept</b> .....	<b>28</b>
<b>4.1 Lernziele</b> .....	<b>28</b>
<b>4.2 Vorwissen</b> .....	<b>28</b>
<b>4.3 Einführung</b> .....	<b>28</b>
<b>4.4 Aufgabe</b> .....	<b>28</b>

<b>4.5 Prüfung.....</b>	<b>28</b>
<b>5 Anforderungsanalyse.....</b>	<b>30</b>
<b>5.1 Aufbau der Tabellen .....</b>	<b>31</b>
5.1.1 Aufbau Use-Case Tabelle .....	31
5.1.2 Aufbau funktionale Anforderungstabelle.....	31
<b>5.2 Grundlegende Vorbedingungen.....</b>	<b>32</b>
<b>5.3 Hauptmenü Use-Cases .....</b>	<b>32</b>
5.3.1 Spiel Starten.....	32
5.3.1.1 Spiel Szene ist geladen.....	32
5.3.2 Tutorial Starten .....	33
5.3.2.1 Tutorial Szene ist geladen .....	33
5.3.3 Sandbox Starten .....	33
5.3.3.1 Sandbox Szene ist geladen .....	34
<b>5.4 In-Game Menü.....</b>	<b>35</b>
5.4.1 Menü aufrufen .....	35
5.4.1.1 In-Game Menü wird angezeigt.....	35
5.4.1.2 In-Game Menü kann bedient werden .....	36
5.4.2 Menü verstecken .....	36
5.4.2.1 Menü wird nicht mehr angezeigt.....	36
5.4.3 Spielszene neu laden.....	37
5.4.3.1 Spiel Szene wird neu geladen .....	37
5.4.4 Spielszene verlassen .....	37
5.4.4.1 Spiel Szene ist verlassen .....	38
<b>5.5 Headset- und Controllerbedienung .....</b>	<b>39</b>
5.5.1 Headset bewegen .....	39
5.5.1.1 Perspektive wird ausgerichtet .....	39
5.5.2 Controller bewegen.....	40
5.5.2.1 Virtuelle Hände werden relativ zum Controller bewegt.....	40
5.5.2.2 Virtuelle Hände werden durch Controller geschlossen .....	40
5.5.2.3 Hände werden durch Controller geöffnet .....	41
5.5.3 Im Raum bewegen .....	41
5.5.3.1 Spieler an beliebigen freien Ort im Raum teleportieren.....	42
<b>5.6 Gatter.....</b>	<b>43</b>
5.6.1 Gatter halten.....	43
5.6.1.1 Gatter wird gehalten.....	43
5.6.1.2 Gatter wird losgelassen .....	43
5.6.2 Gatter platzieren.....	44
5.6.2.1 Gatter wird auf freiem Socket platziert .....	44
5.6.3 Gatter entfernen .....	45
5.6.3.1 Gatter wird vom Socket entfernt .....	45
5.6.4 Gatter löschen .....	45
5.6.4.1 Gatter wird gelöscht .....	46
<b>5.7 Kabel.....</b>	<b>47</b>
5.7.1 Kabel halten .....	47
5.7.1.1 Kabel wird am Kabel-Handle gehalten .....	47
5.7.1.2 Kabel wird in der Luft losgelassen .....	48
5.7.2 Kabel verbinden.....	48
5.7.2.1 Kabel wird auf freiem Kabel-Eingang platziert .....	49
5.7.3 Kabel entfernen.....	49
5.7.3.1 Verbundenes Kabel wird entfernt .....	49
<b>5.8 Stromschalter.....</b>	<b>50</b>
5.8.1 Strom einschalten.....	50
5.8.1.1 Schalter wird von 0 auf 1 gekippt .....	50

5.8.1.2	Strom auf verbundene Objekte weitergeben .....	50
5.8.2	Strom ausschalten .....	51
5.8.2.1	Schalter wird von 1 auf 0 gekippt .....	51
5.8.2.2	Strom auf verbundene Objekte nicht weitergeben .....	51
<b>5.9</b>	<b>Notizen.....</b>	<b>52</b>
5.9.1	Kreide halten.....	52
5.9.1.1	Kreide wird in der Hand gehalten .....	52
5.9.1.2	Kreide wird losgelassen .....	53
5.9.2	Striche mit Kreide platzieren .....	53
5.9.2.1	Kreide platziert Striche zwischen Berührungs punkten mit Wandtafel.....	54
5.9.3	Schwamm halten.....	54
5.9.3.1	Schwamm wird gehalten.....	55
5.9.3.2	Schwamm wird losgelassen .....	55
5.9.4	Striche mit Schwamm entfernen .....	56
5.9.4.1	Schwamm entfernt Striche zwischen Berührungs punkten mit Wandtafel .....	56
<b>5.10</b>	<b>Board .....</b>	<b>57</b>
5.10.1	Board leeren .....	57
5.10.1.1	Board wird geleert.....	57
5.10.2	Pseudo-Gatter erstellen.....	58
5.10.2.1	Pseudo-Gatter ist erstellt .....	58
5.10.3	Label für Pseudo-Gatter bestimmen .....	59
5.10.3.1	Gatter-Label eines Pseudo-Gatter wird durch Pfeiltasten ausgewählt .....	59
	Gatter-Label eines Pseudo-Gatter wird durch Pfeiltasten.....	59
	ausgewählt .....	59
<b>5.11</b>	<b>Wandtafel Use-Cases.....</b>	<b>60</b>
5.11.1	Aufgabe ablesen .....	60
5.11.1.1	Aktuelle Aufgabe wird auf der Wandtafel angezeigt .....	60
	Aktuelle Aufgabe wird auf der Wandtafel angezeigt .....	60
5.11.2	Aufgabe überprüfen.....	61
5.11.2.1	Check-Button wird angezeigt.....	61
	Check-Button wird angezeigt .....	61
5.11.2.2	Falsch gelöste Aufgabe .....	61
	Falsch gelöste Aufgabe.....	61
5.11.2.3	Korrekt gelöste Aufgabe .....	62
	Korrekt gelöste Aufgabe.....	62
5.11.3	Navigieren .....	63
5.11.3.1	Weiter navigieren .....	63
	Weiter navigieren .....	63
5.11.3.2	Zurück navigieren .....	63
	Zurück navigieren .....	63
5.11.4	Hinweis / Lösung beziehen .....	64
5.11.4.1	Kosten von Score abziehen .....	64
	Kosten von Score abziehen .....	64
5.11.4.2	Nächsten Hinweis / Lösung anzeigen .....	64
	Nächsten Hinweis / Lösung anzeigen .....	64
5.11.4.3	Bezahlte Hinweise / Lösungen nicht erneut bezahlen .....	65
	Bezahlte Hinweise / Lösungen nicht erneut bezahlen .....	65
<b>5.12</b>	<b>Automat.....</b>	<b>66</b>
5.12.1	Gatter auswählen .....	66
5.12.1.1	Vorwärts navigieren.....	66
	Vorwärts navigieren .....	66
5.12.1.2	Rückwärts navigieren.....	67
	Rückwärts navigieren .....	67
5.12.2	Gatter beziehen.....	68
5.12.2.1	Neue Gatter-Instanz erscheint .....	68
	Neue Gatter-Instanz erscheint .....	68
<b>5.13</b>	<b>High-Score.....</b>	<b>69</b>

5.13.1	High-Score ablesen.....	69
5.13.1.1	High-Score wird auf Tisch angezeigt.....	69
	High-Score wird auf Tisch angezeigt .....	69
5.13.2	Aktueller Score ablesen.....	69
5.13.2.1	Abziehen der Punkte pro Sekunde .....	70
	Abziehen der Punkte pro Sekunde.....	70
<b>6</b>	<b>Vorgehen / Methoden .....</b>	<b>71</b>
<b>6.1</b>	<b>Projektmanagement.....</b>	<b>71</b>
6.1.1	Agile / SCRUM .....	71
6.1.2	Projektplan.....	71
6.1.3	Meilensteine.....	71
6.1.4	Sprint .....	72
6.1.5	Daily-Standup .....	72
6.1.6	Weekly Check-In .....	72
6.1.7	SCRUM / Sprint Umsetzung in diesem Projekt.....	72
<b>7</b>	<b>Architektur .....</b>	<b>73</b>
<b>7.1</b>	<b>Basics .....</b>	<b>73</b>
7.1.1	Prefabs .....	73
7.1.2	MonoBehaviour .....	74
7.1.3	ScriptableObject .....	74
7.1.4	Unity-Lifecycle.....	74
7.1.5	Collider .....	75
7.1.6	Event-Driven Architecture (EDA) .....	76
<b>7.2</b>	<b>Headset- und Controllerinteraktionen .....</b>	<b>77</b>
7.2.1	Use-Cases .....	77
<b>7.3</b>	<b>Grabbable .....</b>	<b>78</b>
7.3.1	Events .....	78
<b>7.4</b>	<b>Snapzone .....</b>	<b>78</b>
7.4.1	Events .....	80
7.4.2	Klassendiagramm der Snapzone .....	80
7.4.3	Use-Cases .....	81
<b>7.5</b>	<b>Gatter.....</b>	<b>82</b>
7.5.1	Ausgänge .....	82
7.5.2	Gatter-Logik .....	82
7.5.2.1	Klassendiagramm zur Gatterlogik.....	83
7.5.3	Use-Cases .....	83
<b>7.6</b>	<b>Kabel.....</b>	<b>84</b>
7.6.1	Manuelles entfernen.....	84
7.6.2	Automatisches entfernen.....	85
7.6.3	Klassendiagramm des Kabels .....	86
7.6.4	Use-Cases .....	86
<b>7.7</b>	<b>Präsentation .....</b>	<b>87</b>
7.7.1	Slide .....	87
7.7.2	Präsentation.....	89
7.7.3	Presenter Content.....	89
7.7.4	Presenter .....	91
7.7.5	Use-Cases .....	92
<b>7.8</b>	<b>Stromfluss .....</b>	<b>93</b>
7.8.1	Stromfluss des Gatters .....	94
7.8.1.1	Eingehende Events .....	94
7.8.1.2	Swimlane-Diagramm .....	95
7.8.2	Kabel.....	96

7.8.2.1	Eingehende Events .....	96
7.8.2.2	Swimlane-Diagramm .....	96
7.8.3	EnergySource .....	97
7.8.3.1	Eingehende Events .....	97
7.8.3.2	Swimlane-Diagramm .....	97
7.8.4	EnergyDestination .....	97
7.8.4.1	Eingehende Events .....	97
7.8.4.2	Swimlane-Diagramm .....	98
7.8.5	Ungültige Konstellationen .....	98
7.8.5.1	Unplatziertes Gatter .....	98
7.8.5.2	Zyklus .....	99
7.8.6	Use-Cases .....	99
<b>7.9</b>	<b>Gatter Dispenser.....</b>	<b>100</b>
7.9.1	Klassendiagramm .....	100
7.9.2	Toggle .....	101
7.9.2.1	Statusdiagramm .....	101
7.9.2.2	Klassendiagramm .....	102
7.9.3	Button .....	103
7.9.3.1	Statusdiagramm .....	103
7.9.3.2	Klassendiagramm .....	103
<b>8</b>	<b>Entwicklungsentscheidungen.....</b>	<b>104</b>
<b>8.1</b>	<b>Controller vs. Handtracking? .....</b>	<b>104</b>
<b>8.2</b>	<b>Spielsprache? .....</b>	<b>104</b>
<b>8.3</b>	<b>XR Interaction Toolkit vs. Oculus Integration SDK? .....</b>	<b>105</b>
<b>8.4</b>	<b>Single-Player vs. Multi-Player? .....</b>	<b>105</b>
<b>9</b>	<b>Resultate .....</b>	<b>106</b>
<b>9.1</b>	<b>Spielkomponenten .....</b>	<b>106</b>
9.1.1	Kabel .....	106
9.1.2	Gatter .....	107
9.1.3	Board .....	109
9.1.4	Automat .....	111
9.1.5	Wandtafel .....	112
9.1.6	In-Game Menü .....	113
<b>9.2</b>	<b>Szenen .....</b>	<b>114</b>
9.2.1	Design der Szenen .....	114
9.2.2	Hauptmenü-Szene .....	114
9.2.3	Tutorial-Szene .....	115
9.2.4	Spiel-Szene .....	115
9.2.4.1	Spiel Ablauf in der Spiel-Szene .....	116
9.2.4.2	Punkte / Highscore in der Spielszene .....	117
9.2.5	Sandbox-Szene .....	119
<b>9.3</b>	<b>Akustik .....</b>	<b>120</b>
<b>9.4</b>	<b>Gestellte Aufgaben und Lösungen .....</b>	<b>121</b>
9.4.1	NOT Aufgabe .....	121
9.4.2	AND Aufgabe .....	121
9.4.3	OR Aufgabe .....	122
9.4.4	XOR Aufgabe .....	122
9.4.5	Halbaddierer Aufgabe .....	122
9.4.6	Volladdierer Aufgabe .....	123
<b>10</b>	<b>Diskussion .....</b>	<b>124</b>
<b>10.1</b>	<b>Oculus Integration.....</b>	<b>124</b>

10.1.1	Oculus Integration Bugs .....	124
<b>10.2</b>	<b>Debugging .....</b>	<b>125</b>
<b>10.3</b>	<b>Snapzone .....</b>	<b>125</b>
<b>10.4</b>	<b>Löschen eines Gatters .....</b>	<b>125</b>
<b>10.5</b>	<b>Strom Rekursiv berechnen .....</b>	<b>127</b>
<b>10.6</b>	<b>Event Driven Architecture .....</b>	<b>127</b>
10.6.1	Sound.....	127
10.6.2	Tutorial.....	128
10.6.3	Präsentation .....	128
10.6.4	Animationen als Eventhelper .....	128
<b>10.7</b>	<b>Projektablauf .....</b>	<b>130</b>
10.7.1	Pairprogramming.....	130
10.7.2	Spiel-Tests .....	130
10.7.3	Nicht entwickelte Use-Cases .....	130
10.7.3.1	Notizen .....	130
10.7.3.2	Board leeren .....	131
10.7.3.3	Pseudo Gatter .....	131
<b>11</b>	<b>Ausblick .....</b>	<b>132</b>
<b>11.1</b>	<b>Nicht entwickelte Use-Cases .....</b>	<b>132</b>
<b>11.2</b>	<b>Multiplayer .....</b>	<b>132</b>
<b>11.3</b>	<b>Erweiterung Aufgaben &amp; Komponenten .....</b>	<b>132</b>
<b>11.4</b>	<b>Gamification .....</b>	<b>132</b>
<b>12</b>	<b>Literaturverzeichnis .....</b>	<b>134</b>
<i>Abbildungsverzeichnis .....</i>		<i>136</i>
<i>Tabellenverzeichnis.....</i>		<i>139</i>
<i>Anhang .....</i>		<i>142</i>
<i>A.</i>	<i>Epic-Reports zum Projektablauf .....</i>	<i>142</i>
<i>a.</i>	<i>Konzeption.....</i>	<i>142</i>
<i>b.</i>	<i>Einarbeitung.....</i>	<i>143</i>
<i>c.</i>	<i>Infrastruktur .....</i>	<i>144</i>
<i>d.</i>	<i>Entwicklung Prototype.....</i>	<i>145</i>
<i>e.</i>	<i>Entwicklung Feinschliff.....</i>	<i>146</i>
<i>f.</i>	<i>Dokumentation .....</i>	<i>147</i>

# 1 Einleitung

In diesem Kapitel werden die grundlegenden Begriffe erklärt sowie die Spielidee, die Zielsetzung und die Oculus Quest 2 näher erläutert.

## 1.1 Virtual Reality

Virtual Reality ist eine Computersimulation, die zum Ziel hat, eine realistische virtuelle 3D Welt zu erzeugen. Um von einem Benutzer<sup>1</sup> als realistisch wahrgenommen zu werden, müssen folgende drei Punkte für die Simulation erfüllt sein [11]:

- Interaktivität: Die Welt soll höchst interaktiv sein, d.h. alles was man in der echten Welt manipulieren kann, soll auch in der virtuellen Welt zu einem gewissen Grad manipuliert werden können.
- Immersion (Eintauchen): Der Benutzer soll sich komplett von der virtuellen Welt umgeben fühlen. Die komplette Immersion kann anhand verschiedener Sinneswahrnehmungen wie Sehen, Hören, Fühlen, Schmecken, etc. herbeigeführt werden. In VR konzentriert man sich anhand des Stands der Technik auf visuelles, haptisches und tonales Feedback. Eine erfolgreiche Immersion veranlasst den Benutzer, die echte Welt für einen Moment zu vergessen.
- Präsenz in VR: „The sense of being there“ (Held & Durlach, Sheridan, Zeltzer, 1992). Die VR-Präsenz erfordert, dass der Benutzer sich als ein Teil der virtuellen Welt fühlt.

## 1.2 Definition Virtual Reality

Virtual Reality ist eine 3D Computersimulation, welche den Benutzer verhilft, eine immersive, interaktive computergenerierte synthetische Repräsentation einer realistischen oder imaginären Umgebung zu erleben. Die Umgebung reagiert in Echtzeit auf die Benutzerinteraktion durch verschiedene Sinneskanäle, mit dem Ziel in dem Benutzer ein Gefühl der Präsenz zu erzeugen. [11]

## 1.3 Serious Game

Ein Serious Game ist eine bestimmte Art von Spiel, das normalerweise in eine pädagogische Umgebung eingebettet ist. Das Serious Game hebt sich vom normalen Spiel insofern ab, dass sein Hauptfokus nicht auf Unterhaltung liegt, sondern auf Lernen, Lehren oder Beibringen verschiedener Kenntnisse. [12]

---

<sup>1</sup> In der folgenden Arbeit wird lesbarkeitshalber nur von Benutzer gesprochen. Dieser Term versteht sich als geschlechtsneutral und meint immer Benutzerin und Benutzer.

## 1.4 Spielidee

In diesem Abschnitt wird die Inspiration sowie die Idee hinter diesem Serious Game näher erläutert.

### 1.4.1 Inspiration

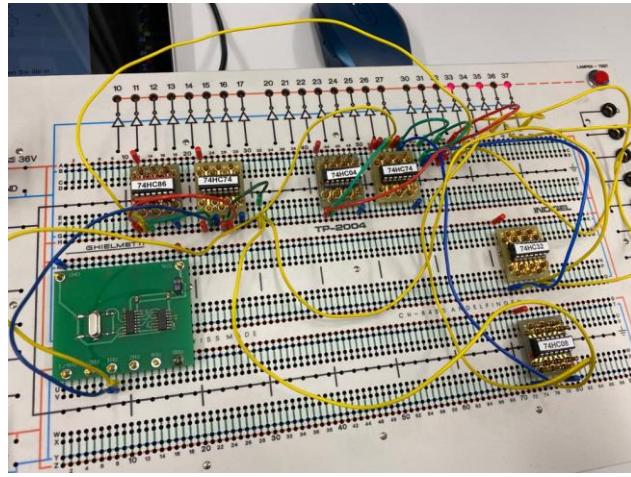


Abbildung 1: Indigel Board aus dem CTIT Unterricht

Die Inspiration zur Spielidee kam aus dem ZHAW Computertechnik für Informatik (CTIT) Praktikum zu den Gatterlogiken. In diesen Praktika wurde anhand von verschiedenen Bausteinen Halbaddierer, Volladdierer, Ampelsystem, etc. gebaut. Ein Beispiel für ein erstelltes Ampelsystem auf dem Indigel-Board im Praktikum ist in der Abbildung 1 erkennbar.

### 1.4.2 Idee

Die Idee ist es, anhand eines Startkomponenten, in unserem Fall ein NAND-Gatter, alle weiteren Gatter bis und mit Volladdierer auf einem Indigel-Board ähnlichen Board mit Glühbirnen, zu bauen.

Der Spieler<sup>2</sup> befindet sich in einem laborähnlichen Raum, mit einem Spielboard auf einem Tisch und die Aufgaben werden an einer Wandtafel angezeigt. Der Spieler baut Schritt für Schritt alle Gatter auf dem Board mit den vorher bereits erspielten Gattern zusammen. Der Spieler beginnt mit NAND dann NOT, AND, OR, XOR, Halbaddierer und schlussendlich erstellt er einen Volladdierer (Wahrheitstabellen zu den genannten Gattern sind im Abschnitt 3.1 Logische Verknüpfungen zu finden).

---

<sup>2</sup> In der folgenden Arbeit wird lesbarkeitshalber nur von Spieler gesprochen. Dieser Term versteht sich als geschlechtsneutral und meint immer Spielerin und Spieler.

#### 1.4.2.1 Design Ideen

In den untenstehenden Abbildungen kann man die Design Ideen für das Serious Game erkennen. Die *Abbildung 4* zeigt die Idee für den Raum mit der Wandtafel für die Aufgabenstellung. *Abbildung 2* und *Abbildung 3* zeigen die Paper Prototypen<sup>3</sup> für den Raum und das Board.

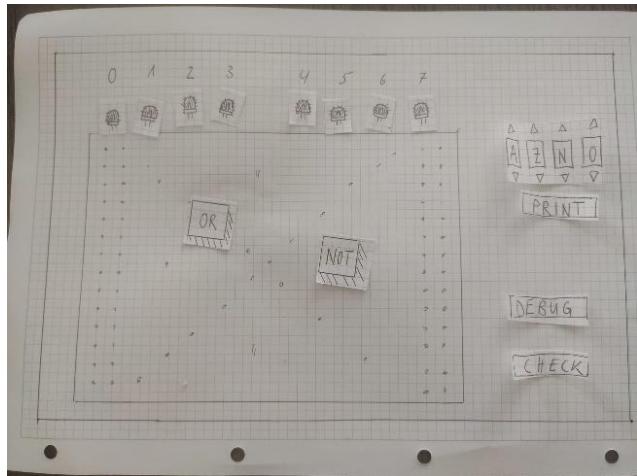


Abbildung 2: Prototyp des Boards

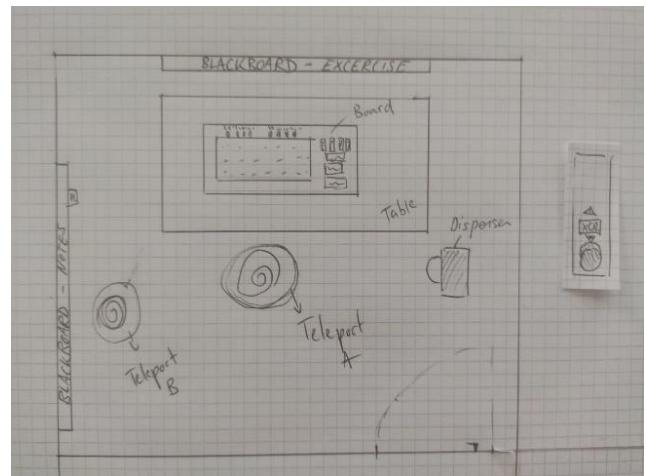


Abbildung 3: Prototyp des Raumes

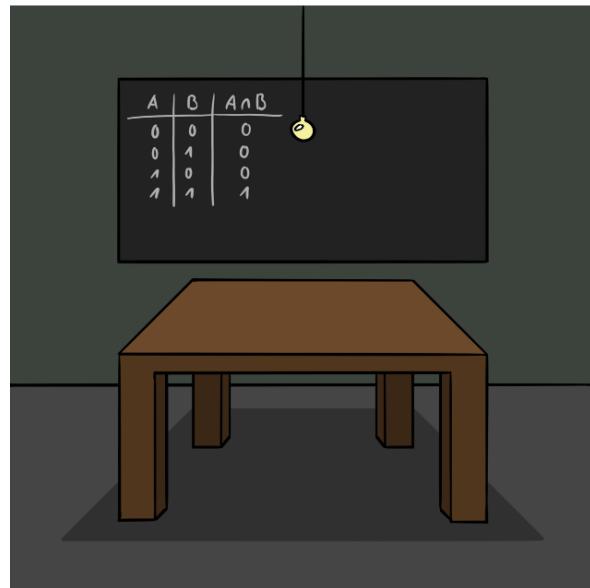


Abbildung 4: Sketch des Raumes aus der Perspektive des Spielers

<sup>3</sup> Paper Prototype -> Methode für Design Prozess: <https://www.uxpin.com/studio/blog/paper-prototyping-the-practical-beginners-guide/>

### 1.4.3 Warum mit VR?

1. Das Serious Game, wie in der Spielidee beschrieben, ist inspiriert von dem Indigel-Board, das im CTIT Informatikunterricht an der ZHAW von den Studenten in Praktika benutzt wurde. Um die Funktionsweise, wie zum Beispiel Gatter auf dem Board platzieren, naturgetreu abzubilden, bietet sich VR mit handgesteuerten Controllern an.
2. Durch die VR Plattform, können Dinge sichtbar gemacht werden, die man im echten Leben nicht sieht. In dem Serious Game sieht man den Stromfluss und erkennt anhand von Farben, welche Information über das Kabel geschickt wird.  
Im Spiel stehen dem Spieler "unendliche" Ressourcen zur Verfügung, es muss nicht aufgeräumt werden und es gibt keine defekten Komponenten. In den Praktika war es jeweils ein Kampf alle «intakten» Komponenten von den anderen Studenten zu erhalten.
3. In den Praktika musste man aufpassen, dass man das Board mit einem zyklischen Stromkreis nicht kurzschießt. Dies führte zu einem gewissen Druck. Im Serious Game hingegen kann der Spieler auf dem Board alle Konstellationen ausprobieren und so mehr Verständnis für die Logikgatter sowie die Stromkreise sammeln, ohne Angst zu haben, dass man teures Equipment zerstört.

## 1.5 Zielsetzung

Damit angehenden Ingenieuren die Konzepte der Schaltungslogik in einer sicheren und experimentierfreundlichen Umgebung vermittelt werden können, soll ein Szenario in VR entwickelt werden. Dem Spieler werden darin Theorie vermittelt sowie praktische Aufgaben gestellt. Die Aufgaben basieren auf real anwendbaren Schaltungen; wie dem Halbaddierer und dem Volladdierer. Zusätzlich werden auch Aufgaben anhand boolescher Matrizen mit Input und Output Werten abgebildet.

## 1.6 Oculus Quest 2

Das Serious Game wird mit dem VR-Headset Oculus Quest 2 (auf Abbildung 5 erkennbar) realisiert. Das Headset läuft auf einem Android basierten Betriebssystem. Die Brille ist zusätzlich mit zwei handgesteuerten Controllern ausgestattet.



Abbildung 5: Oculus Quest 2 Headset [13]

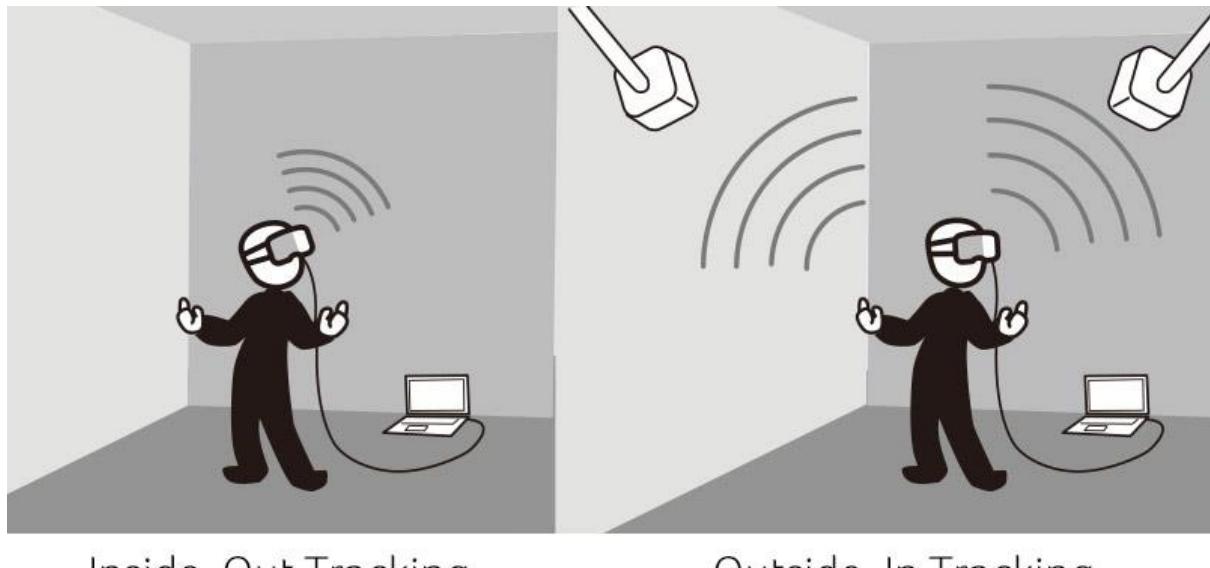
### 1.6.1 Spezifikationen der Oculus Quest 2

Das Oculus Quest 2 VR-Headset besitzt die folgenden Spezifikationen:

VR Category	All-in-one VR
Tracking	Oculus Insight inside-out tracking
Input	Two Oculus Touch controllers
Resolution	1832x1920 per eye
Display Panel	Fast-switch LCD
Audio	Integrated positional audio
Processor	Qualcomm® Snapdragon™ XR2 Platform
Battery Life	~2-3 hours
Storage	128GB, 256GB
Supported Usage Modes	Seated, standing, room scale
Glasses Friendly	Comfortable to wear with glasses or RX inserts

Tabelle 4: Oculus Quest 2 Spezifikationen [13]

Die Oculus Quest 2 verwendet Inside-Out Tracking. Das Inside-Out Tracking funktioniert über integrierte Kameras in der Brille. Es werden keine zusätzlichen Kameras gebraucht, um die Position des Benutzers im Raum festzustellen. Die Oculus Brille funktioniert kabellos. Es wird kein zusätzliches Gerät wie Computer oder Gamekonsole benötigt, um Medien auf der Brille wiederzugeben.



Inside-Out Tracking

Outside-In Tracking

Abbildung 6: Tracking Modi eines VR Headsets [10]

### 1.6.2 Controller Mapping der Oculus Quest 2 Controller

In Abbildung 7 erkennt man die verschiedenen Bezeichnungen für die Inputobjekte der Controller.

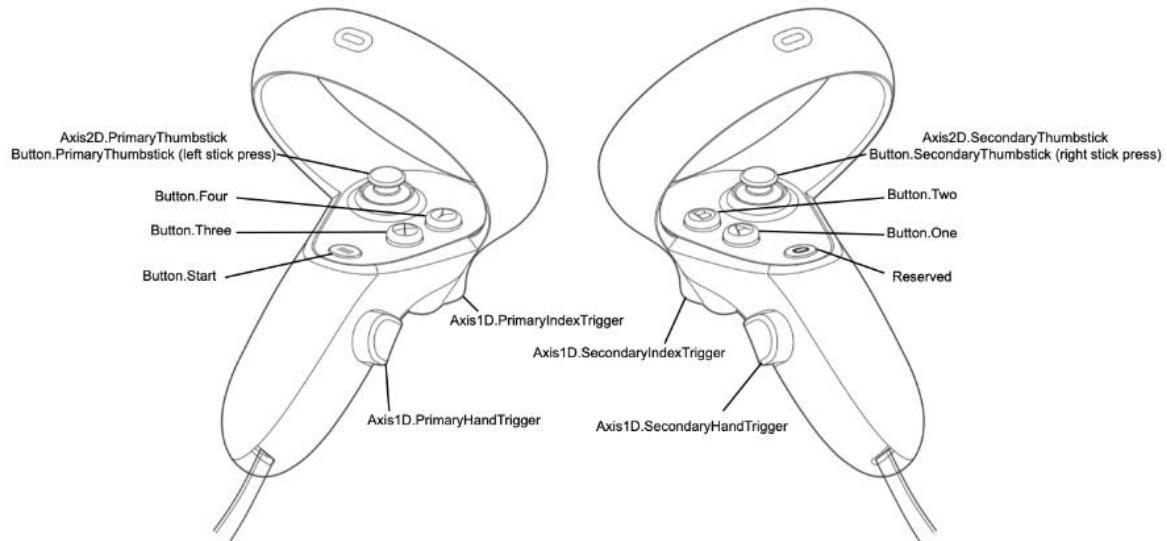


Abbildung 7: Controller Mapping [14]

## 2 Ausgangslage

Im folgenden Kapitel werden existierende Lösungen vorgestellt. Die folgenden Spiele wurden ausgewählt, da sie Ähnlichkeiten zum entwickelnden Serious Game aufweisen oder dieses inspiriert haben.

### 2.1 Existierende Lösungen

Virtual Reality hat sich in den letzten Jahren immer stärker in der Gaming-Industrie etabliert [15]. Durch Game-Engines wie Unity oder Unreal wird Entwicklern und etablierten Studios, ein leichter Einstieg in den VR Markt ermöglicht. Dies führt dazu, dass immer mehr neue oder adaptierte Spiele für Benutzer zur Verfügung stehen.

In diesem Kapitel werden Spiele beschrieben, in denen der Benutzer mittels Instruktionen eine gewünschte Aufgabe realisieren (oder programmieren) muss. Auf Grund der limitierten Menge an verfügbaren VR-Spielen zu diesem Thema, werden auch Spiele für andere Plattformen berücksichtigt (PC, Web, Konsole, Mobile, etc.).

#### 2.1.1 Please, Don't Touch Anything 3D

«Please, Don't Touch Anything 3D» ist ein Puzzle Spiel für VR, welches von dem Studio Four Quarters entwickelt und von Bulkypix und Plug-In-Digital veröffentlicht wurde. Der Spieler wird dabei im Sinne eines «Escape Rooms» in einem mysteriösen Raum eingeschlossen und aufgefordert, nichts zu berühren. Initial wird dem Spieler ein grosser roter Button auf einem Tisch angeboten. Entgegen dem Rat nichts zu berühren, werden dem Spieler durch Interaktionen mit Gegenständen im Raum immer mehr Rätsel präsentiert. Im Verlauf des Spiels werden immer mehr Werkzeuge und Schalter (und dadurch auch neue Interaktionsmöglichkeiten) freigeschaltet. [16]



Abbildung 8: Ausschnitt der Interaktionsmöglichkeiten [17]

## 2.1.2 Nandgame

«Nandgame» ist ein Web basiertes Puzzle Spiel, welches von Olav Junker Kjær entwickelt wurde. Der Spieler wird darin über einzelne Aufgaben simple Komponenten eines PCs bauen. Beispielweise soll im ersten Level mit zwei Relais ein NAND-Gatter gebaut werden. Im zweiten Level wird basierend auf diesem NAND-Gatter ein NOT-Gatter gebaut. Dies geht der Reihe nach weiter zum AND-, OR- und XOR-Gatter. In späteren Schritten werden auch komplexere Strukturen wie Halbaddierer und Volladdierer realisiert. [18]

Dem Spieler werden in diesem Spiel keine richtigen Hilfestellungen sowie theoretische Informationen angeboten. Das bedeutet, dass der Spieler entweder durch Probieren oder durch eine eigene Recherche auf die Lösung geraten muss.

Der logische Ablauf in unserem Serious Game folgt hingegen dem Ablauf des NAND-Games. In unserem Game beginnt man jedoch mit einem NAND-Gatter und arbeitet sich bis zum Volladdierer durch.

The screenshot shows the Nandgame interface. At the top, there are navigation links: 'Nandgame' (highlighted in dark grey), 'Solve Level', 'Levels' (selected), 'Donate', and 'About'. On the right, there is a language switch to 'English'. Below the navigation bar, there are buttons for 'Level Help', 'Check solution', 'Clear canvas', 'Clear all levels', and 'Skip level'.

**And**

An and gate's output is 1 when both inputs are 1:

Input	Output		
a   b	0   0	0	✓
0   1	0   1	0	✓
1   0	0   0	0	✓
1   1	1   1	1	✓

**Toolbox**

The toolbox contains two logic gates: a 'nand' gate (represented by a rectangle with two inputs 'a' and 'b' and one output) and an 'inv' gate (represented by a circle with one input and one output). There is also a 'Clear canvas' button.

**Logic Circuit Diagram**

The main area shows a logic circuit. The output of a 'nand' gate is connected to the input of an 'inv' gate. The 'inv' gate's output is then connected to one input of another 'nand' gate. The other input of this second 'nand' gate is connected to the 'a' input. The 'b' input is also connected to the second 'nand' gate. The final output of the second 'nand' gate is shown as '1'.

**Inputs**

Below the circuit, there are two input boxes labeled 'a' and 'b', each with a value of '0'.

Abbildung 9: Beschreibung und Aufgabe (Links) sowie das Platzieren und Verbinden der Gatter (Mitte). [18]

### 2.1.3 Logic World

«Logic World» ist ein 3-dimensionales Simulationsspiel für den PC. Der Spieler kann darin beliebige Schaltkreise aus der 1st oder 3rd Person bauen. Durch ein Baukasten-System kann der Spieler gewaltige Schaltkreise in kurzer Zeit aufbauen und simulieren. Im Early-Access Trailer wurden ein Maltool, ein «Schiffe versenken» und ein Computer zur Berechnung der Fibonacci-Sequenz gezeigt. Das Spiel kann als Single oder Multiplayer gespielt werden.

[19]

Das Spiel befindet sich aktuell noch im Early-Access Modus. [20]



Abbildung 10: Ausschnitt welcher das Baukastensystem sowie die Verbindungen aufzeigt. [20]

## 2.1.4 Minecraft

«Minecraft» ist ein Sandbox-Spiel welches auf dem Handy, PC, Konsole als auch in VR gespielt werden kann. Minecraft wurde im Jahr 2011 für PC veröffentlicht und später im Jahr 2014 durch Microsoft akquiriert. Bis heute werden regelmässig neue Funktionen und Interaktionen zum Spiel hinzugefügt. [21]

Eine spezielle Interaktion im Spiel ermöglicht es logische Schaltungen zu bauen. Dies kann mit dem im Spiel gefundenen Material «Red Stone» bewerkstelligt werden. Ambitionierte Spieler waren bereits in der Lage komplexe Architekturen wie einen funktionierenden Computer anzufertigen. [22]



Abbildung 11: Beispiel einer AND Schaltung mit Red-Stone. [23]

### 2.1.5 Human Resource Machine

«Human Resource Machine» ist ein Puzzle Spiel für PC welches von der «Tomorrow Corporation» entwickelt und veröffentlicht wurde. Der Spieler muss darin seine Mitarbeiter programmieren so dass diese eine bestimmte Aufgabe korrekt erledigen. Diese Aufgaben bestehen daraus das ein Band mit eingehenden Dokumenten korrekt auf das Band mit den ausgehenden Dokumenten gelegt werden soll [24]. Dem Spieler werden beim korrekten Lösen einer Aufgabe kontinuierlich neue, schwierigere Aufgaben gestellt.

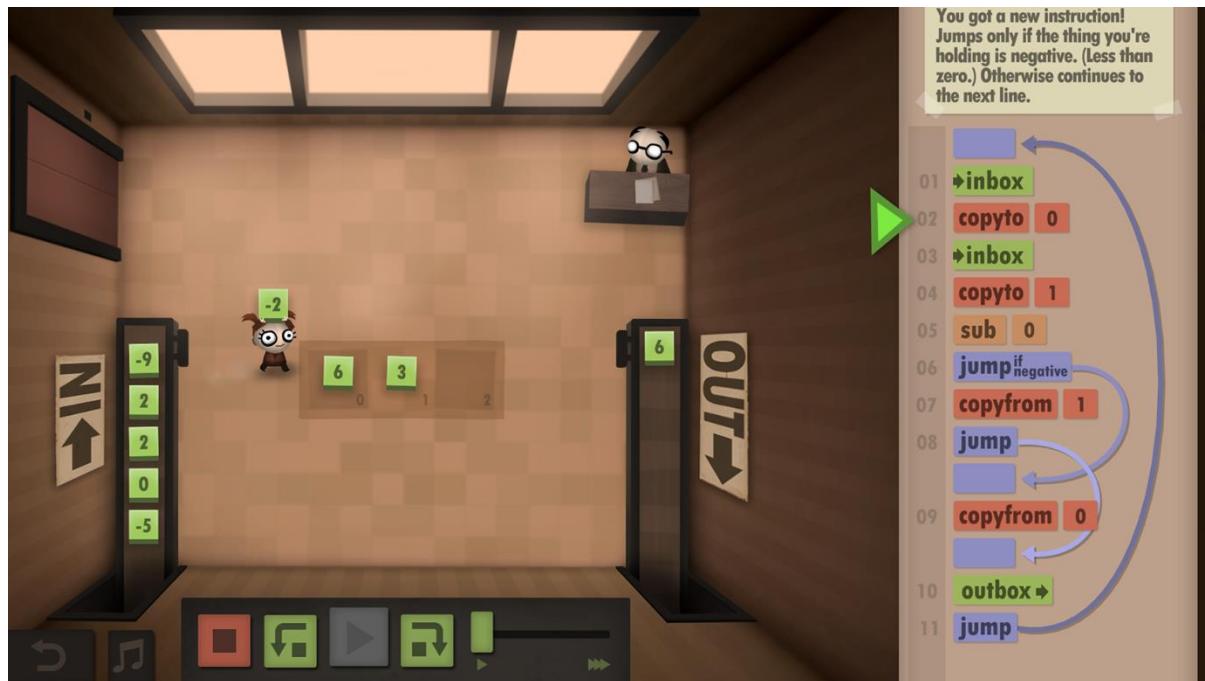


Abbildung 12: Perspektive des Spielers mit den Instruktionen an den Mitarbeiter (Rechts). [25]

### 3 Theoretische Grundlagen

In diesem Kapitel werden die theoretischen Grundlagen zu den logischen Verknüpfungen anhand abgebildeter Wahrheitstabellen und der Binären-Addition erklärt.

#### 3.1 Logische Verknüpfungen

Logische Verknüpfungen oder Junktoren sind Teil der logischen Operatoren und bilden den Kern der Aussagenlogik. Junktoren ermöglichen die Komposition von Aussagen, um komplexere Aussagen zu bilden. In diesem Unterkapitel werden alle elementaren Junktoren, welche im Serious Game erbaut werden, beschrieben.

##### 3.1.1 NAND

Die NAND-Verknüpfung verbindet zwei Aussagen so, dass die Verknüpfung genau dann wahr ist, wenn nicht beide Aussagen wahr sind. [26]

Input		Output
A	B	$A \cdot B$
0	0	1
0	1	1
1	0	1
1	1	0

Tabelle 5: Wahrheitstabelle des NAND

##### 3.1.2 NOT

Die Negation invertiert eine einzelne Aussage so, dass diese genau dann wahr ist, wenn die Aussage falsch ist.

Input	Output
A	$\neg A$
0	1
1	0

Tabelle 6: Wahrheitstabelle des NOT

### 3.1.3 AND

Die AND-Verknüpfung (alternativ auch als Konjunktion beschrieben) verbindet zwei Aussagen so dass die Verknüpfung genau dann wahr ist, wenn beide Aussagen wahr sind. [27]

Input		Output
A	B	$A \wedge B$
0	0	0
0	1	0
1	0	0
1	1	1

Tabelle 7: Wahrheitstabelle des AND

### 3.1.4 OR

Die OR-Verknüpfung (alternativ auch als Disjunktion beschrieben) verbindet zwei Aussagen so, dass die Verknüpfung genau dann wahr ist, wenn mindestens eine der beiden Aussagen wahr ist. [28]

Input		Output
A	B	$A \vee B$
0	0	0
0	1	1
1	0	1
1	1	1

Tabelle 8: Wahrheitstabelle des OR

### 3.1.5 XOR

Die XOR-Verknüpfung verbindet zwei Aussagen so, dass die Verknüpfung genau dann wahr ist, wenn genau eine Aussage wahr ist. [29]

Input		Output
A	B	$A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

Tabelle 9: Wahrheitstabelle des XOR

### 3.1.6 Binäre Addition

Bei der Binären Addition rechnet man nur mit zwei Ziffern „1“ und „0“. Das Carry (Übertrags-Bit) wird gesetzt, wenn die Summe  $\geq 2$  ( $1+1$ ) ist. [30]

#### Binary Addition of Two Bits

0	0	1	1
<u>+0</u>	<u>+1</u>	<u>+0</u>	<u>+1</u>
0	1	1 (carry)	1 ← 0

Abbildung 13: Binäre Addition [30]

#### 3.1.6.1 HALFADDER

Der HALFADDER (Halbaddierer) besteht aus einer AND und einer XOR Verknüpfung. In der untenstehenden Tabelle sieht man neu bei den Outputs einmal SUM und CARRY. Output 1 steht für die Summe und Output 2 für das Übertrags-Bit.

Input		Output1	Output2
A	B	CARRY	SUM
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Tabelle 10: Wahrheitstabelle zum Halbaddierer

### 3.1.6.2 FULLADDER

Der FULLADDED (Volladdierer) besteht aus zwei Halbaddierer und einer OR Verknüpfung. Analog zum Halbaddierer, besitzt der Volladdierer einen Output für die Summe sowie einen Output für das Übertrags-Bit (CARRY).

Input			Output1	Output2
A	B	C	CARRY	SUM
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Tabelle 11: Wahrheitstabelle zum Volladdierer

## 4 Didaktisches Konzept

Um einen klaren Lerninhalt für das Serious Game zu definieren, wurde ein didaktisches Konzept erstellt. Als Grundlage wurden Lernziele definiert, welche durch drei Schritte dem Benutzer vermittelt werden. Die drei Schritte, bestehend aus Einführung, Aufgaben und Prüfung werden in diesem Kapitel genauer beschrieben.

### 4.1 Lernziele

Die Lernziele beziehen sich ursprünglich auf den CTIT Unterricht der Informatikstudenten der ZHAW. Aufgrund des Lernplanwechsels des Studiengangs im Jahr 2019 wurde alternativ der CT Unterricht eingeführt. Der CT Unterricht vermittelt nicht mehr dieselben Lerninhalte wie der CTIT Unterricht, und hat das Indigel-Board aus den Praktika gestrichen und durch andere Lerninhalte ersetzt. Anhand dieses Wechsels in der Unterrichtsform, werden die Lernziele nun allgemein zum Thema Gatterlogik definiert.

- Der Spieler kann anhand einer booleschen Matrix eine gleichwertige Komposition von Logikgattern aufbauen.
- Der Spieler kann mittels der Komposition von Logikgattern einen HALFADDERR aufbauen.
- Der Spieler kann mittels der Komposition von Logikgattern einen FULLADDER aufbauen.

### 4.2 Vorwissen

Der Spieler sollte vorab wissen, wie man Wahrheitstabellen richtig liest, da diese als Grundlagen des Spiels dienen. Es handelt sich um einen linearen Spielablauf, jede Aufgabe baut auf der vorherigen Aufgabe auf, so dass kein anderes Vorwissen benötigt wird. Die Theoretischen Grundlagen können im *Kapitel 3 Theoretische Grundlagen* nachgelesen werden.

### 4.3 Einführung

Dem Spieler werden zu Beginn des Serious Games die grundlegenden Interaktionen, die Funktion des Boards, sowie der einzelnen Gatter beigebracht. Die Einführung ist interaktiv, das bedeutet, dass der Spieler aufgefordert wird, mit einzelnen Elementen zu interagieren und diese auszuprobieren.

### 4.4 Aufgabe

Nachdem der Spieler die Einführung abgeschlossen hat, werden der Reihe nach einzelne Aufgaben gestellt. Zu den komplexesten Aufgaben gehören das Aufbauen eines Halb- und eines Volladdierers.

### 4.5 Prüfung

Die Prüfung erfolgt automatisch beim Realisieren der Aufgaben. Sobald die Aufgabe korrekt gelöst wurde, kann der Spieler zur nächsten Aufgabe vorrücken. Richtig gelöst ist eine

Aufgabe, wenn die Wahrheitstabelle so auf dem Board mit den vorhandenen Gattern abgebildet wurde, dass die richtigen Output LED's analog zur Aufgabenstellung aufleuchten.

## 5 Anforderungsanalyse

Alle im unteren Diagramm aufgeführten Use-Cases für das Serious Game werden in diesem Kapitel in tabellarischer Form aufgezeigt.

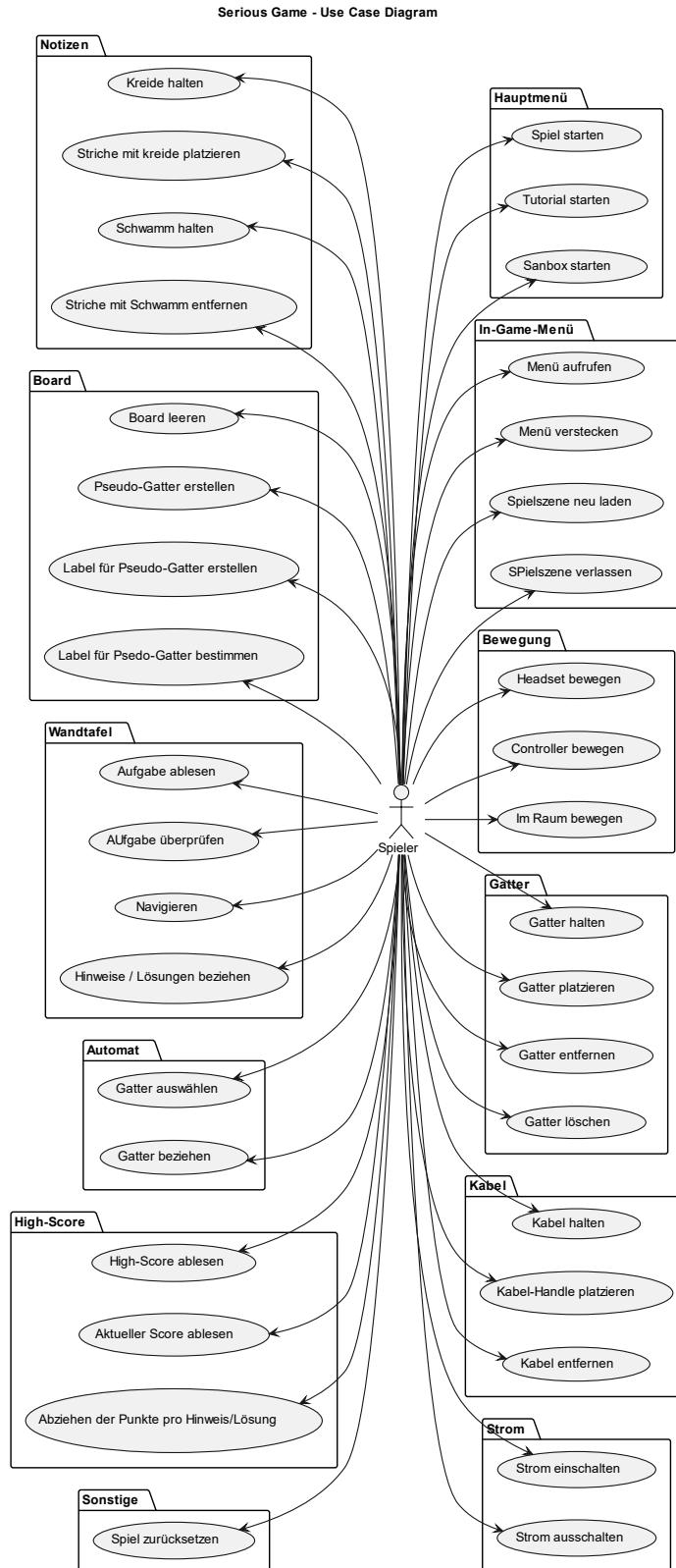


Abbildung 14: Use-Case Diagramm

## 5.1 Aufbau der Tabellen

Damit die Use-Cases, sowie deren funktionale Anforderungen, einfach und verständlich sind, wurden zwei tabellarische Formate definiert.

### 5.1.1 Aufbau Use-Case Tabelle

<b>Use Case ID</b>	<b>Eindeutige Identifikation der Form UC-##</b>
<b>Name des Use Cases</b>	Aussagekräftiger Name des Use-Cases
<b>Kurzbeschreibung</b>	Ein Satz, der den Use-Case beschreibt
<b>Priorität</b>	Priorität des Use-Cases: Hoch, Mittel, Niedrig
<b>Akteur</b>	Akteur, der den Use-Case ausführt
<b>Vorbedingungen</b>	Notwendige Bedingungen, die erfüllt sein müssen, um den Use-Case erfolgreich umzusetzen
<b>Resultat</b>	Beschreibung des Resultats, nach erfolgreicher Beendigung des Use-Cases
<b>Hauptszenario</b>	Ablauf des Use Cases
<b>Abweichungen</b>	Alternative Abläufe des Hauptszenarios

Tabelle 12: Vorlage eines Use Cases

### 5.1.2 Aufbau funktionale Anforderungstabelle

<b>Anforderungs ID</b>	<b>Eindeutige Identifikation der Form FA-##-##</b>
<b>Name der Anforderung</b>	Aussagekräftiger Name der Anforderung
<b>Kurzbeschreibung</b>	Ein Satz, der die Anforderung beschreibt
<b>Priorität</b>	Priorität der Anforderung: Hoch, Mittel, Niedrig
<b>Vorbedingungen</b>	Notwendige Bedingungen, die erfüllt sein müssen, um die Anforderung erfolgreich umzusetzen
<b>Akzeptanzkriterium</b>	Kriterium zur Überprüfung, ob die Anforderung erfüllt ist.
<b>Audiofeedback</b>	Ob beim Erfüllen der Anforderung ein Audiofeedback abgespielt wird, wenn ja welches.

Tabelle 13: Vorlage funktionale Anforderung

## 5.2 Grundlegende Vorbedingungen

Alle unten aufgeführten Use-Cases haben dieselben drei grundlegenden Anforderungen, die vorab erfüllt sein müssen, um die Use-Cases richtig umsetzen zu können:

- Die Applikation ist gestartet.
- Der Spieler hat das Headset aufgesetzt.
- Der Spieler hält die Controller.

## 5.3 Hauptmenü Use-Cases

In diesem Abschnitt werden die Use-Cases (sowie deren funktionale Anforderungen) zum Hauptmenü näher beschrieben.

### 5.3.1 Spiel Starten

<b>Use Case ID</b>	<b>UC-01</b>
<b>Name des Use Cases</b>	Spiel Starten
<b>Kurzbeschreibung</b>	Der Spieler kann das Spiel aus dem Hauptmenü aus starten.
<b>Priorität</b>	Hoch
<b>Akteur</b>	Spieler
<b>Vorbedingungen</b>	1. Das Hauptmenü ist aktiv.
<b>Resultat</b>	Der Spieler befindet sich im Spiel und kann spielen.
<b>Hauptszenario</b>	1. Spieler startet Applikation. 2. Spieler wählt Spiel im Startmenü aus. 3. Spiel wird gestartet.
<b>Abweichungen</b>	-

Tabelle 14: Use-Case - Spiel Starten

#### 5.3.1.1 Spiel Szene ist geladen

<b>Anforderungs ID</b>	<b>FA-01-01</b>
<b>Name der Anforderung</b>	Spieldaten sind geladen
<b>Kurzbeschreibung</b>	Die Spieldaten werden geladen. Der Spieler wird am Spielstartpunkt initialisiert.
<b>Priorität</b>	Hoch
<b>Vorbedingungen</b>	-
<b>Akzeptanzkriterium</b>	1. Die Spieldaten sind geladen. 2. Das Hauptmenü wird nicht mehr angezeigt. 3. Der Spieler befindet sich vor dem Labortisch.
<b>Audiofeedback</b>	-

Tabelle 15: Funktionale Anforderung - Spieldaten sind geladen

### 5.3.2 Tutorial Starten

<b>Use Case ID</b>	<b>UC-02</b>
<b>Name des Use Cases</b>	Tutorial Starten
<b>Kurzbeschreibung</b>	Der Spieler kann das Tutorial aus dem Hauptmenü aus starten.
<b>Priorität</b>	Hoch
<b>Akteur</b>	Spieler
<b>Vorbedingungen</b>	1. Das Hauptmenü ist aktiv.
<b>Resultat</b>	Der Spieler befindet sich im Tutorial und kann spielen.
<b>Hauptszenario</b>	1. Spieler startet Applikation. 2. Spieler wählt Spiel im Startmenü aus. 3. Spiel wird gestartet.
<b>Abweichungen</b>	-

Tabelle 16: Use-Case - Tutorial Starten

#### 5.3.2.1 Tutorial Szene ist geladen

<b>Anforderungs ID</b>	<b>FA-02-01</b>
<b>Name der Anforderung</b>	Tutorial Szene ist geladen
<b>Kurzbeschreibung</b>	Die Tutorialszene wird geladen. Der Spieler wird am Spielstartpunkt initialisiert.
<b>Priorität</b>	Hoch
<b>Vorbedingungen</b>	-
<b>Akzeptanzkriterium</b>	1. Die Tutorialszene ist geladen. 2. Das Hauptmenü wird nicht mehr angezeigt. 3. Der Spieler befindet sich vor dem Labortisch.
<b>Audiofeedback</b>	-

Tabelle 17: Funktionale Anforderung - Tutorial Szene ist geladen

### 5.3.3 Sandbox Starten

<b>Use Case ID</b>	<b>UC-03</b>
<b>Name des Use Cases</b>	Sandbox Starten
<b>Kurzbeschreibung</b>	Der Spieler kann die Sandbox aus dem Hauptmenü aus starten.
<b>Priorität</b>	Hoch
<b>Akteur</b>	Spieler
<b>Vorbedingungen</b>	1. Das Hauptmenü ist aktiv.
<b>Resultat</b>	Der Spieler befindet sich in der Sandbox und kann spielen.
<b>Hauptszenario</b>	1. Spieler startet Applikation. 2. Spieler wählt Spiel im Startmenü aus. 3. Spiel wird gestartet.
<b>Abweichungen</b>	-

Tabelle 18: Use-Case - Sandbox starten

### 5.3.3.1 Sandbox Szene ist geladen

<b>Anforderungs ID</b>	<b>FA-03-01</b>
<b>Name der Anforderung</b>	Sandboxszene ist geladen
<b>Kurzbeschreibung</b>	Die Sandboxeszene wird geladen. Der Spieler wird am Spielstartpunkt initialisiert.
<b>Priorität</b>	Hoch
<b>Vorbedingungen</b>	-
<b>Akzeptanzkriterium</b>	1. Die Sandboxeszene ist geladen. 2. Das Hauptmenü wird nicht mehr angezeigt. 3. Der Spieler befindet sich vor dem Labortisch.
<b>Audiofeedback</b>	-

Tabelle 19: Funktionale Anforderung - Sandboxeszene ist geladen

## 5.4 In-Game Menü

In diesem Abschnitt werden die Use-Cases (sowie deren funktionale Anforderungen) zum In-Game Menü näher beschrieben.

### 5.4.1 Menü aufrufen

<b>Use Case ID</b>	<b>UC-04</b>
<b>Name des Use Cases</b>	Menü aufrufen
<b>Kurzbeschreibung</b>	Der Spieler kann während dem Spiel jederzeit das In-Game Menü mit dem Button.Start / Button.One Knopf aufrufen.
<b>Priorität</b>	Hoch
<b>Akteur</b>	Spieler
<b>Vorbedingungen</b>	1. Das In-Game Menü ist geschlossen.
<b>Resultat</b>	Das In-Game Menü wird angezeigt.
<b>Hauptszenario</b>	1. Der Spieler öffnet In-Game Menü mit dem Button.Start Knopf.
<b>Abweichungen</b>	1. Der Spieler öffnet In-Game Menü mit dem Button.One Knopf.

Tabelle 20: Use-Case - Menü aufrufen

#### 5.4.1.1 In-Game Menü wird angezeigt

<b>Anforderungs ID</b>	<b>FA-04-01</b>
<b>Name der Anforderung</b>	In-Game Menü wird angezeigt
<b>Kurzbeschreibung</b>	Das In-Game Menü ist geladen und wird im Sichtfeld des Spielers angezeigt.
<b>Priorität</b>	Hoch
<b>Vorbedingungen</b>	-
<b>Akzeptanzkriterium</b>	1. Das In-Game Menü wird angezeigt. 2. Das In-Game Menü bewegt sich relativ zum Sichtfeld des Spielers.
<b>Audiofeedback</b>	-

Tabelle 21: Funktionale Anforderung - In-Game Menü wird angezeigt

#### 5.4.1.2 In-Game Menü kann bedient werden

<b>Anforderungs ID</b>	<b>FA-04-02</b>
<b>Name der Anforderung</b>	In-Game Menü kann bedient werden
<b>Kurzbeschreibung</b>	Das In-Game Menü ist geladen und kann mit Ray-Tracing bedient werden.
<b>Priorität</b>	Hoch
<b>Vorbedingungen</b>	1. Das In-Game Menü ist geladen
<b>Akzeptanzkriterium</b>	1. Das In-Game Menü kann mit Raytracing bedient werden.
<b>Audiofeedback</b>	-

Tabelle 22: Funktionale Anforderung - In-Game Menü kann bedient werden

#### 5.4.2 Menü verstecken

<b>Use Case ID</b>	<b>UC-05</b>
<b>Name des Use Cases</b>	Menü verstecken
<b>Kurzbeschreibung</b>	Der Spieler kann das In-Game Menü über Knopfdruck (Button.Start / Button.One) oder über Menü-Knopf schliessen.
<b>Priorität</b>	Hoch
<b>Akteur</b>	Spieler
<b>Vorbedingungen</b>	1. Das In-Game Menü ist geöffnet und wird angezeigt.
<b>Resultat</b>	Das In-Game Menü ist geschlossen und wird nicht mehr angezeigt.
<b>Hauptszenario</b>	1. Spieler schliesst das In-Game Menü mit dem Menü-Knopf „Continue“.
<b>Abweichungen</b>	1. Spieler schliesst das In-Game Menü mit dem Button.Start Knopf. 2. Spieler schliesst das In-Game Menü mit dem Button.One Knopf.

Tabelle 23: Use-Case - Menü verstecken

#### 5.4.2.1 Menü wird nicht mehr angezeigt

<b>Anforderungs ID</b>	<b>FA-05-01</b>
<b>Name der Anforderung</b>	In-Game Menü wird nicht mehr angezeigt
<b>Kurzbeschreibung</b>	Das In-Game Menü ist weg und wird nicht mehr im Sichtfeld des Spielers angezeigt.
<b>Priorität</b>	Hoch
<b>Vorbedingungen</b>	1. Das In-Game Menü wird angezeigt.
<b>Akzeptanzkriterium</b>	1. Das In-Game Menü wird nicht mehr angezeigt.
<b>Audiofeedback</b>	-

Tabelle 24: Funktionale Anforderung - In-Game Menü wird nicht mehr angezeigt

### 5.4.3 Spielszene neu laden

<b>Use Case ID</b>	<b>UC-06</b>
<b>Name des Use Cases</b>	Spielszene neu laden
<b>Kurzbeschreibung</b>	Der Spieler kann über den In-Game Menü Knopf „Reload“ die Szene neu laden.
<b>Priorität</b>	Hoch
<b>Akteur</b>	Spieler
<b>Vorbedingungen</b>	1. Das In-Game Menü ist geöffnet.
<b>Resultat</b>	Die Spiel Szene ist neu geladen.
<b>Hauptszenario</b>	1. Der Spieler öffnet In-Game Menü. 2. Der Spieler wählt „Reload“ aus.
<b>Abweichungen</b>	-

Tabelle 25: Use-Case - Spielszene neu laden

#### 5.4.3.1 Spiel Szene wird neu geladen

<b>Anforderungs ID</b>	<b>FA-06-01</b>
<b>Name der Anforderung</b>	Spielszene wird neu geladen
<b>Kurzbeschreibung</b>	Die Spielszene wird geladen. Der Spieler wird am Spielstartpunkt initialisiert.
<b>Priorität</b>	Hoch
<b>Vorbedingungen</b>	-
<b>Akzeptanzkriterium</b>	1. Die Spielszene ist neu geladen. 2. Das Spiel beginnt von vorne. 3. Das Hauptmenü wird nicht mehr angezeigt. 4. Der Spieler befindet sich vor dem Labortisch.
<b>Audiofeedback</b>	-

Tabelle 26: Funktionale Anforderung - Spielszene wird neu geladen

### 5.4.4 Spielszene verlassen

<b>Use Case ID</b>	<b>UC-07</b>
<b>Name des Use Cases</b>	Spielszene verlassen
<b>Kurzbeschreibung</b>	Der Spieler kann über den In-Game Menü Knopf „Exit“ das Spiel verlassen.
<b>Priorität</b>	Hoch
<b>Akteur</b>	Spieler
<b>Vorbedingungen</b>	1. Das In-Game Menü ist geöffnet.
<b>Resultat</b>	Der Spieler befindet sich in der Hauptmenü Szene.
<b>Hauptszenario</b>	1. Der Spieler öffnet In-Game Menü. 2. Der Spieler wählt „Exit“ aus.
<b>Abweichungen</b>	-

Tabelle 27: Use-Case - Spielszene verlassen

#### *5.4.4.1 Spiel Szene ist verlassen*

<b>Anforderungs ID</b>	<b>FA-07-01</b>
<b>Name der Anforderung</b>	Spielszene wird verlassen
<b>Kurzbeschreibung</b>	Die Spielszene wird verlassen. Der Spieler befindet sich wieder im Hauptmenü.
<b>Priorität</b>	Hoch
<b>Vorbedingungen</b>	-
<b>Akzeptanzkriterium</b>	1. Der Spieler befindet sich im Hauptmenü.
<b>Audiofeedback</b>	-

*Tabelle 28: Funktionale Anforderung - Spielszene wird verlassen*

## 5.5 Headset- und Controllerbedienung

In diesem Abschnitt werden die Use-Cases (sowie deren funktionale Anforderungen) zur Headset- und Controllerbedienung näher beschrieben.

### 5.5.1 Headset bewegen

<b>Use Case ID</b>	<b>UC-08</b>
<b>Name des Use Cases</b>	Headset bewegen
<b>Kurzbeschreibung</b>	Der Spieler kann das Headset bewegen und die Perspektive wird relativ dazu ausgerichtet.
<b>Priorität</b>	Hoch
<b>Akteur</b>	Spieler
<b>Vorbedingungen</b>	-
<b>Resultat</b>	Der Spieler bewegt das Headset.
<b>Hauptszenario</b>	1. Der Spieler bewegt sich im Raum und das Headset bewegt sich mit.
<b>Abweichungen</b>	-

Tabelle 29: Use-Case - Headset bewegen

#### 5.5.1.1 Perspektive wird ausgerichtet

<b>Anforderungs ID</b>	<b>FA-08-01</b>
<b>Name der Anforderung</b>	Perspektive wird ausgerichtet.
<b>Kurzbeschreibung</b>	Die Perspektive wird relativ um Headset ausgerichtet.
<b>Priorität</b>	Hoch
<b>Vorbedingungen</b>	-
<b>Akzeptanzkriterium</b>	1. Die Perspektive wird relativ um Headset ausgerichtet.
<b>Audiofeedback</b>	-

Tabelle 30: Funktionale Anforderung - Perspektive wird ausgerichtet

## 5.5.2 Controller bewegen

<b>Use Case ID</b>	<b>UC-09</b>
<b>Name des Use Cases</b>	Controller bewegen
<b>Kurzbeschreibung</b>	Der Spieler kann die Controller bewegen und die virtuellen Hände werden relativ dazu ausgerichtet.
<b>Priorität</b>	Hoch
<b>Akteur</b>	Spieler
<b>Vorbedingungen</b>	-
<b>Resultat</b>	<ol style="list-style-type: none"> <li>1. Der Spieler bewegt die virtuellen Hände.</li> <li>2. Der Spieler schliesst die virtuelle Hand.</li> <li>3. Der Spieler öffnet die virtuelle Hand.</li> </ol>
<b>Hauptszenario</b>	Der Spieler benutzt die Controller zur Bewegung der Hände.
<b>Abweichungen</b>	-

Tabelle 31: Use-Case - Controller bewegen

### 5.5.2.1 Virtuelle Hände werden relativ zum Controller bewegt

<b>Anforderungs ID</b>	<b>FA-09-01</b>
<b>Name der Anforderung</b>	Virtuelle Hände werden relativ zum Controller bewegt.
<b>Kurzbeschreibung</b>	Die virtuellen Hände werden durch den Controller gesteuert.
<b>Priorität</b>	Hoch
<b>Vorbedingungen</b>	-
<b>Akzeptanzkriterium</b>	1. Die virtuellen Hände bewegen sich relativ zu den echten Händen im Raum.
<b>Audiofeedback</b>	-

Tabelle 32: Funktionale Anforderung - Virtuelle Hände werden relativ zum Controller bewegt

### 5.5.2.2 Virtuelle Hände werden durch Controller geschlossen

<b>Anforderungs ID</b>	<b>FA-09-02</b>
<b>Name der Anforderung</b>	Virtuelle Hände werden durch die Controller geschlossen.
<b>Kurzbeschreibung</b>	Die virtuellen Hände werden durch Drücken des Axis1D.PrimaryHandTrigger / Axis1DSecondaryHandTrigger Knopfes auf dem Controller geschlossen.
<b>Priorität</b>	Hoch
<b>Vorbedingungen</b>	1. Die Hände sind geöffnet.
<b>Akzeptanzkriterium</b>	1. Die Hände werden nach Drücken des Axis1D.PrimaryHandTrigger / Axis1DSecondaryHandTrigger Knopfes geschlossen.
<b>Audiofeedback</b>	-

Tabelle 33: Funktionale Anforderung - Virtuelle Hände werden durch Controller geschlossen

### 5.5.2.3 Hände werden durch Controller geöffnet

<b>Anforderungs ID</b>	<b>FA-09-03</b>
<b>Name der Anforderung</b>	Virtuelle Hände werden durch die Controller geöffnet.
<b>Kurzbeschreibung</b>	Die virtuellen Hände werden durch Loslassen des Axis1D.PrimaryHandTrigger / Axis1DSecondaryHandTrigger Knopfes auf dem Controller geöffnet.
<b>Priorität</b>	Hoch
<b>Vorbedingungen</b>	1. Die Hände sind geschlossen.
<b>Akzeptanzkriterium</b>	1. Die Hände werden nach Loslassen des Axis1D.PrimaryHandTrigger / Axis1DSecondaryHandTrigger Knopfes geöffnet.
<b>Audiofeedback</b>	-

Tabelle 34: Funktionale Anforderung - Hände werden durch Controller geöffnet

### 5.5.3 Im Raum bewegen

<b>Use Case ID</b>	<b>UC-10</b>
<b>Name des Use Cases</b>	Im Raum bewegen
<b>Kurzbeschreibung</b>	Der Spieler kann sich frei im Raum durch Teleportieren bewegen.
<b>Priorität</b>	Hoch
<b>Akteur</b>	Spieler
<b>Vorbedingungen</b>	1. Der Zielort ist nicht durch ein Objekt blockiert.
<b>Resultat</b>	Der Spieler befindet sich nach der Teleportation am gewünschten Zielort.
<b>Hauptszenario</b>	<ol style="list-style-type: none"> <li>1. Spieler möchte zu einem bestimmten Zielort teleportieren.</li> <li>2. Spieler zeigt mit der Hand an den gewünschten Zielort.</li> <li>3. Spieler drückt den Index-Trigger am Controller.</li> <li>4. Spieler teleportiert.</li> <li>5. Spieler befindet sich am gewünschten Zielort.</li> </ol>
<b>Abweichungen</b>	-

Tabelle 35: Use-Case - Im Raum bewegen

### 5.5.3.1 Spieler an beliebigen freien Ort im Raum teleportieren

<b>Anforderungs ID</b>	<b>FA-10-01</b>
<b>Name der Anforderung</b>	Spieler an beliebigen freien Ort im Raum teleportieren
<b>Kurzbeschreibung</b>	Der Spieler wird durch Drücken des PrimaryIndexTrigger Knopfes an die gewünschte Stelle teleportiert.
<b>Priorität</b>	Hoch
<b>Vorbedingungen</b>	1. Der gewünschte Ort ist frei.
<b>Akzeptanzkriterium</b>	1. Der Spieler befindet sich am gewünschten Ort nach dem Teleportieren.
<b>Audiofeedback</b>	-

Tabelle 36: Funktionale Anforderung - Spieler an beliebigen freien Ort im Raum teleportieren

## 5.6 Gatter

In diesem Abschnitt werden die Use-Cases (sowie deren funktionale Anforderungen) zum Gatter näher beschrieben.

### 5.6.1 Gatter halten

<b>Use Case ID</b>	<b>UC-11</b>
<b>Name des Use Cases</b>	Gatter halten
<b>Kurzbeschreibung</b>	Das Gatter wird vom Spieler in einer seiner Hände gehalten.
<b>Priorität</b>	Hoch
<b>Akteur</b>	Spieler
<b>Vorbedingungen</b>	1. Die Hand zum Halten ist frei, d.h. hält nichts.
<b>Resultat</b>	Das Gatter wird in der Hand gehalten.
<b>Hauptszenario</b>	1. Der Spieler bewegt seine Hand zum Gatter, welches er halten möchte. 2. Der Spieler drückt Knopf am Controller. 3. Der Spieler hält Gatter in der Hand.
<b>Abweichungen</b>	-

Tabelle 37: Use-Case - Gatter halten

#### 5.6.1.1 Gatter wird gehalten

<b>Anforderungs ID</b>	<b>FA-11-01</b>
<b>Name der Anforderung</b>	Gatter wird gehalten
<b>Kurzbeschreibung</b>	Das Gatter wird durch das Drücken und gedrückt halten des Axis1D.PrimaryHandTrigger / Axis1DSecondaryHandTrigger in der Hand gehalten.
<b>Priorität</b>	Hoch
<b>Vorbedingungen</b>	1. Die Hand ist frei.
<b>Akzeptanzkriterium</b>	1. Das Gatter wird in der gewünschten Hand gehalten.
<b>Audiofeedback</b>	-

Tabelle 38: Funktionale Anforderung - Gatter wird gehalten

#### 5.6.1.2 Gatter wird losgelassen

<b>Anforderungs ID</b>	<b>FA-11-02</b>
<b>Name der Anforderung</b>	Gatter wird losgelassen
<b>Kurzbeschreibung</b>	Das Gatter wird durch das Loslassen des Axis1D.PrimaryHandTrigger / Axis1DSecondaryHandTrigger losgelassen.
<b>Priorität</b>	Hoch
<b>Vorbedingungen</b>	1. Eine Hand hält das Gatter.
<b>Akzeptanzkriterium</b>	1. Das Gatter ist nicht mehr in der Hand.
<b>Audiofeedback</b>	-

Tabelle 39: Funktionale Anforderung - Gatter wird losgelassen

## 5.6.2 Gatter platzieren

<b>Use Case ID</b>	<b>UC-12</b>
<b>Name des Use Cases</b>	Gatter platzieren
<b>Kurzbeschreibung</b>	Der Spieler kann ein Gatter oberhalb eines freien Sockets loslassen. Das Gatter wird auf dem Socket platziert.
<b>Priorität</b>	Hoch
<b>Akteur</b>	Spieler
<b>Vorbedingungen</b>	<ol style="list-style-type: none"> <li>1. Das Gatter ist in der Hand.</li> <li>2. Der Platz ist frei auf dem Board.</li> </ol>
<b>Resultat</b>	Das Gatter befindet sich auf dem Board.
<b>Hauptszenario</b>	<ol style="list-style-type: none"> <li>1. Der Spieler hält Gatter in der Hand.</li> <li>2. Der Spieler lässt das Gatter oberhalb eines freien Socket.</li> <li>3. Die Vorschau erscheint.</li> <li>4. Der Spieler lässt Gatter los.</li> <li>5. Das Gatter snappt auf das Board.</li> <li>6. Die Vorschau verschwindet.</li> </ol>
<b>Abweichungen</b>	-

Tabelle 40: Use-Case - Gatter platzieren

### 5.6.2.1 Gatter wird auf freiem Socket platziert

<b>Anforderungs ID</b>	<b>FA-12-01</b>
<b>Name der Anforderung</b>	Gatter wird auf freiem Socket platziert
<b>Kurzbeschreibung</b>	Das Gatter wird auf dem freien Socket platziert und die Vorschau verschwindet.
<b>Priorität</b>	Hoch
<b>Vorbedingungen</b>	<ol style="list-style-type: none"> <li>1. Gatter ist in der Hand.</li> <li>2. Socket auf dem Board ist frei.</li> <li>3. Gatter wird losgelassen.</li> </ol>
<b>Akzeptanzkriterium</b>	<ol style="list-style-type: none"> <li>1. Das Gatter ist nicht mehr in der Hand.</li> <li>2. Das Gatter befindet sich auf dem Socket.</li> <li>3. Die Vorschau ist verschwunden.</li> </ol>
<b>Audiofeedback</b>	Snap-Geräusch

Tabelle 41: Funktionale Anforderung - Gatter wird auf freiem Socket platziert

### 5.6.3 Gatter entfernen

<b>Use Case ID</b>	<b>UC-13</b>
<b>Name des Use Cases</b>	Gatter entfernen
<b>Kurzbeschreibung</b>	Gatter wird vom Board mit einer Ziehbewegung entfernt. Alle Kabel werden unsnappt.
<b>Priorität</b>	Hoch
<b>Akteur</b>	Spieler
<b>Vorbedingungen</b>	1. Das Gatter befindet sich auf dem Board.
<b>Resultat</b>	Das Gatter befindet sich nicht mehr auf dem Board.
<b>Hauptszenario</b>	1. Der Spieler entfernt das Gatter, indem er daran zieht.
<b>Abweichungen</b>	-

Tabelle 42: Use-Case - Gatter entfernen

#### 5.6.3.1 Gatter wird vom Socket entfernt

<b>Anforderungs ID</b>	<b>FA-13-01</b>
<b>Name der Anforderung</b>	Gatter wird vom Socket entfernt
<b>Kurzbeschreibung</b>	Das Gatter wird vom Socket entfernt.
<b>Priorität</b>	Hoch
<b>Vorbedingungen</b>	1. Gatter befindet sich auf dem Board.
<b>Akzeptanzkriterium</b>	1. Gatter befindet sich nicht mehr auf dem Board. 2. Alle Kabel sind unsnappt. 3. Gatter befindet sich in der Hand des Spielers.
<b>Audiofeedback</b>	Snap-Geräusch

Tabelle 43: Funktionale Anforderung - Gatter wird vom Socket entfernt

### 5.6.4 Gatter löschen

<b>Use Case ID</b>	<b>UC-14</b>
<b>Name des Use Cases</b>	Gatter löschen
<b>Kurzbeschreibung</b>	Gatter wird vom Spieler auf den Boden geworfen, um gelöscht zu werden.
<b>Priorität</b>	Tief
<b>Akteur</b>	Spieler
<b>Vorbedingungen</b>	1. Der Spieler hält Gatter in der Hand.
<b>Resultat</b>	Das Gatter ist gelöscht.
<b>Hauptszenario</b>	1. Der Spieler lässt Gatter über dem Boden los. 2. Das Gatter fällt auf den Boden. 3. Das Gatter wird gelöscht.
<b>Abweichungen</b>	1. Das Gatter fällt nicht auf den Boden, sondern auf ein Objekt. 2. Das Gatter wird nicht gelöscht.

Tabelle 44: Use-Case - Gatter löschen

#### *5.6.4.1 Gatter wird gelöscht*

<b>Anforderungs ID</b>	<b>FA-14-01</b>
<b>Name der Anforderung</b>	Gatter wird gelöscht
<b>Kurzbeschreibung</b>	Das Gatter wird beim Berühren des Bodens aus der Spielszene entfernt.
<b>Priorität</b>	Tief
<b>Vorbedingungen</b>	1. Das Gatter lebt in der Szene.
<b>Akzeptanzkriterium</b>	1. Das Gatter ist aus der Spielszene entfernt.
<b>Audiofeedback</b>	Lösch-Geräusch

*Tabelle 45: Funktionale Anforderung - Gatter wird gelöscht*

## 5.7 Kabel

In diesem Abschnitt werden die Use-Cases (sowie deren funktionale Anforderungen) zum Kabel näher beschrieben.

### 5.7.1 Kabel halten

<b>Use Case ID</b>	<b>UC-15</b>
<b>Name des Use Cases</b>	Kabel halten
<b>Kurzbeschreibung</b>	Das Kabel wird vom Spieler in einer seiner Hände gehalten. Kabel bewegt sich mit der Hand mit und wird länger oder kürzer.
<b>Priorität</b>	Hoch
<b>Akteur</b>	Spieler
<b>Vorbedingungen</b>	1. Die Hand zum Halten ist frei, d.h. hält nichts.
<b>Resultat</b>	Das Kabel wird am Kabel-Handle in der Hand gehalten.
<b>Hauptszenario</b>	<ol style="list-style-type: none"> <li>Der Spieler bewegt seine Hand zum Kabel, das er halten möchte.</li> <li>Der Spieler drückt Knopf am Controller.</li> <li>Der Spieler hält Kabel in der Hand.</li> <li>Der Spieler kann am Kabel ziehen.</li> </ol>
<b>Abweichungen</b>	-

Tabelle 46: Use-Case - Kabel halten

#### 5.7.1.1 Kabel wird am Kabel-Handle gehalten

<b>Anforderungs ID</b>	<b>FA-15-01</b>
<b>Name der Anforderung</b>	Kabel wird am Kabel-Handle gehalten
<b>Kurzbeschreibung</b>	Das Kabel wird durch das Drücken und gedrückt halten des Axis1D.PrimaryHandTrigger / Axis1DSecondaryHandTrigger am Kabel-Handle in der Hand gehalten.
<b>Priorität</b>	Hoch
<b>Vorbedingungen</b>	1. Die Hand ist frei.
<b>Akzeptanzkriterium</b>	1. Das Kabel wird am Kabel-Handle wird in der gewünschten Hand gehalten.
<b>Audiofeedback</b>	-

Tabelle 47: Funktionale Anforderung - Kabel wird am Kabel-Handle gehalten

### 5.7.1.2 Kabel wird in der Luft losgelassen

<b>Anforderungs ID</b>	<b>FA-15-02</b>
<b>Name der Anforderung</b>	Kabel wird in der Luft losgelassen
<b>Kurzbeschreibung</b>	Das Kabel wird durch das Loslassen des Axis1D.PrimaryHandTrigger / Axis1DSecondaryHandTrigger losgelassen.
<b>Priorität</b>	Hoch
<b>Vorbedingungen</b>	<ol style="list-style-type: none"> <li>1. Eine Hand hält das Kabel.</li> <li>2. Das Kabel befindet sich nicht in der Nähe eines Kabel-Eingangs.</li> </ol>
<b>Akzeptanzkriterium</b>	<ol style="list-style-type: none"> <li>1. Das Kabel ist nicht mehr in der Hand.</li> <li>2. Das Kabel verkürzt sich zurück zur ursprünglichen Position.</li> </ol>
<b>Audiofeedback</b>	-

Tabelle 48: Funktionale Anforderung - Kabel loslassen

### 5.7.2 Kabel verbinden

<b>Use Case ID</b>	<b>UC-16</b>
<b>Name des Use Cases</b>	Kabel verbinden
<b>Kurzbeschreibung</b>	Der Spieler kann ein Kabel oberhalb eines freien Kabel-Eingangs loslassen und dadurch damit verbinden. Das Kabel wird nun zwischen dem Kabel-Ausgang und dem Kabel-Eingang entlang gezogen.
<b>Priorität</b>	Hoch
<b>Akteur</b>	Spieler
<b>Vorbedingungen</b>	<ol style="list-style-type: none"> <li>1. Das Kabel ist in der Hand.</li> <li>2. Ein Eingang eines Gatters ist frei.</li> </ol>
<b>Resultat</b>	Das Kabel ist verbunden mit dem Gatter.
<b>Hauptszenario</b>	<ol style="list-style-type: none"> <li>1. Der Spieler zieht das Kabel zu einem freien Kabel-Eingang.</li> <li>2. Die Vorschau erscheint.</li> <li>3. Der Spieler lässt Kabel los.</li> <li>4. Das Kabel verbindet sich mit dem Kabel-Eingang.</li> <li>5. Die Vorschau verschwindet.</li> </ol>
<b>Abweichungen</b>	-

Tabelle 49: Use-Case - Kabel verbinden

### 5.7.2.1 Kabel wird auf freiem Kabel-Eingang platziert

<b>Anforderungs ID</b>	<b>FA-16-01</b>
<b>Name der Anforderung</b>	Kabel wird am Kabel-handle auf freiem Gatter-Socket platziert
<b>Kurzbeschreibung</b>	Das Gatter wird auf dem freien Socket platziert und die Vorschau verschwindet.
<b>Priorität</b>	Hoch
<b>Vorbedingungen</b>	<ol style="list-style-type: none"> <li>1. Das Kabel ist in der Hand.</li> <li>2. Der Socket für das Kabel auf dem Board ist frei.</li> <li>3. Der Kabel-Handle wird losgelassen.</li> </ol>
<b>Akzeptanzkriterium</b>	<ol style="list-style-type: none"> <li>1. Das Kabel ist nicht mehr in der Hand.</li> <li>2. Das Kabel befindet sich auf dem Gatter-Socket.</li> <li>3. Die Vorschau ist verschwunden.</li> </ol>
<b>Audiofeedback</b>	Snap-Geräusch

Tabelle 50: Funktionale Anforderung - Kabel wird auf freiem Kabel-Eingang platziert

### 5.7.3 Kabel entfernen

<b>Use Case ID</b>	<b>UC-17</b>
<b>Name des Use Cases</b>	Kabel entfernen
<b>Kurzbeschreibung</b>	Der Spieler kann ein platziertes Kabel wegziehen und dadurch entfernen.
<b>Priorität</b>	Hoch
<b>Akteur</b>	Spieler
<b>Vorbedingungen</b>	<ol style="list-style-type: none"> <li>1. Das Kabel ist verbunden.</li> </ol>
<b>Resultat</b>	Das Kabel ist entfernt.
<b>Hauptszenario</b>	<ol style="list-style-type: none"> <li>1. Der Spieler zieht am verbundenen Kabel, um es zu entfernen.</li> </ol>
<b>Abweichungen</b>	<ol style="list-style-type: none"> <li>1. Der Spieler zieht am Gatter</li> </ol>

Tabelle 51: Use-Case - Kabel entfernen

### 5.7.3.1 Verbundenes Kabel wird entfernt

<b>Anforderungs ID</b>	<b>FA-17-01</b>
<b>Name der Anforderung</b>	Verbundenes Kabel wird entfernt
<b>Kurzbeschreibung</b>	Das Kabel wird vom Gatter-Socket entfernt, indem es in der Mitte am Handle weggezogen wird.
<b>Priorität</b>	Hoch
<b>Vorbedingungen</b>	<ol style="list-style-type: none"> <li>1. Ein Kabel ist platziert.</li> <li>2. Die Hand hält das Kabel.</li> <li>3. Das Kabel wurde über eine bestimmte Distanz weggezogen.</li> </ol>
<b>Akzeptanzkriterium</b>	<ol style="list-style-type: none"> <li>1. Das Kabel ist entfernt</li> <li>2. Der Strom wird nicht mehr weitergeleitet</li> </ol>
<b>Audiofeedback</b>	Snap-Geräusch

Tabelle 52: Funktionale Anforderung - Verbundenes Kabel wird entfernt

## 5.8 Stromschalter

In diesem Abschnitt werden die Use-Cases (sowie deren funktionale Anforderungen) zum Stromschalter näher beschrieben.

### 5.8.1 Strom einschalten

<b>Use Case ID</b>	<b>UC-18</b>
<b>Name des Use Cases</b>	Strom einschalten
<b>Kurzbeschreibung</b>	Der Strom wird über einen Schalter eingeschaltet.
<b>Priorität</b>	Hoch
<b>Akteur</b>	Spieler
<b>Vorbedingungen</b>	1. Der Schalter auf Position 0.
<b>Resultat</b>	Der Strom wird weitergeleitet.
<b>Hauptszenario</b>	1. Der Spieler kippt den Schalter von 0 auf 1.
<b>Abweichungen</b>	-

Tabelle 53: Use-Case - Strom einschalten

#### 5.8.1.1 Schalter wird von 0 auf 1 gekippt

<b>Anforderungs ID</b>	<b>FA-18-01</b>
<b>Name der Anforderung</b>	Schalter wird von 0 auf 1 gekippt
<b>Kurzbeschreibung</b>	Der Schalter wird von 0 auf 1 gekippt.
<b>Priorität</b>	Hoch
<b>Vorbedingungen</b>	1. Der Schalter ist auf Position 0.
<b>Akzeptanzkriterium</b>	1. Der Schalter ist auf Position 1
<b>Audiofeedback</b>	Switch-Sound

Tabelle 54: Funktionale Anforderung - Schalter wird von 0 auf 1 gekippt

#### 5.8.1.2 Strom auf verbundene Objekte weitergeben

<b>Anforderungs ID</b>	<b>FA-18-02</b>
<b>Name der Anforderung</b>	Strom auf verbundene Objekte weitergeben
<b>Kurzbeschreibung</b>	Der Strom wird anhand der Schalterposition 1 weitergegeben und grün eingefärbt.
<b>Priorität</b>	Hoch
<b>Vorbedingungen</b>	1. Ein Kabel zwischen dem Strom-Schalter und einem Element-Eingang ist platziert.
<b>Akzeptanzkriterium</b>	1. Am Element-Eingang kommt der Strom richtig an. 2. Das Kabel ist grün eingefärbt.
<b>Audiofeedback</b>	-

Tabelle 55: Funktionale Anforderung - Strom auf verbundene Objekte weitergeben

## 5.8.2 Strom ausschalten

<b>Use Case ID</b>	<b>UC-19</b>
<b>Name des Use Cases</b>	Strom ausschalten
<b>Kurzbeschreibung</b>	Der Strom wird über einen Schalter ausgeschaltet.
<b>Priorität</b>	Hoch
<b>Akteur</b>	Spieler
<b>Vorbedingungen</b>	1. Der Schalter auf Position 1.
<b>Resultat</b>	Kein Strom wird weitergeleitet.
<b>Hauptszenario</b>	1. Der Spieler kippt den Schalter von 0 auf 1.
<b>Abweichungen</b>	-

Tabelle 56: Use-Case - Strom ausschalten

### 5.8.2.1 Schalter wird von 1 auf 0 gekippt

<b>Anforderungs ID</b>	<b>FA-19-01</b>
<b>Name der Anforderung</b>	Schalter wird von 0 auf 1 gekippt
<b>Kurzbeschreibung</b>	Der Schalter wird von 0 auf 1 gekippt.
<b>Priorität</b>	Hoch
<b>Vorbedingungen</b>	1. Der Schalter ist auf Position 0.
<b>Akzeptanzkriterium</b>	1. Der Schalter ist auf Position 1.
<b>Audiofeedback</b>	Switch-Sound

Tabelle 57: Funktionale Anforderung - Schalter wird von 1 auf 0 gekippt

### 5.8.2.2 Strom auf verbundene Objekte nicht weitergeben

<b>Anforderungs ID</b>	<b>FA-19-02</b>
<b>Name der Anforderung</b>	Strom auf verbundene Objekte nicht weitergeben
<b>Kurzbeschreibung</b>	Der Strom wird anhand der Schalterposition 0 nicht weitergegeben und das Kabel wird rot eingefärbt.
<b>Priorität</b>	Hoch
<b>Vorbedingungen</b>	1. Ein Kabel zwischen dem Strom-Schalter und einem Element-Eingang ist platziert.
<b>Akzeptanzkriterium</b>	1. Am Element-Eingang kommt der Strom nicht an. 2. Das Kabel wird rot eingefärbt.
<b>Audiofeedback</b>	-

Tabelle 58: Funktionale Anforderung - Strom auf verbundene Objekte nicht weitergeben

## 5.9 Notizen

In diesem Abschnitt werden die Use-Cases (sowie deren funktionale Anforderungen) zu den Notizen näher beschrieben.

### 5.9.1 Kreide halten

<b>Use Case ID</b>	<b>UC-20</b>
<b>Name des Use Cases</b>	Kreide halten
<b>Kurzbeschreibung</b>	Der Spieler kann eine Kreide mit den Händen aufnehmen, halten und loslassen.
<b>Priorität</b>	Tief
<b>Akteur</b>	Spieler
<b>Vorbedingungen</b>	1. Es existiert eine Kreide im Raum.
<b>Resultat</b>	Der Spieler hält Kreide in der Hand.
<b>Hauptszenario</b>	1. Der Spieler berührt mit der Hand die Kreide. 2. Der Spieler schliesst die Hand. 3. Der Spieler bewegt die Hand. 4. Der Spieler öffnet die Hand.
<b>Abweichungen</b>	-

Tabelle 59: Use-Case - Kreide halten

#### 5.9.1.1 Kreide wird in der Hand gehalten

<b>Anforderungs ID</b>	<b>FA-20-01</b>
<b>Name der Anforderung</b>	Kreide wird gehalten
<b>Kurzbeschreibung</b>	Die Kreide wird durch das Drücken und gedrückt halten des Axis1D.PrimaryHandTrigger / Axis1DSecondaryHandTrigger in der Hand gehalten.
<b>Priorität</b>	Tief
<b>Vorbedingungen</b>	1. Die Hand ist frei.
<b>Akzeptanzkriterium</b>	1. Die Kreide wird in der gewünschten Hand gehalten.
<b>Audiofeedback</b>	-

Tabelle 60: Funktionale Anforderung - Kreide wird in der Hand gehalten

### 5.9.1.2 Kreide wird losgelassen

<b>Anforderungs ID</b>	<b>FA-20-02</b>
<b>Name der Anforderung</b>	Kreide wird losgelassen
<b>Kurzbeschreibung</b>	Die Kreide wird durch das Loslassen des Axis1D.PrimaryHandTrigger / Axis1DSecondaryHandTrigger losgelassen.
<b>Priorität</b>	Hoch
<b>Vorbedingungen</b>	1. Eine Hand hält die Kreide.
<b>Akzeptanzkriterium</b>	1. Die Kreide ist nicht mehr in der Hand.
<b>Audiofeedback</b>	-

Tabelle 61: Funktionale Anforderung - Kreide wird losgelassen

### 5.9.2 Striche mit Kreide platzieren

<b>Use Case ID</b>	<b>UC-21</b>
<b>Name des Use Cases</b>	Striche mit Kreide platzieren
<b>Kurzbeschreibung</b>	Der Spieler kann mit einer Kreide Striche auf einer Wandtafel platzieren.
<b>Priorität</b>	Tief
<b>Akteur</b>	Spieler
<b>Vorbedingungen</b>	1. Es existiert eine Wandtafel im Raum. 2. Die Hand hält die Kreide. 3. Die Kreide berührt die Wandtafel.
<b>Resultat</b>	Ein Strich ist auf der Wandtafel ersichtlich.
<b>Hauptszenario</b>	1. Der Spieler nimmt die Kreide in die Hand. 2. Der Spieler bewegt die Hand zu der Wandtafel. 3. Der Spieler bewegt die Hand über die Wandtafel.
<b>Abweichungen</b>	-

Tabelle 62: Use-Case - Striche mit Kreide platzieren

### 5.9.2.1 Kreide platziert Striche zwischen Berührungspunkten mit Wandtafel

<b>Anforderungs ID</b>	<b>FA-21-01</b>
<b>Name der Anforderung</b>	Kreide platziert Striche zwischen Berührungspunkten mit Wandtafel
<b>Kurzbeschreibung</b>	Mit der Kreide wird auf die Wandtafel gezeichnet.
<b>Priorität</b>	Tief
<b>Vorbedingungen</b>	<ol style="list-style-type: none"> <li>1. Eine Hand hält die Kreide.</li> <li>2. Die Kreide berührt die Wandtafel.</li> </ol>
<b>Akzeptanzkriterium</b>	<ol style="list-style-type: none"> <li>1. Die Striche sind zwischen den Berührungs punkten platziert.</li> <li>2. Die Striche bleiben auf der Wandtafel bestehen.</li> </ol>
<b>Audiofeedback</b>	Kreide-Sound

Tabelle 63: Funktionale Anforderung - Kreide platziert Striche zwischen Berührungs punkten mit Wandtafel

### 5.9.3 Schwamm halten

<b>Use Case ID</b>	<b>UC-22</b>
<b>Name des Use Cases</b>	Schwamm halten
<b>Kurzbeschreibung</b>	Der Spieler kann den Schwamm mit den Händen aufnehmen, halten und loslassen.
<b>Priorität</b>	Tief
<b>Akteur</b>	Spieler
<b>Vorbedingungen</b>	<ol style="list-style-type: none"> <li>1. Es existiert ein Schwamm im Raum.</li> </ol>
<b>Resultat</b>	Der Spieler hält Schwamm in der Hand.
<b>Hauptszenario</b>	<ol style="list-style-type: none"> <li>1. Der Spieler berührt mit der Hand den Schwamm.</li> <li>2. Der Spieler schliesst die Hand.</li> <li>3. Der Spieler bewegt die Hand.</li> <li>4. Der Spieler öffnet die Hand.</li> </ol>
<b>Abweichungen</b>	-

Tabelle 64: Use-Case - Schwamm halten

### 5.9.3.1 Schwamm wird gehalten

<b>Anforderungs ID</b>	<b>FA-22-01</b>
<b>Name der Anforderung</b>	Schwamm wird gehalten
<b>Kurzbeschreibung</b>	Der Schwamm wird durch das Drücken und gedrückt halten des Axis1D.PrimaryHandTrigger / Axis1DSecondaryHandTrigger in der Hand gehalten.
<b>Priorität</b>	Tief
<b>Vorbedingungen</b>	1. Die Hand ist frei.
<b>Akzeptanzkriterium</b>	1. Der Schwamm wird in der gewünschten Hand gehalten.
<b>Audiofeedback</b>	-

Tabelle 65: Funktionale Anforderung - Schwamm wird gehalten

### 5.9.3.2 Schwamm wird losgelassen

<b>Anforderungs ID</b>	<b>FA-22-02</b>
<b>Name der Anforderung</b>	Gatter wird losgelassen
<b>Kurzbeschreibung</b>	Der Schwamm wird durch das Loslassen des Axis1D.PrimaryHandTrigger / Axis1DSecondaryHandTrigger losgelassen.
<b>Priorität</b>	Tief
<b>Vorbedingungen</b>	1. Eine Hand hält das Gatter.
<b>Akzeptanzkriterium</b>	1. Der Schwamm ist nicht mehr in der Hand.
<b>Audiofeedback</b>	-

Tabelle 66: Funktionale Anforderung - Schwamm wird losgelassen

#### 5.9.4 Striche mit Schwamm entfernen

<b>Use Case ID</b>	<b>UC-23</b>
<b>Name des Use Cases</b>	Striche mit Schwamm entfernen
<b>Kurzbeschreibung</b>	Der Spieler kann mit einem Schwamm Striche auf der Wandtafel entfernen.
<b>Priorität</b>	Tief
<b>Akteur</b>	Spieler
<b>Vorbedingungen</b>	<ol style="list-style-type: none"> <li>1. Die Hand hält den Schwamm.</li> <li>2. Es befinden sich Striche auf der Wandtafel.</li> <li>3. Der Schwamm berührt die Striche.</li> </ol>
<b>Resultat</b>	Der gewünschte Strich wird entfernt.
<b>Hauptszenario</b>	<ol style="list-style-type: none"> <li>1. Der Spieler nimmt den Schwamm in die Hand.</li> <li>2. Der Spieler bewegt die Hand zu der Wandtafel.</li> <li>3. Der Spieler bewegt die Hand über die Striche auf der Wandtafel.</li> </ol>
<b>Abweichungen</b>	-

Tabelle 67: Use-Case - Striche mit Schwamm entfernen

##### 5.9.4.1 Schwamm entfernt Striche zwischen Berührpunkten mit Wandtafel

<b>Anforderungs ID</b>	<b>FA-23-01</b>
<b>Name der Anforderung</b>	Schwamm entfernt Striche zwischen Berührpunkten mit Wandtafel
<b>Kurzbeschreibung</b>	Der Schwamm entfernt Striche auf der Wandtafel.
<b>Priorität</b>	Tief
<b>Vorbedingungen</b>	<ol style="list-style-type: none"> <li>1. Hand hält den Schwamm.</li> <li>2. Der Schwamm berührt die Wandtafel.</li> <li>3. Der Schwamm berührt den Strich.</li> </ol>
<b>Akzeptanzkriterium</b>	1. Die Striche zwischen den Berührpunkten sind entfernt.
<b>Audiofeedback</b>	Wisch-Sound

Tabelle 68: Funktionale Anforderung - Schwamm entfernt Striche zwischen Berührpunkten mit Wandtafel

## 5.10 Board

In diesem Abschnitt werden die Use-Cases (sowie deren funktionale Anforderungen) zum Board näher beschrieben.

### 5.10.1 Board leeren

<b>Use Case ID</b>	<b>UC-24</b>
<b>Name des Use Cases</b>	Board leeren
<b>Kurzbeschreibung</b>	Board wird mit Drücken eines Knopfes auf dem Board geleert.
<b>Priorität</b>	Hoch
<b>Akteur</b>	Spieler
<b>Vorbedingungen</b>	1. Das Board ist nicht leer, d.h. mindestens ein Objekt auf dem Board.
<b>Resultat</b>	Das Board ist leer.
<b>Hauptszenario</b>	1. Der Spieler drückt auf den Knopf, um das Board zu leeren.
<b>Abweichungen</b>	-

Tabelle 69: Use-Case - Board leeren

#### 5.10.1.1 Board wird geleert

<b>Anforderungs ID</b>	<b>FA-24-01</b>
<b>Name der Anforderung</b>	Board wird geleert
<b>Kurzbeschreibung</b>	Das Board wird über Knopfdruck geleert.
<b>Priorität</b>	Mittel
<b>Vorbedingungen</b>	1. Das Board ist nicht leer, d.h. mindestens ein Objekt auf dem Board.
<b>Akzeptanzkriterium</b>	1. Das Board ist leer. 2. Die Objekte auf dem Board sind gelöscht.
<b>Audiofeedback</b>	Lösch-Sound

Tabelle 70: Funktionale Anforderung - Board wird geleert

## 5.10.2 Pseudo-Gatter erstellen

<b>Use Case ID</b>	<b>UC-25</b>
<b>Name des Use Cases</b>	Pseudo-Gatter erstellen
<b>Kurzbeschreibung</b>	Das Pseudo-Gatter wird aus der gelegten Gatter-Konstellation auf dem Board erstellt.
<b>Priorität</b>	Mittel
<b>Akteur</b>	Spieler
<b>Vorbedingungen</b>	1. Eine valide Konstellation von Gattern liegt auf dem Board.
<b>Resultat</b>	Ein Pseudo-Gatter ist erstellt mit ausgewählten Gatter-Label.
<b>Hauptszenario</b>	<ol style="list-style-type: none"> <li>1. Der Spieler legt gewünschte Konstellation auf das Board.</li> <li>2. Der Spieler wählt Gatter-Label aus.</li> <li>3. Der Spieler drückt den entsprechenden Knopf zur Erstellung.</li> <li>4. Das Gatter wird im Automaten hinterlegt.</li> </ol>
<b>Abweichungen</b>	-

Tabelle 71: Use-Case - Pseudo-Gatter erstellen

### 5.10.2.1 Pseudo-Gatter ist erstellt

<b>Anforderungs ID</b>	<b>FA-25-01</b>
<b>Name der Anforderung</b>	Pseudo-Gatter ist erstellt
<b>Kurzbeschreibung</b>	Das Pseudo-Gatter wird über Knopfdruck erstellt.
<b>Priorität</b>	Mittel
<b>Vorbedingungen</b>	1. Eine gültige Konstellation von Gattern liegt auf dem Board.
<b>Akzeptanzkriterium</b>	<ol style="list-style-type: none"> <li>1. Das Pseudo-Gatter wurde mit der richtigen Konfiguration erstellt.</li> <li>2. Das Pseudo-Gatter kann wie ein normales Gatter verwendet werden.</li> </ol>
<b>Audiofeedback</b>	Maschinen-Sound

Tabelle 72: Funktionale Anforderung - Pseudo Gatter ist erstellt

### 5.10.3 Label für Pseudo-Gatter bestimmen

<b>Use Case ID</b>	<b>UC-26</b>
<b>Name des Use Cases</b>	Label für Pseudo-Gatter bestimmen
<b>Kurzbeschreibung</b>	Das Label für ein Pseudo-Gatter wird über Pfeiltasten auf dem Board bestimmt.
<b>Priorität</b>	Mittel
<b>Akteur</b>	Spieler
<b>Vorbedingungen</b>	-
<b>Resultat</b>	Das Label ist bestimmt.
<b>Hauptszenario</b>	1. Der Spieler bestimmt Label über Pfeiltasten.
<b>Abweichungen</b>	-

Tabelle 73: Use-Case - Label für Pseudo-Gatter bestimmen

#### 5.10.3.1 Gatter-Label eines Pseudo-Gatter wird durch Pfeiltasten ausgewählt

<b>Anforderungs ID</b>	<b>FA-26-01</b>
<b>Name der Anforderung</b>	Gatter-Label eines Pseudo-Gatter wird durch Pfeiltasten ausgewählt
<b>Kurzbeschreibung</b>	Vierstelliges Label durch Pfeiltasten bestimmen.
<b>Priorität</b>	Mittel
<b>Vorbedingungen</b>	1. Eine gültige Konstellation von Gattern liegt auf dem Board.
<b>Akzeptanzkriterium</b>	1. Die Pfeiltasten wählen Buchstaben der Reihe nach aus. 2. Alle vier Buchstaben sind gesetzt.
<b>Audiofeedback</b>	Button-Sound

Tabelle 74: Funktionale Anforderung - Gatter-Label eines Pseudo-Gatter wird durch Pfeiltasten ausgewählt

## 5.11 Wandtafel Use-Cases

In diesem Abschnitt werden die Use-Cases (sowie deren funktionale Anforderungen) zu der Wandtafel näher beschrieben.

### 5.11.1 Aufgabe ablesen

<b>Use Case ID</b>	<b>UC-27</b>
<b>Name des Use Cases</b>	Aufgabe ablesen
<b>Kurzbeschreibung</b>	Der Spieler kann die aktuelle Aufgabe auf einer Wandtafel ablesen.
<b>Priorität</b>	Hoch
<b>Akteur</b>	Spieler
<b>Vorbedingungen</b>	1. Der Spieler befindet sich im Spiel. 2. Die Aufgabe wird an der Wandtafel angezeigt.
<b>Resultat</b>	Der Spieler liest Aufgabe.
<b>Hauptszenario</b>	1. Der Spieler richtet sich zur Wandtafel aus. 2. Der Spieler liest Aufgabe an der Wandtafel.
<b>Abweichungen</b>	-

Tabelle 75: Use-Case - Aufgabe ablesen

#### 5.11.1.1 Aktuelle Aufgabe wird auf der Wandtafel angezeigt

<b>Anforderungs ID</b>	<b>FA-27-01</b>
<b>Name der Anforderung</b>	Aktuelle Aufgabe wird auf der Wandtafel angezeigt
<b>Kurzbeschreibung</b>	Die aktuelle Aufgabe wird auf der Wandtafel mit dem dazugehörigen Text und Bild angezeigt.
<b>Priorität</b>	Hoch
<b>Vorbedingungen</b>	1. Der Spieler befindet sich im Spiel.
<b>Akzeptanzkriterium</b>	1. Der Text und das Bild der Aufgabe werden auf der Wandtafel angezeigt.
<b>Audiofeedback</b>	-

Tabelle 76: Funktionale Anforderung - Aktuelle Aufgabe wird auf der Wandtafel angezeigt

## 5.11.2 Aufgabe überprüfen

<b>Use Case ID</b>	<b>UC-28</b>
<b>Name des Use Cases</b>	Aufgabe überprüfen
<b>Kurzbeschreibung</b>	Der Spieler kann die aktuelle Aufgabe auf der Wandtafel mit einem Knopf überprüfen lassen.
<b>Priorität</b>	Hoch
<b>Akteur</b>	Spieler
<b>Vorbedingungen</b>	1. Die Aufgabe ist gestellt
<b>Resultat</b>	Der Spieler wird über Audio-Feedback auf den Erfolg der Aufgabe hingewiesen. Wurde Die Aufgabe richtig gelöst, wird auf der Wandtafel eine neue Folie angezeigt.
<b>Hauptszenario</b>	<ol style="list-style-type: none"> <li>1. Der Spieler löst Aufgabe.</li> <li>2. Der Spieler klickt auf den Überprüf-Knopf auf der Wandtafel.</li> <li>3. Der Spieler wird über den Erfolg informiert.</li> <li>4. Der Spieler fährt fort.</li> </ol>
<b>Abweichungen</b>	1. Der Spieler probiert weiter an der Aufgabe.

Tabelle 77: Use-Case - Aufgabe überprüfen

### 5.11.2.1 Check-Button wird angezeigt

<b>Anforderungs ID</b>	<b>FA-28-01</b>
<b>Name der Anforderung</b>	Check-Button wird angezeigt
<b>Kurzbeschreibung</b>	Der Check-Button wird anstelle des Weiter-Buttons angezeigt.
<b>Priorität</b>	Hoch
<b>Vorbedingungen</b>	<ol style="list-style-type: none"> <li>1. Der Spieler befindet sich im Spiel.</li> <li>2. Eine Aufgabe wird auf der Wandtafel angezeigt.</li> </ol>
<b>Akzeptanzkriterium</b>	<ol style="list-style-type: none"> <li>1. Der Weiter-Button wird nicht angezeigt.</li> <li>2. Der Check-Button wird angezeigt.</li> </ol>
<b>Audiofeedback</b>	-

Tabelle 78: Funktionale Anforderung - Check-Button wird angezeigt

### 5.11.2.2 Falsch gelöste Aufgabe

<b>Anforderungs ID</b>	<b>FA-28-02</b>
<b>Name der Anforderung</b>	Falsch gelöste Aufgabe
<b>Kurzbeschreibung</b>	Die aktuelle Konstellation auf dem Board löst die aktuelle Aufgabe nicht.
<b>Priorität</b>	Hoch
<b>Vorbedingungen</b>	<ol style="list-style-type: none"> <li>1. Der Spieler befindet sich im Spiel.</li> <li>2. Eine Aufgabe wird auf der Wandtafel angezeigt.</li> <li>3. Die Konstellation auf dem Board löst die aktuelle Aufgabe nicht.</li> </ol>
<b>Akzeptanzkriterium</b>	1. Die aktuelle Aufgabe wird weiterhin angezeigt.
<b>Audiofeedback</b>	Wrong Answer

Tabelle 79: Funktionale Anforderung - Falsch gelöste Aufgabe

### *5.11.2.3 Korrekt gelöste Aufgabe*

<b>Anforderungs ID</b>	<b>FA-28-03</b>
<b>Name der Anforderung</b>	Korrekt gelöste Aufgabe
<b>Kurzbeschreibung</b>	Die aktuelle Konstellation auf dem Board löst die aktuelle Aufgabe.
<b>Priorität</b>	Hoch
<b>Vorbedingungen</b>	<ol style="list-style-type: none"><li>1. Der Spieler befindet sich im Spiel.</li><li>2. Eine Aufgabe wird auf der Wandtafel angezeigt.</li><li>3. Die Konstellation auf dem Board löst die aktuelle Aufgabe.</li></ol>
<b>Akzeptanzkriterium</b>	1. Die nächste Folie wird auf der Wandtafel angezeigt.
<b>Audiofeedback</b>	Right Answer

*Tabelle 80: Funktionale Anforderung - Richtig gelöste Aufgabe*

### 5.11.3 Navigieren

<b>Use Case ID</b>	<b>UC-29</b>
<b>Name des Use Cases</b>	Navigieren
<b>Kurzbeschreibung</b>	Spieler navigiert auf der Wandtafel hin und her mit den entsprechenden Knöpfen.
<b>Priorität</b>	Hoch
<b>Akteur</b>	Spieler
<b>Vorbedingungen</b>	1. Die Wandtafel zeigt Aufgaben an.
<b>Resultat</b>	Der Spieler gelangt zur nächsten / vorherigen Folie.
<b>Hauptszenario</b>	1. Der Spieler klickt auf «Next». 2. Die nächste Folie wird angezeigt.
<b>Abweichungen</b>	1. Der Spieler klickt auf «Previous». 2. Die vorherige Folie wird angezeigt.

Tabelle 81: Use-Case - Navigieren

#### 5.11.3.1 Weiter navigieren

<b>Anforderungs ID</b>	<b>FA-29-01</b>
<b>Name der Anforderung</b>	Weiter navigieren
<b>Kurzbeschreibung</b>	Der Spieler kann auf die nächste Folie durch einen Klick auf den Weiter-Button navigieren.
<b>Priorität</b>	Hoch
<b>Vorbedingungen</b>	1. Der Spieler befindet sich im Spiel. 2. Die aktuelle Folie besitzt einen Weiter-Button.
<b>Akzeptanzkriterium</b>	1. Die nächste Folie wird auf der Wandtafel angezeigt.
<b>Audiofeedback</b>	-

Tabelle 82: Funktionale Anforderung - Weiter navigieren

#### 5.11.3.2 Zurück navigieren

<b>Anforderungs ID</b>	<b>FA-29-02</b>
<b>Name der Anforderung</b>	Zurück navigieren
<b>Kurzbeschreibung</b>	Der Spieler kann auf die vorherige Folie durch einen Klick auf den Weiter-Button navigieren.
<b>Priorität</b>	Hoch
<b>Vorbedingungen</b>	1. Der Spieler befindet sich im Spiel. 2. Die aktuelle Folie besitzt einen Zurück-Button.
<b>Akzeptanzkriterium</b>	1. Die vorherige Folie wird auf der Wandtafel angezeigt.
<b>Audiofeedback</b>	-

Tabelle 83: Funktionale Anforderung - Zurück navigieren

#### 5.11.4 Hinweis / Lösung beziehen

<b>Use Case ID</b>	<b>UC-30</b>
<b>Name des Use Cases</b>	Hinweis oder Lösung beziehen
<b>Kurzbeschreibung</b>	Der Spieler kann auf eine anderen Wandtafel Hinweise und Lösungen zur aktuellen Aufgabe anzeigen lassen.
<b>Priorität</b>	Mittel
<b>Akteur</b>	Spieler
<b>Vorbedingungen</b>	1. Der Spieler befindet sich in einer Aufgabe.
<b>Resultat</b>	Der Spieler sieht auf der Wandtafel den Hinweis / die Lösung.
<b>Hauptszenario</b>	1. Der Spieler erkaufst sich Hinweis / Lösung. 2. Ein Hinweis/Lösung zur aktuellen Aufgabe wird angezeigt.
<b>Abweichungen</b>	-

Tabelle 84: Use-Case - Hinweise / Lösung beziehen

#### 5.11.4.1 Kosten von Score abziehen

<b>Anforderungs ID</b>	<b>FA-30-01</b>
<b>Name der Anforderung</b>	Kosten von Score abziehen
<b>Kurzbeschreibung</b>	Sobald der Spieler einen Hinweis zur aktuellen Aufgabe bezieht, werden die Kosten dazu von dem aktuellen Score abgezogen.
<b>Priorität</b>	Hoch
<b>Vorbedingungen</b>	1. Der Spieler befindet sich im Spiel. 2. Der Spieler will einen Hinweis oder eine Lösung anzeigen.
<b>Akzeptanzkriterium</b>	1. Die Kosten wurden vom aktuellen Score abgezogen.
<b>Audiofeedback</b>	-

Tabelle 85: Funktionale Anforderung - Kosten von Score abziehen

#### 5.11.4.2 Nächsten Hinweis / Lösung anzeigen

<b>Anforderungs ID</b>	<b>FA-30-02</b>
<b>Name der Anforderung</b>	Nächsten Hinweis / Lösung anzeigen
<b>Kurzbeschreibung</b>	Sobald der Spieler einen Hinweis zur aktuellen Aufgabe bezieht, wird dieser auf der Wandtafel angezeigt.
<b>Priorität</b>	Hoch
<b>Vorbedingungen</b>	1. Der Spieler befindet sich im Spiel. 2. Die aktuelle Folie besitzt einen Zurück-Button.
<b>Akzeptanzkriterium</b>	1. Der Hinweis wird auf der Wandtafel angezeigt.
<b>Audiofeedback</b>	-

Tabelle 86: Funktionale Anforderung - Nächsten Hinweis / Lösung anzeigen

#### *5.11.4.3 Bezahlte Hinweise / Lösungen nicht erneut bezahlen*

<b>Anforderungs ID</b>	<b>FA-30-03</b>
<b>Name der Anforderung</b>	Bezahlte Hinweise / Lösungen nicht erneut bezahlen
<b>Kurzbeschreibung</b>	Falls ein Spieler einen Hinweis bereits bezahlt hat, kann er diesen ohne weitere Kosten erneut anzeigen lassen.
<b>Priorität</b>	Hoch
<b>Vorbedingungen</b>	<ol style="list-style-type: none"> <li>1. Der Spieler befindet sich im Spiel.</li> <li>2. Der Spieler will einen bereits bezahlten Hinweis oder Lösung anzeigen lassen.</li> </ol>
<b>Akzeptanzkriterium</b>	<ol style="list-style-type: none"> <li>1. Der Hinweis wird auf der Wandtafel angezeigt.</li> <li>2. Die Kosten wurden nicht erneut vom aktuellen Score abgezogen.</li> </ol>
<b>Audiofeedback</b>	-

*Tabelle 87: Funktionale Anforderung - Bezahlte Hinweise / Lösungen nicht erneut bezahlen*

## 5.12 Automat

In diesem Abschnitt werden die Use-Cases (sowie deren funktionale Anforderungen) zum Automaten näher beschrieben.

### 5.12.1 Gatter auswählen

<b>Use Case ID</b>	<b>UC-31</b>
<b>Name des Use Cases</b>	Gatter auswählen
<b>Kurzbeschreibung</b>	Gatter kann über Pfeiltasten ausgewählt werden.
<b>Priorität</b>	Hoch
<b>Akteur</b>	Spieler
<b>Vorbedingungen</b>	<ol style="list-style-type: none"> <li>1. Ein Gatter steht im Automaten zur Verfügung.</li> <li>2. Der Spieler steht vor dem Automaten.</li> </ol>
<b>Resultat</b>	Der Gatter-Name wird im Bildschirm des Automaten angezeigt.
<b>Hauptszenario</b>	<ol style="list-style-type: none"> <li>1. Der Spieler drückt auf die Pfeiltaste zum Navigieren.</li> <li>2. Der Spieler sieht den Namen des gewünschten Gatters im Anzeigefenster des Automaten.</li> </ol>
<b>Abweichungen</b>	-

Tabelle 88: Use-Case - Gatter auswählen

#### 5.12.1.1 Vorwärts navigieren

<b>Anforderungs ID</b>	<b>FA-31-01</b>
<b>Name der Anforderung</b>	Vorwärts navigieren
<b>Kurzbeschreibung</b>	Der Spieler drückt auf den rechten Pfeil und wählt damit das nächste Gatter im Automaten aus.
<b>Priorität</b>	Hoch
<b>Vorbedingungen</b>	<ol style="list-style-type: none"> <li>1. Der Spieler befindet sich im Spiel.</li> <li>2. Der Spieler drückt auf «→»-Knopf.</li> </ol>
<b>Akzeptanzkriterium</b>	<ol style="list-style-type: none"> <li>1. Das nächste Gatter wird ausgewählt.</li> <li>2. Der Name des nächsten Gatters wird angezeigt.</li> </ol>
<b>Audiofeedback</b>	-

Tabelle 89: Funktionale Anforderung - Vorwärts navigieren

### *5.12.1.2 Rückwärts navigieren*

<b>Anforderungs ID</b>	<b>FA-31-02</b>
<b>Name der Anforderung</b>	Rückwärts navigieren
<b>Kurzbeschreibung</b>	Der Spieler drückt auf den linken Pfeil und wählt damit das vorherige Gatter im Automaten aus.
<b>Priorität</b>	Hoch
<b>Vorbedingungen</b>	<ol style="list-style-type: none"><li>1. Der Spieler befindet sich im Spiel.</li><li>2. Der Spieler drückt auf «←»-Knopf.</li></ol>
<b>Akzeptanzkriterium</b>	<ol style="list-style-type: none"><li>1. Das vorherige Gatter wird ausgewählt.</li><li>2. Der Name des vorherigen Gatters wird angezeigt.</li></ol>
<b>Audiofeedback</b>	-

*Tabelle 90: Funktionale Anforderung - Rückwärts navigieren*

### 5.12.2 Gatter beziehen

<b>Use Case ID</b>	<b>UC-32</b>
<b>Name des Use Cases</b>	Gatter beziehen
<b>Kurzbeschreibung</b>	Gatter wird über einen Knopf im Automaten ausgegeben.
<b>Priorität</b>	Hoch
<b>Akteur</b>	Spieler
<b>Vorbedingungen</b>	<ol style="list-style-type: none"> <li>1. Das Gatter steht im Automaten zur Verfügung.</li> <li>2. Der Spieler steht vor dem Automaten.</li> </ol>
<b>Resultat</b>	Der Spieler erhält Gatter aus dem Automaten.
<b>Hauptszenario</b>	<ol style="list-style-type: none"> <li>1. Der Spieler drückt auf «GET»-Knopf .</li> <li>2. Das neue Gatter erscheint.</li> </ol>
<b>Abweichungen</b>	-

Tabelle 91: Use-Case - Gatter beziehen

#### 5.12.2.1 Neue Gatter-Instanz erscheint

<b>Anforderungs ID</b>	<b>FA-32-01</b>
<b>Name der Anforderung</b>	Neue Gatter-Instanz erscheint
<b>Kurzbeschreibung</b>	Eine neue Instanz des aktuell ausgewählten Gatters erscheint.
<b>Priorität</b>	Mittel
<b>Vorbedingungen</b>	<ol style="list-style-type: none"> <li>1. Der Spieler befindet sich im Spiel.</li> <li>2. Der Spieler drückt auf «GET»-Knopf.</li> </ol>
<b>Akzeptanzkriterium</b>	1. Die neue Gatter-Instanz erscheint.
<b>Audiofeedback</b>	-

Tabelle 92: Funktional Anforderung - Neue Gatter-Instanz erscheint

## 5.13 High-Score

In diesem Abschnitt werden die Use-Cases (sowie deren funktionale Anforderungen) zur High-Score näher beschrieben.

### 5.13.1 High-Score ablesen

<b>Use Case ID</b>	<b>UC-33</b>
<b>Name des Use Cases</b>	High-Score ablesen
<b>Kurzbeschreibung</b>	Der Spieler kann neben dem Board auf dem Tisch den High-Score ablesen.
<b>Priorität</b>	Mittel
<b>Akteur</b>	Spieler
<b>Vorbedingungen</b>	1. Die High-Score ist gesetzt.
<b>Resultat</b>	Der Spieler liest High-Score.
<b>Hauptszenario</b>	1. Der Spieler liest High-Score.
<b>Abweichungen</b>	-

Tabelle 93: Use-Case - High-Score ablesen

#### 5.13.1.1 High-Score wird auf Tisch angezeigt

<b>Anforderungs ID</b>	<b>FA-33-01</b>
<b>Name der Anforderung</b>	High-Score wird auf Tisch angezeigt
<b>Kurzbeschreibung</b>	Die aktuelle High-Score wird auf dem Tisch im Labor angezeigt.
<b>Priorität</b>	Mittel
<b>Vorbedingungen</b>	1. Der Spieler befindet sich im Spiel.
<b>Akzeptanzkriterium</b>	1. Die aktuelle High-Score wird angezeigt.
<b>Audiofeedback</b>	-

Tabelle 94: Funktionale Anforderung - High-Score wird auf Tisch angezeigt

### 5.13.2 Aktueller Score ablesen

<b>Use Case ID</b>	<b>UC-34</b>
<b>Name des Use Cases</b>	Aktueller Score ablesen
<b>Kurzbeschreibung</b>	Der Spieler liest aktuellen Score auf der Wandtafel ab.
<b>Priorität</b>	Mittel
<b>Akteur</b>	Spieler
<b>Vorbedingungen</b>	1. Das Spiel ist gestartet.
<b>Resultat</b>	Der Spieler sieht aktuellen Score.
<b>Hauptszenario</b>	1. Der Spieler richtet Headset in Richtung der Wandtafel aus. 2. Der Spieler liest aktuellen Score ab.
<b>Abweichungen</b>	-

Tabelle 95: Use-Case - Aktueller Score ablesen

### *5.13.2.1 Abziehen der Punkte pro Sekunde*

<b>Anforderungs ID</b>	<b>FA-34-01</b>
<b>Name der Anforderung</b>	Abziehen der Punkte pro Sekunde
<b>Kurzbeschreibung</b>	Pro Sekunde wird vom Score automatisch ein Punkt abgezogen.
<b>Priorität</b>	Mittel
<b>Vorbedingungen</b>	1. Die erste Aufgabe ist gestartet.
<b>Akzeptanzkriterium</b>	1. Pro Sekunde wird ein Punkt vom Score abgezogen.
<b>Audiofeedback</b>	-

*Tabelle 96: Funktionale Anforderung - Abziehen der Punkte pro Sekunde*

## 6 Vorgehen / Methoden

Für die Umsetzung der Use-Cases sowie für das Dokumentieren der Arbeit hat man sich im Rahmen dieser Bachelorarbeit für eine agile Vorgehensweise mit SCRUM entschieden.

### 6.1 Projektmanagement

Um eine strukturierte Arbeit sicher zu stellen, hat man sich für das gratis Projektmanagement Tool JIRA<sup>4</sup> entschieden. Auf der JIRA Plattform werden nach Abschluss der Anforderungsanalyse alle Use-Cases als Stories in den Backlog übertragen. Die Stories werden in einem Intervall von einwöchigen Sprints geplant und abgearbeitet.

#### 6.1.1 Agile / SCRUM

SCRUM ist ein bewährtes und weit verbreitetes Framework im Softwareprojektmanagement. Es ermöglicht den Entwicklern eine iterative und flexible Arbeitsweise.

#### 6.1.2 Projektplan

Zum Beginn des Projekts wurden insgesamt sieben Meilensteine (Epics in JIRA) festgelegt und grob mit Stories und Tasks befüllt. Ein Meilenstein gilt als erledigt, sofern alle darin enthaltenen Stories abgeschlossen sind.

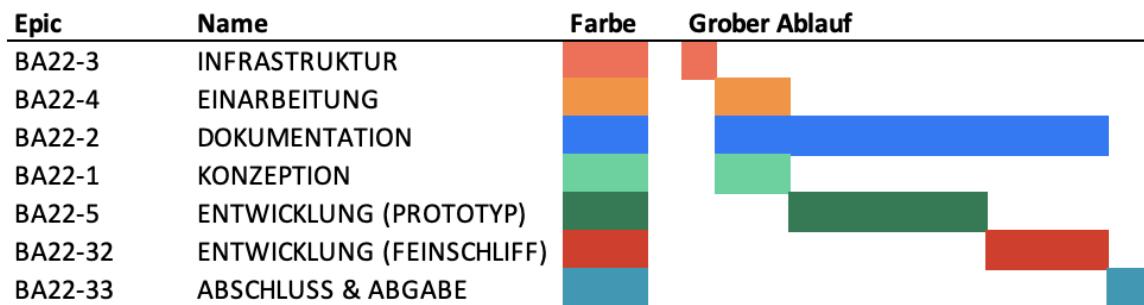


Abbildung 15: Grober Ablauf des Projekts ohne explizite Deadlines.

Der durchgeführte Projektlauf ist im Anhang unter Abschnitt A *Epic-Reports zum Projektlauf* im Detail anhand von Epic-Reports mit Stories und Diagrammen aufgeführt.

#### 6.1.3 Meilensteine

Für jeden Task und Story wird eine Aufwandschätzung in Stunden definiert. Die Summe aller Tasks und Stories in einem Meilenstein ergeben den geschätzten Aufwand des besagten Meilensteines.

<sup>4</sup> Auffindbar unter : <https://www.atlassian.com/software/jira>

#### 6.1.4 Sprint

Aus der untenstehenden Abbildung kann man den Sprintablauf eines SCRUM Sprints entnehmen.

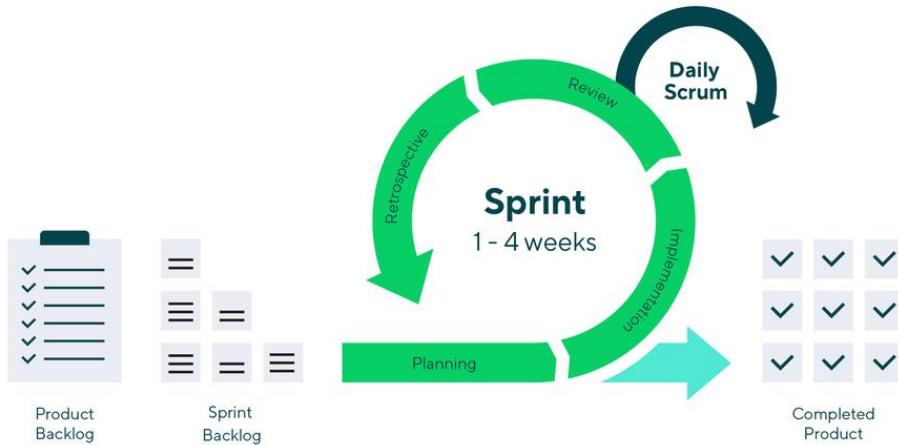


Abbildung 16: Sprint Ablauf [31]

#### 6.1.5 Daily-Standup

Daily-Standups (Daily Scrum) werden auf Grund der kurzen Sprintdauer sowie der kleinen Teamgrösse nur bei Bedarf durchgeführt. Ein Grossteil des Informationsflusses entsteht spontan und macht es dadurch überflüssig zusätzliche Meetings abzuhalten.

#### 6.1.6 Weekly Check-In

Wöchentlich werden Meetings mit dem Betreuer durchgeführt, um den aktuellen Stand des Projekts (Review Phase im Sprint) sowie organisatorische Tasks festzulegen. Zu jedem Check-In wird ein Protokoll erstellt, indem die offenen Punkte sowie die weiterführenden Arbeiten festgehalten werden.

#### 6.1.7 SCRUM / Sprint Umsetzung in diesem Projekt

Aufgrund des kleinen Teams und den wöchentlichen Statusmeetings wurde die Sprintdauer auf eine Woche festgelegt. Dadurch bestand die Möglichkeit akute Probleme oder Fehleinschätzungen zeitnah zu behandeln.

Der Backlog<sup>5</sup> wurde in zwei Schritten gefüllt, einmal grobe Stories zum Beginn der Arbeit und in einem zweiten Schritt wurde der Backlog anhand der Anforderungsanalyse komplett gefüllt. Jede Woche wurde der neue Sprint geplant (Planning Phase im Sprint) und der Sprintbacklog gefüllt. Die Stories konnten frei von den Teammitglieder abgearbeitet werden (Implementation Phase im Sprint). Auf ein Retrospective und Daily-Standups wurde verzichtet, da der Informationsfluss spontan und bei Bedarf stattgefunden hatte.

<sup>5</sup> Mehr zum Backlog: <https://www.scrum.org/resources/what-is-a-product-backlog>

## 7 Architektur

Das Serious Game besteht aus diversen Komponenten, welche auf unterschiedliche Arten miteinander kommunizieren müssen, damit die gewünschten Interaktionen funktionieren. In diesem Kapitel werden die Architekturen der einzelnen Komponenten, deren Funktionsweise und Interaktionen vorgestellt.

### 7.1 Basics

Unity stellt in ihrem Framework vorinstallierte Tools zur Verfügung, welche es dem Spieleentwickler<sup>6</sup> ermöglichen, sich vollkommen auf Features zu konzentrieren, ohne sich dabei mit technischen Aspekten eines Spiels befassen zu müssen. Beispiele dafür sind Rendering, Kollisionen, Sound, etc.

#### 7.1.1 Prefabs

Ein Prefab ist ein Game Objekt welches als Asset abgespeichert wird. Unter einem Prefab kann man sich eine Schablone vorstellen, die es dem Entwickler ermöglicht, bestimmte Konstellationen aus Game Objekten wieder zu verwenden. Es ist möglich verschachtelte Prefabs zu erstellen. Wie auf Abbildung 17 ersichtlich, wurde beim Gatter ein verschachteltes Prefab erstellt, um die Kabel darin zu platzieren.

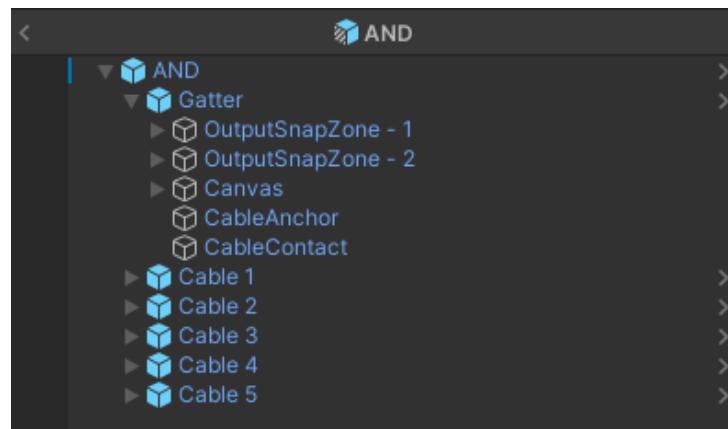


Abbildung 17: Hierarchie des AND-Gatter Prefab

Der Editor verwendet die folgenden Symbole, um Prefabs und Non-Prefabs zu unterscheiden:

Symbol	Typ
	Prefab
	Non-Prefab

Tabelle 97: Unity Prefab Symbole

<sup>6</sup> In der folgenden Arbeit wird lesbarkeitshalber nur von Entwickler gesprochen. Dieser Term versteht sich als geschlechtsneutral und meint immer Entwicklerin und Entwickler.

### 7.1.2 MonoBehaviour

Die MonoBehaviour Klasse ist eine Klasse, welche einem Objekt (Technisch: GameObject) Logik verleihen kann. Der Entwickler erbt mit seiner Klasse von MonoBehaviour und kann dadurch auf verschiedene Unity-Lifecycle Methoden zugreifen.

Eine MonoBehaviour wird generell als Komponente (Component) bezeichnet. Objekte können eine beliebige Menge an Components zugewiesen haben. Dies ermöglicht dem Entwickler die eigenen Interaktionen in mehrere, einzelne Components zu unterteilen und dadurch eine Trennung der Zuständigkeiten zu schaffen. Eine technische Einschränkung ist, dass die MonoBehaviours dadurch nur in der Szene leben können und nicht ausserhalb (z.B. von ScriptableObjects) referenziert werden können.

### 7.1.3 ScriptableObject

Die ScriptableObject Klasse ist eine Klasse, ähnlich wie die MonoBehaviour Klasse. Die beiden Klassen unterscheiden sich dadurch, dass ScriptableObjects nicht zu einem GameObject gehören (und dadurch auch nicht in der Szene leben) kann.

Durch diese Eigenschaft eignen sich ScriptableObject gut für Daten- und komplexe Konfigurationsobjekte. MonoBehaviours können ScriptableObjects referenzieren, umgekehrt ist das aber nicht möglich (Ausgenommen Prefabs).

Speziell an ScriptableObjects ist, dass diese, als Asset-File abgespeichert werden können und dadurch manuell (im Editor) vom Entwickler mit Komponenten verknüpft werden können. In Abbildung 18 sieht man die ScriptableObjects abgespeichert als Assets.

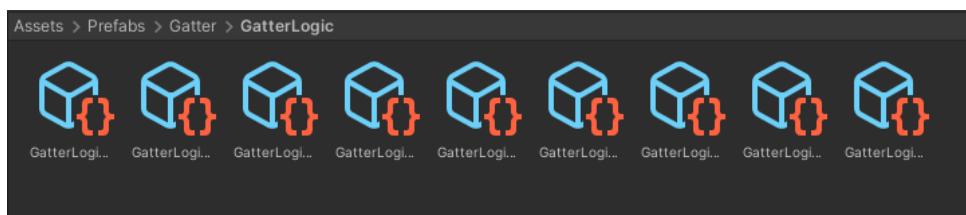


Abbildung 18: Gatter-Logiken abgespeichert als Asset

### 7.1.4 Unity-Lifecycle

Der Unity Lifecycle besteht aus diversen Methoden, welche aus MonoBehaviours und aus ScriptableObjects referenziert werden können. In der folgenden Tabelle werden die Methoden, welche im Serious Game am häufigsten verwendet werden, vorgestellt.

Methode	Kompatibilität	Beschreibung
Awake()	MonoBehaviour, ScriptableObject	Initialisierung des Objects. Normalerweise wird in diesem Schritt die Referenzierung zwischen Komponenten durchgeführt.
Start()	MonoBehaviour,	Wird vor dem ersten Frame ausgeführt. Ab diesem Schritt können Änderungen an der Ausrichtung, Position, etc. vorgenommen werden.

OnEnable()	MonoBehaviour, ScriptableObject	Wird ausgeführt sobald das Object aktiviert wird und gerendert werden soll. Wird meistens für das Registrieren von Listenern <sup>7</sup> verwendet.
OnDisable()	MonoBehaviour, ScriptableObject	Wird ausgeführt sobald das Object deaktiviert wird und nicht mehr gerendert werden soll. Wird meistens für das Deregistrieren von Listenern verwendet.
Update()	MonoBehaviour,	Wird in jedem Frame aufgerufen. Wird für kontinuierliche Änderungen an einem Object verwendet.
OnDestroy()	MonoBehaviour, ScriptableObject	Wird aufgerufen sobald ein Object komplett gelöscht werden soll.

Tabelle 98: Beschreibung der meistverwendeten Unity-Lifecycle Methoden

### 7.1.5 Collider

Das Collider Framework ermöglicht es dem Entwickler, die Kollisionen zwischen zwei Objekten zu behandeln. Es existieren zwei unterschiedliche Varianten von Kollisionen, welche in untenstehender Tabelle erkennbar sind.

Kollisionstyp	Beschreibung
Standard	Standardkollisionen sind Kollisionen wie im richtigen Leben. Beispielsweise wenn ein Ball gegen eine Wand geworfen wird, kollidiert dieser mit der Wand und prallt davon ab.
Trigger	Triggerkollisionen sind Kollision, welche keine physischen Interaktionen haben. Beispielsweise wenn überprüft werden soll, ob sich ein Objekt innerhalb eines bestimmten Bereichs befindet.

Tabelle 99: Kollisionstypen in Unity

Der Entwickler kann diverse Collider zu einem Objekt als Mesh hinzufügen. Normalerweise unterscheiden sich diese in der geometrischen Form. Würfelförmige Objekte können beispielsweise einen BoxCollider verwenden, während runde Objekte einen SphereCollider verwenden können.

---

<sup>7</sup> Mehr zu Listener: <https://www.computerhope.com/jargon/e/event-listener.htm#:~:text=The%20listener%20is%20programmed%20to,to%20as%20an%20event%20handler>.

### 7.1.6 Event-Driven Architecture (EDA)

Um hohe Flexibilität zu gewährleisten, wurde zu Beginn entschieden, dass eine durch Events gesteuerte Architektur realisiert werden soll. Diese Events beziehen sich auf technische und domain-spezifische Events. Ein Beispiel für technische Events wären Kollisionen zwischen GameObjects. Ein Beispiel für domain-spezifische Events wäre das Verbinden eines Kabels mit einem Gatter.

Unity bietet dafür die Klasse UnityEvent, welche im Editor mit einem kleinen Dialog bearbeitet werden kann. Dadurch können Events interpretiert und weitergeleitet werden, ohne diese Verbindungen selbst im Code zu realisieren. Eine Instanz von UnityEvent kann als ein universell einsetzbares Observer-Pattern betrachtet werden.

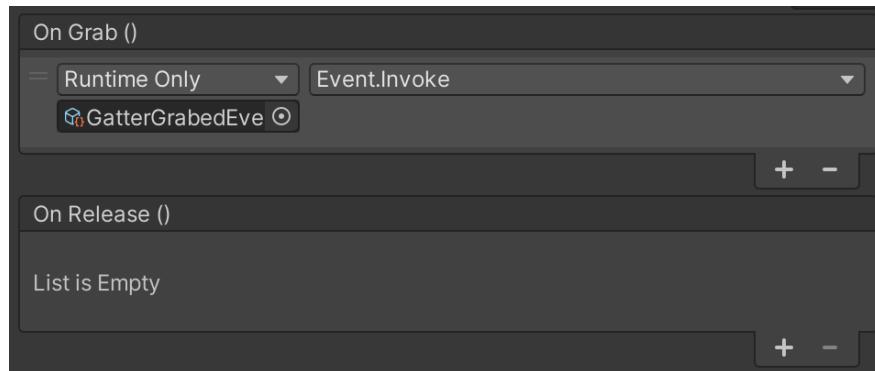


Abbildung 19: Beispiel eines UnityEvent Editor welcher öffentliche Methoden auf anderen Objekten ausführen kann

## 7.2 Headset- und Controllerinteraktionen

Von Oculus wird aktiv die Oculus Integration SDK entwickelt. Die SDK ist ein Open-Source Framework, welches die grundlegenden Interaktion mit einem Oculus Headset bereitstellt. Ein Entwickler kann dadurch ohne Entwicklungsaufwand die VR-Grundfunktionen wie die Teleportation (Bewegung in der Spielwelt) und die Interaktionen mit den Händen in sein Spiel einbauen.

### 7.2.1 Use-Cases

Die Headset- und Controllerinteraktionen wurden basierend auf den folgenden Use-Cases und funktionalen Anforderungen entwickelt (*Tabelle 100: Use-Cases der Headset- und Controllerinteraktionen*).

Use-Case	Funktionale Anforderung
Headset bewegen	Perspektive wird ausgerichtet
Controller bewegen	Virtuelle Hände werden relativ zum Controller bewegt
Controller bewegen	Virtuelle Hände werden durch Controller geschlossen
Controller bewegen	Hände werden durch Controller geöffnet
Im Raum bewegen	Spieler an beliebigen freien Ort im Raum teleportieren

*Tabelle 100: Use-Cases der Headset- und Controllerinteraktionen*

## 7.3 Grabbable

Ein Grabbable bezeichnet ein Objekt, welches vom Spieler in die Hand genommen werden kann. Das Einzige was als Entwickler benötigt wird, um ein Objekt grabbable zu machen, ist ein Collider. Grabbable selbst erweitert die OVRGrabbable Klasse, welche von der Oculus Integration SDK bereitgestellt wird.

### 7.3.1 Events

Die OVRGrabbable Klasse ist so weit eingeschränkt, dass der Entwickler keine Möglichkeit hat, einen Listener auf OnGrab und OnRelease zu registrieren. Die Klasse Grabbable wurde darum für das Serious Game um UnityEvents für diese beiden Events erweitert.

Event	Beschreibung
OnGrab	Sobald ein Objekt in die Hand genommen wird.
OnRelease	Sobald ein Objekt von der Hand losgelassen wird.

Tabelle 101: Events eines Grabbable

## 7.4 Snapzone

Eine Snapzone kann ein bestimmtes Objekt (Technisch: als Snappable bezeichnet) entgegennehmen und festhalten. Sobald das Snappable in einen bestimmten Bereich eindringt (und nicht gleichzeitig noch vom Spieler gehalten wird), wird es innerhalb dieser Snapzone fix platziert. Dieser Prozess wird normalerweise als Snapping bezeichnet. Jedes Snappable ist auch ein Grabbable aber nicht umgekehrt.

Als visuelle Unterstützung wird dem Spieler eine Vorschau auf einer Snapzone angezeigt. Die Vorschau soll dem Spieler anzeigen, wo ein Snappable hin gesnapppt würde, sofern es losgelassen wird.

Anwendungsfälle in dem Serious Game sind beispielsweise, sobald ein Kabel mit einem Gatter verbunden werden soll oder sobald ein Gatter auf dem Board platziert werden soll.

In Abbildung 20 sind die grün umrandeten Snapzones mit Pfeilen gekennzeichnet.

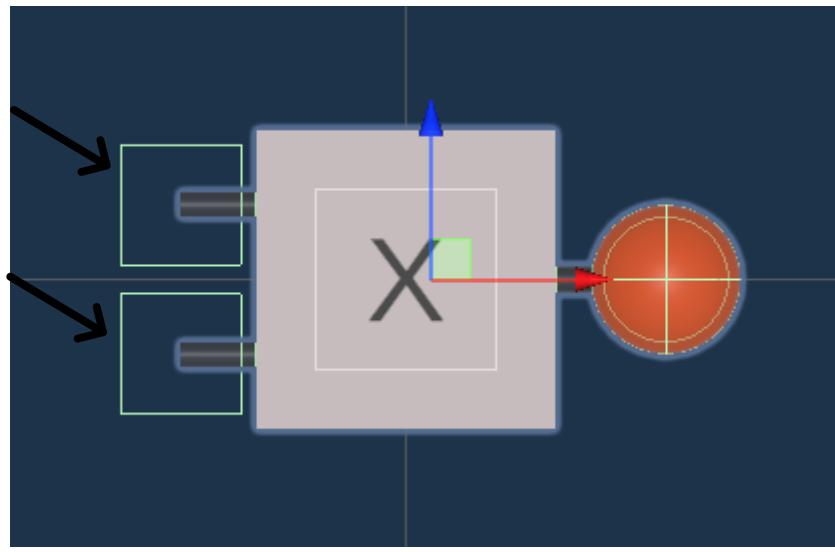


Abbildung 20: Vogelperspektive eines einfachen Gatters mit eingefärbten Snap-Zonen

Ein anderes Beispiel für eine Snapzone befindet sich, wie in Abbildung 21 erkennbar, auf dem Socket für das Gatter. Speziell daran ist, dass die Snapzone explizit in die Höhe gezogen wurde, damit die Vorschau besser sichtbar ist.

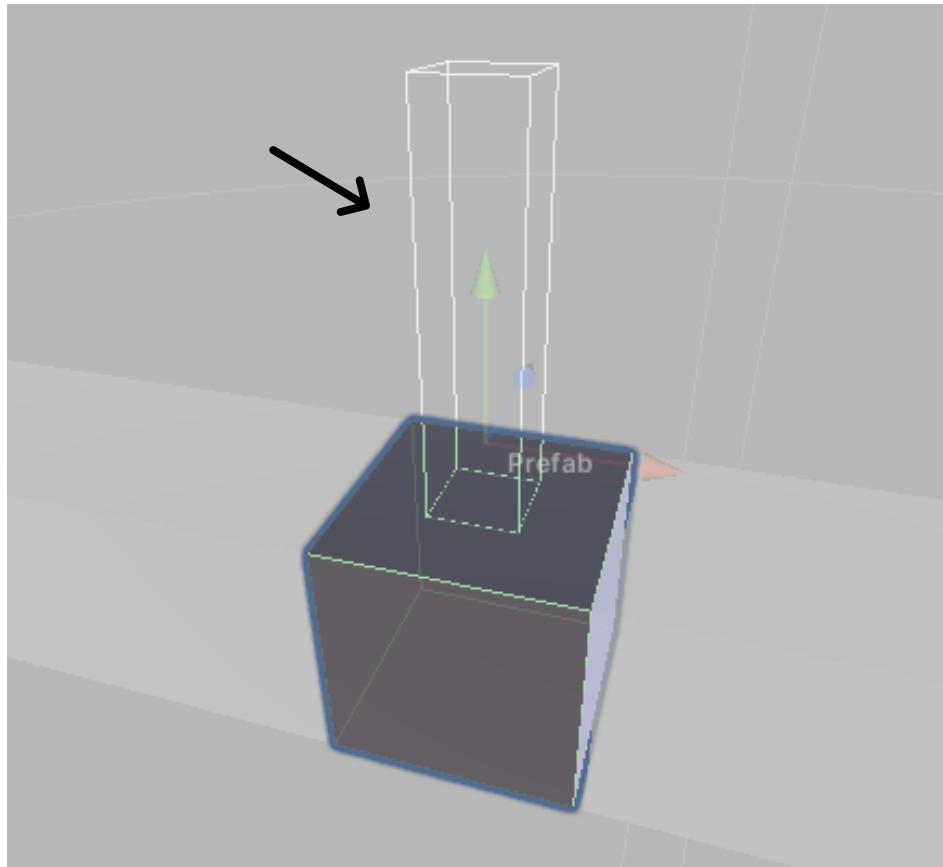


Abbildung 21: Seitenansicht eines Socket für Gatter

Damit eine Snapzone sinnvoll funktioniert, werden die kompatiblen Snappables eingeschränkt. Das hat zur Folge, dass nur bestimmte Snappables auf eine Snapzone gesnapppt werden können.

Snapzone	Beschreibung
GatterSnapZone	Akzeptiert nur einzelne Objekte mit einer Komponente des Typen Gatter.
CableOutputSnapZone	Akzeptiert nur einzelne Objekte mit einer Komponente des Typen CableOutputConnector.

Tabelle 102: Arten von SnapZonen

#### 7.4.1 Events

Snapzones (und Snappables) bieten dem Entwickler die folgenden Events:

Event	Beschreibung
OnSnap	Sobald ein Snappable auf eine Snapzone gesnappt wird.
OnUnsnap	Sobald ein Snappable von einer Snapzone entfernt wird.

Tabelle 103: Events einer Snapzone

#### 7.4.2 Klassendiagramm der Snapzone

Die Struktur der Klassen der Snapzone ist in dem folgenden Diagramm ersichtlich:

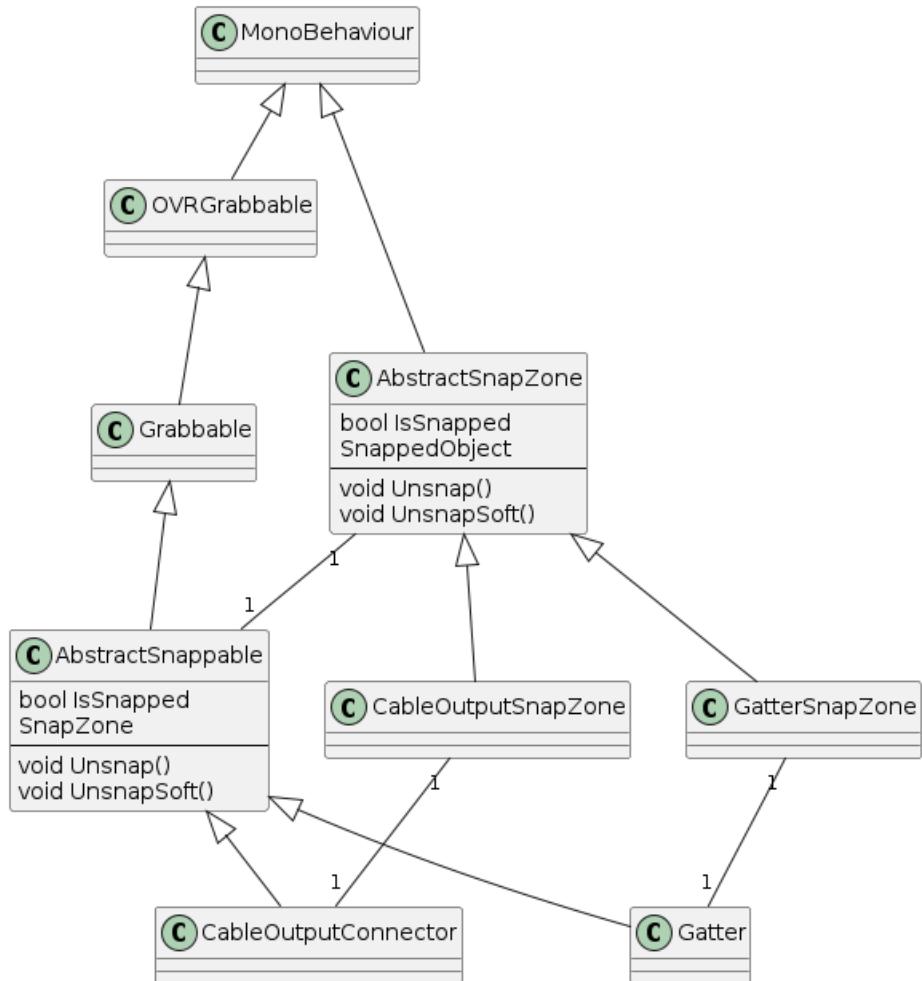


Abbildung 22: Klassendiagramm der Snapzone

### 7.4.3 Use-Cases

Die Snapzone wurde basierend auf den folgenden Use-Cases und funktionalen Anforderungen entwickelt (*Tabelle 104: Use-Cases der Snap-Zone*).

Use-Case	Funktionale Anforderung
Gatter platzieren	Gatter wird auf freiem Socket platziert
Gatter entfernen	Gatter wird vom Socket entfernt
Kabel verbinden	Kabel wird auf freiem Kabel-Eingang platziert
Kabel entfernen	Verbundenes Kabel wird entfernt

*Tabelle 104: Use-Cases der Snap-Zone*

## 7.5 Gatter

Das Gatter ist ein Snappable welches zusätzlich zum Snapping diverse weitere Interaktionen ermöglicht. Eine davon ist der Stromfluss, welcher im *Abschnitt 7.8* beschrieben wird.

### 7.5.1 Ausgänge

Die elementaren Gatter wie NAND, NOT, AND, etc. besitzen jeweils genau einen Ausgang. Ausnahmen dafür sind die Gatter HA (HALFADDER) und FA (FULLADDER) welche jeweils zwei Ausgänge behandeln müssen. Im Serious Game wird dies so gehandhabt, dass auf einem Gatter Objekt gleichzeitig zwei Gatter Komponenten platziert werden. Dies ist ersichtlich in *Abbildung 23*.

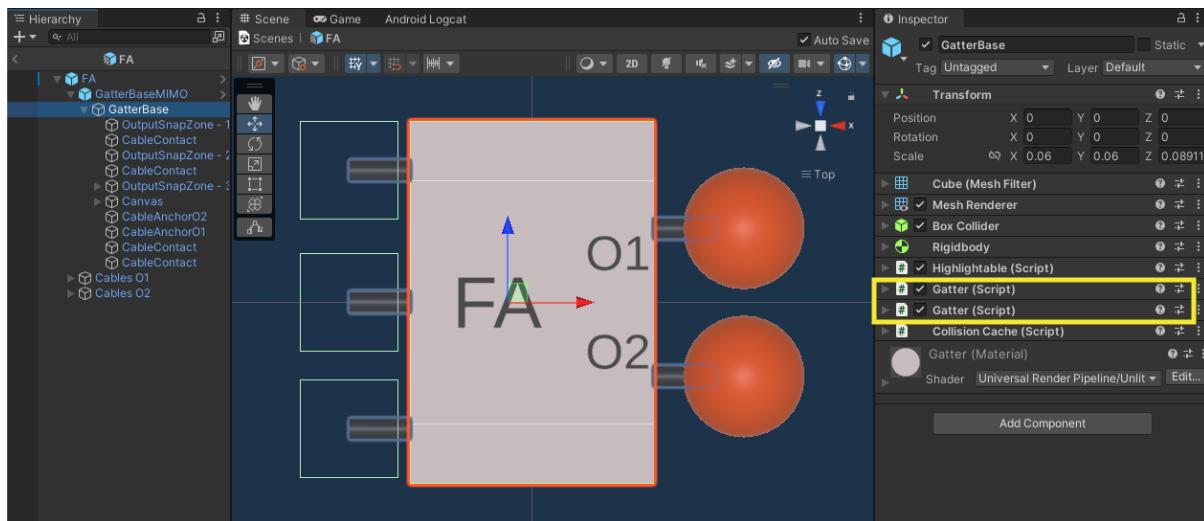


Abbildung 23: Gatter mit mehreren Gatter Komponenten

### 7.5.2 Gatter-Logik

Das Gatter wurde so designed, dass die Logik, welche bestimmt ob und wie Strom am Ausgang fliesst, austausch- und erweiterbar ist. Für diese Situation bietet sich das Strategy-Pattern<sup>8</sup> an. Im folgenden Diagramm sind alle Implementationen der abstrakten GatterLogik aufgeführt.

In *Abbildung 24* erkennt man die Zuweisung der Gatterlogik im Unity-Editor.

Jede Gatter Komponente kann genau eine Gatter-Logik Instanz halten. Dies bedeutet, dass jede Gatter Komponente genau einen Gatter Ausgang behandeln kann.

<sup>8</sup> Mehr zum Strategy-Pattern: [https://www.tutorialspoint.com/design\\_pattern/strategy\\_pattern.htm](https://www.tutorialspoint.com/design_pattern/strategy_pattern.htm)

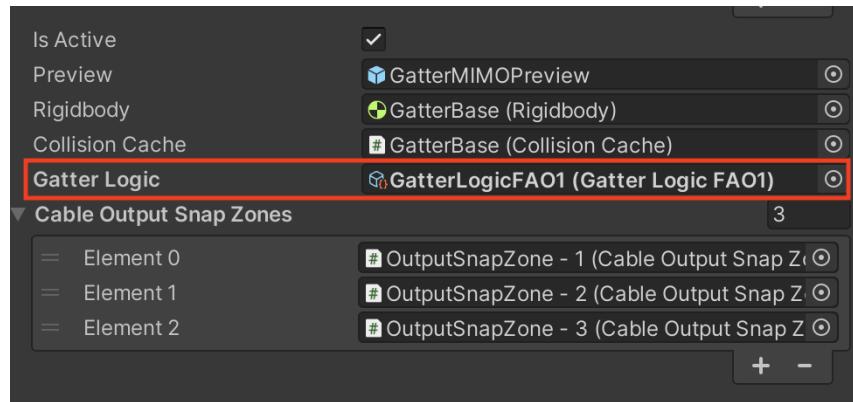


Abbildung 24: Zuweisung einer Gatter-Logik (Volladdierer Output 1) zu einer Gatter-Komponenten

### 7.5.2.1 Klassendiagramm zur Gatterlogik

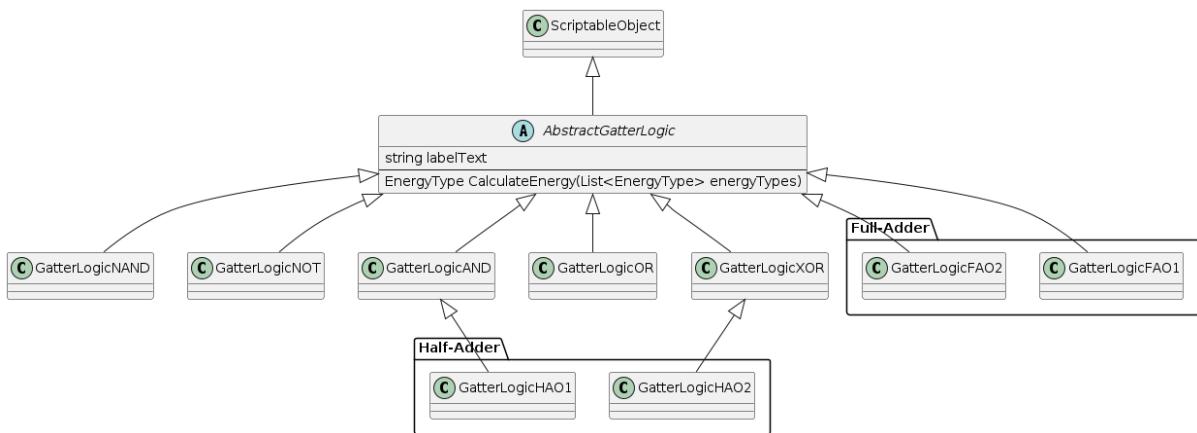


Abbildung 25: Klassendiagramm der Gatter-Logiken

### 7.5.3 Use-Cases

Das Gatter wurde basierend auf den folgenden Use-Cases und funktionalen Anforderungen entwickelt (*Tabelle 105: Use-Cases des Gatters*).

Use-Case	Funktionale Anforderung
Gatter halten	Gatter wird gehalten
Gatter loslassen	Gatter wird losgelassen
Gatter löschen	Gatter wird gelöscht

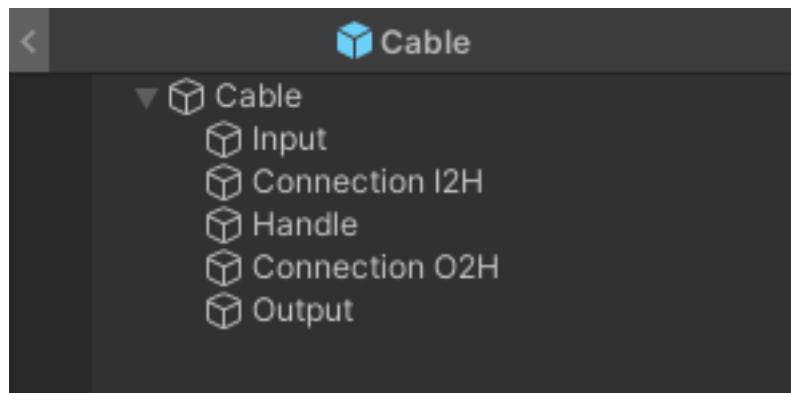
Tabelle 105: Use-Cases des Gatters

## 7.6 Kabel

Das Kabel besteht aus insgesamt fünf Objekten. Input, Output und Handle ermöglichen es dem Spieler das Kabel zu greifen und zu ziehen. Die beiden Verbindungsstücke dazwischen (genannt Connection) werden ausgerichtet und in die Länge gezogen, sodass es wie ein Kabel aussieht. Aus der untenstehenden *Tabelle 106* und *Abbildung 26* kann man den Aufbau des Kabels entnehmen.

Objekt	Beschreibung
Input	Anfang des Kabels. Dieses Objekt repräsentiert einen Anker für das Verbindungsstück. Der Spieler kann nicht damit interagieren.
Connection I2H	Das erste Verbindungsstück zwischen dem Input und dem Handle.
Handle	Ein Handle, um das Kabel zu entfernen, sobald dieses platziert wurde. Der Spieler kann nicht damit interagieren, solange das Kabel nicht platziert ist.
Connection O2H	Das zweite Verbindungsstück zwischen dem Handle und dem Output.
Output	Der Output, um das Kabel zu platzieren. Der Spieler kann nicht damit interagieren, sobald das Kabel platziert wurde. Der Output ist ein Snappable (CableOutputConnector), welches sich mit einer CableSnapZone verbinden kann.

*Tabelle 106: Beschreibung der Objekte innerhalb des Kabels*



*Abbildung 26: Hierarchie des Kabel Prefab*

### 7.6.1 Manuelles entfernen

Ein platziertes Kabel kann entfernt werden, indem der Handle über die konfigurierbare Mindestdistanz weg von der ursprünglichen Position gezogen und schlussendlich losgelassen wird. Dieser Vorgang erkennt man in *Abbildung 27*.

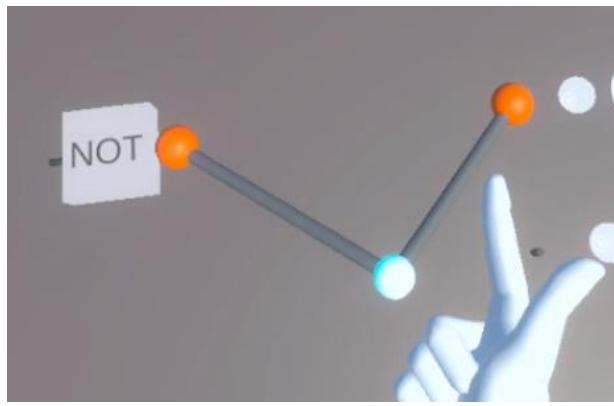


Abbildung 27: Entfernen eines Kabels durch ziehen am Handle

Da es sich beim Output um ein Snappable handelt, kann das Entfernen des Kabels durch ein Unsnap Aufruf bewerkstelligt werden. Sobald ein Kabel entfernt wird, wird der Output an die gleiche Position gesetzt wie der Input. Dadurch wird das Kabel wieder an den Ursprungsort zurückversetzt. Für das blosse Auge sieht es aber so aus, als wäre es entfernt worden.

### 7.6.2 Automatisches entfernen

Falls ein verbundenes Gatter an eine, für den Spieler unerreichbare Stelle geraten soll, wird das Kabel automatisch entfernt. Dies wird durch die Distanz zwischen dem Input und dem Output bestimmt. Sobald diese Distanz eine konfigurierbare Schwelle überschreitet, wird das Kabel automatisch entfernt.

### 7.6.3 Klassendiagramm des Kabels

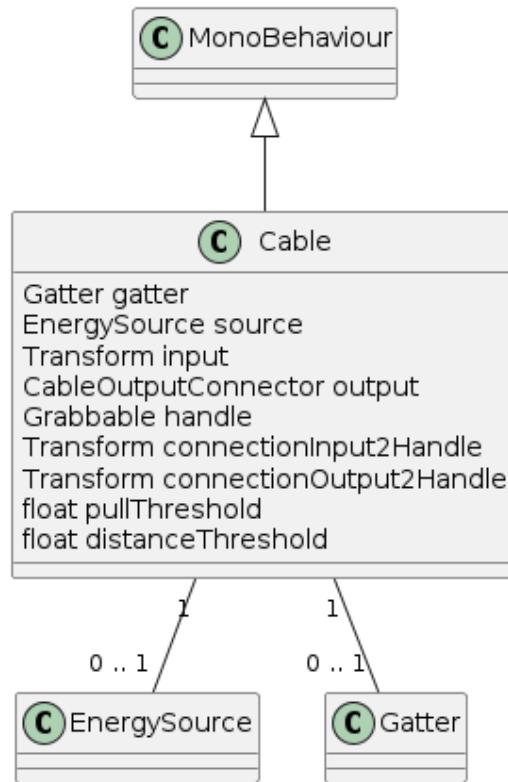


Abbildung 28: Klassendiagramm des Kabels

### 7.6.4 Use-Cases

Das Kabel wurde basierend auf den folgenden Use-Cases und funktionalen Anforderungen entwickelt (*Tabelle 107: Use-Cases des Kabels*).

Use-Case	Funktionale Anforderung
Kabel halten	Kabel wird am Kabel-Handle gehalten
Kabel halten	Kabel wird in der Luft losgelassen

Tabelle 107: Use-Cases des Kabels

## 7.7 Präsentation

Die Präsentation besteht aus zwei Teilen, dem Presenter, welcher eine Präsentation innerhalb einer Szene dem Spieler präsentieren kann und der Präsentation selbst, welche der Entwickler durch ScriptableObjects im Editor zusammenstellen kann.

### 7.7.1 Slide

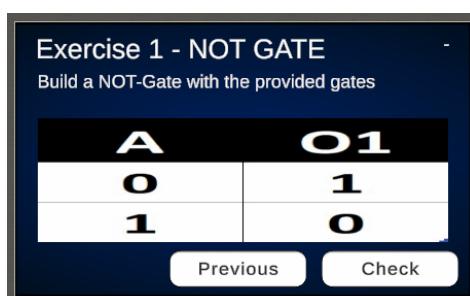
Eine Slide repräsentiert eine Folie innerhalb einer Präsentation. Es wurden spezifisch für das Serious Game und das Tutorial modifizierte Folien erstellt, damit diese mit der Navigation interagieren können. In *Abbildung 31* kann man die Konfiguration einer Folie im Unity-Editor erkennen.

Modus	Beschreibung
Spiel	Die AbstractExerciseSlide ermöglicht es dem Entwickler eine Aufgabe (Technisch: Exercise) zu hinterlegen. Sofern die aktuelle Aufgabe auf dem Board nicht gelöst wurde, wird die Navigation zur nächsten Folie unterdrückt.
Tutorial	Die AbstractTutorialSlide ermöglicht dem Entwickler automatisch zur nächsten Folie zu navigieren, sobald ein bestimmter Event eingetreten ist.  Beispiele für Tutorial Events wären «Ein Gatter in die Hand nehmen», «Ein platziertes Kabel entfernen».

*Tabelle 108: Spezifische Slides für Spielmodus*

Der Entwickler hat die Möglichkeit pro Folie die Labels der Navigationsbuttons zu definieren. Dies ist notwendig, da je nach Kontext der Folie ein fix festgelegtes Label keinen Sinn machen würde.

Im Fall einer Aufgabe, soll der Button nicht mit «Next», sondern «Check» beschrieben werden, da, sofern die Aufgabe nicht korrekt gelöst wurde, nicht auf die nächste Folie navigiert werden kann. In *Abbildung 29* und *Abbildung 30* erkennt man die unterschiedlichen Labels für die Navigationsbuttons.



*Abbildung 29: Überschriebenes Label bei Exercise-Folien*



*Abbildung 30: Default Label für Next-Button*

Um ausserhalb mit einer Folie zu interagieren, bietet eine Folie jeweils zwei UnityEvents für das Erscheinen (Technisch: Show) und Verschwinden (Technisch: Hide) an. Diese werden beispielsweise im Tutorial für das Ein- und Ausblenden der Highlightings auf Objekten verwendet.

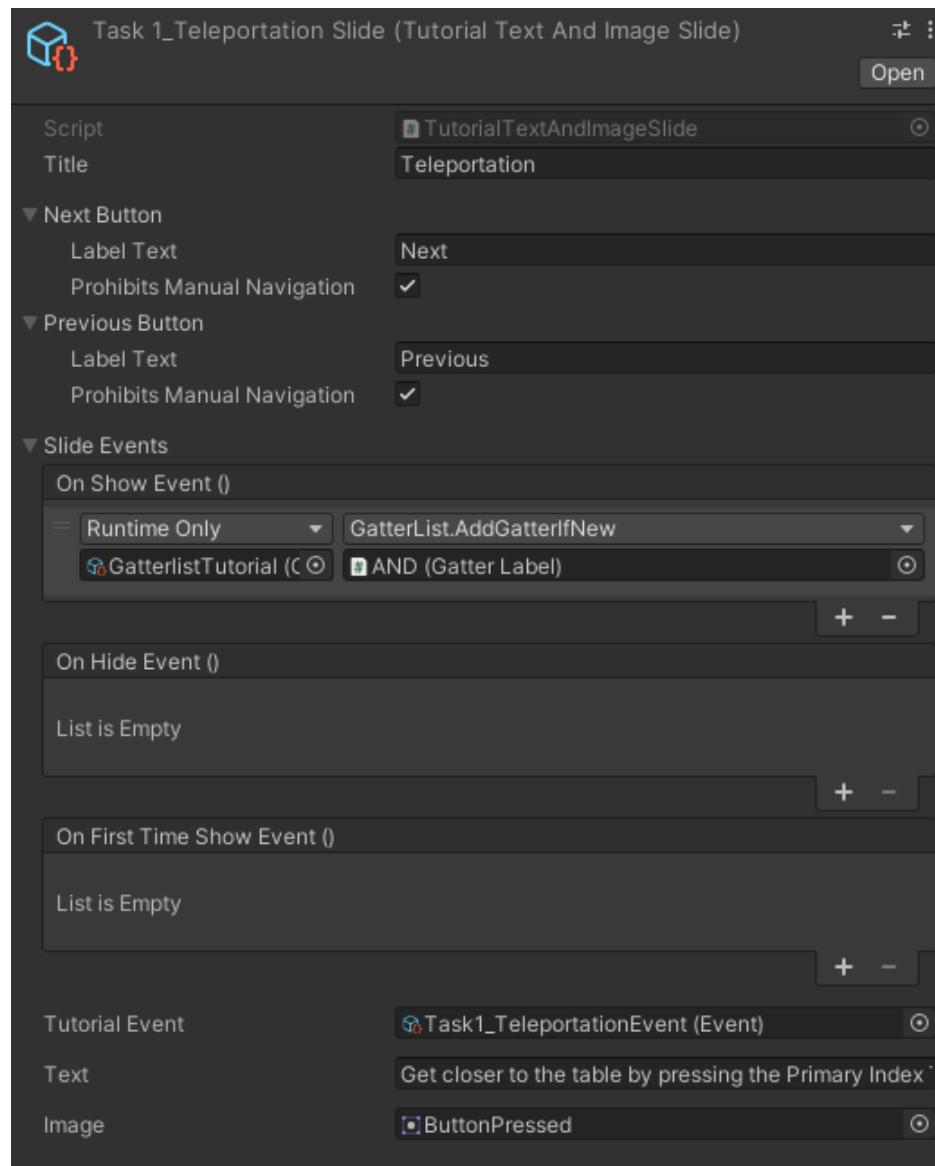


Abbildung 31: Konfigurationsansicht einer Tutorial-Folie

Das folgende Klassendiagramm (*Abbildung 32*) zeigt den Aufbau und die Hierarchie der verschiedenen Arten von Folien.

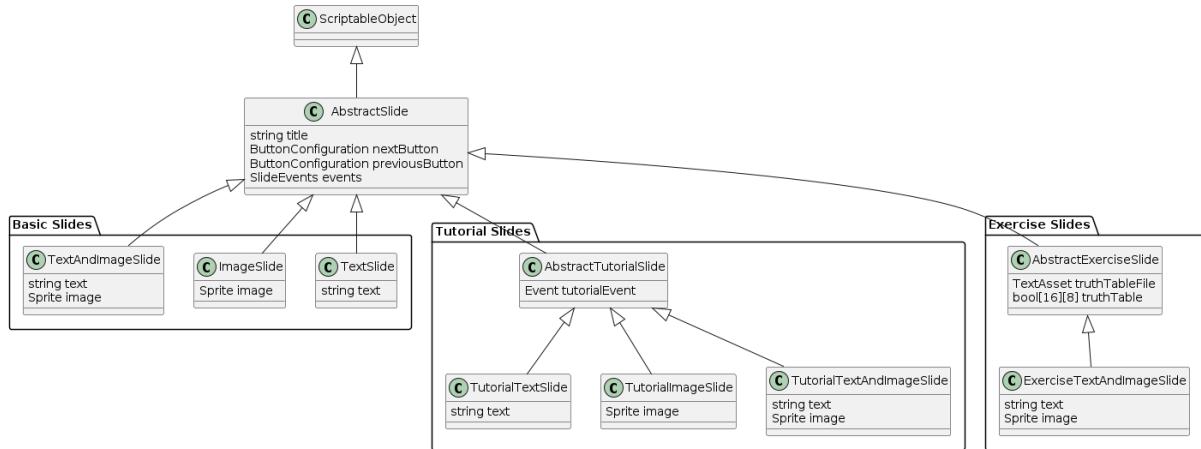


Abbildung 32: Klassendiagramm der Folien

### 7.7.2 Präsentation

Eine Präsentation beinhaltet bestimmte Folien mit einer definierten Reihenfolge. Im Prinzip ermöglicht eine Präsentation eine Gruppierung von Folien. Die Folien können sich simultan in mehreren Präsentationen befinden.

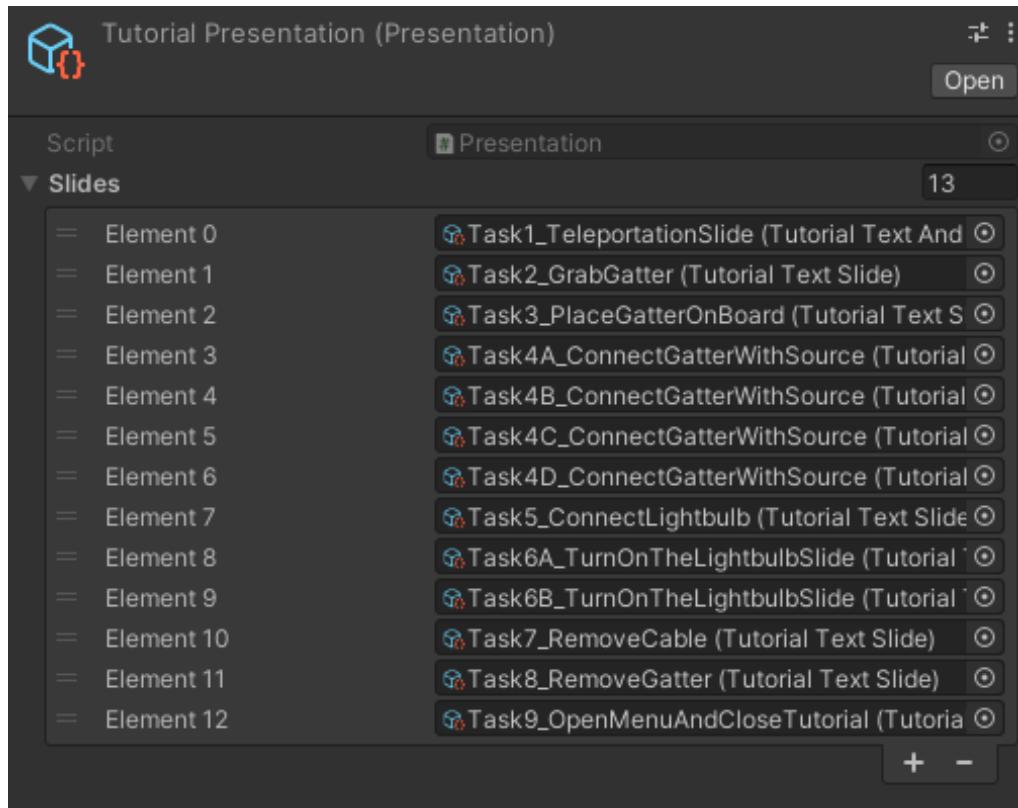


Abbildung 33: Aufbau einer Präsentation

### 7.7.3 Presenter Content

Eine Instanz von PresenterContent bildet einen Wrapper für eine Präsentation. Dieser Wrapper wird benötigt, um den Inhalt eines Presenters zur Laufzeit austauschen zu können. Beispielsweise werden die Hints zu einer Aufgabe in einer Präsentation abgelegt und beim Wechseln einer Aufgabe ausgetauscht.

Im folgenden Klassendiagramm (*Abbildung 34*) wird aufgezeigt, wie durch den PresenterContent auf eine Folie referenziert werden kann.

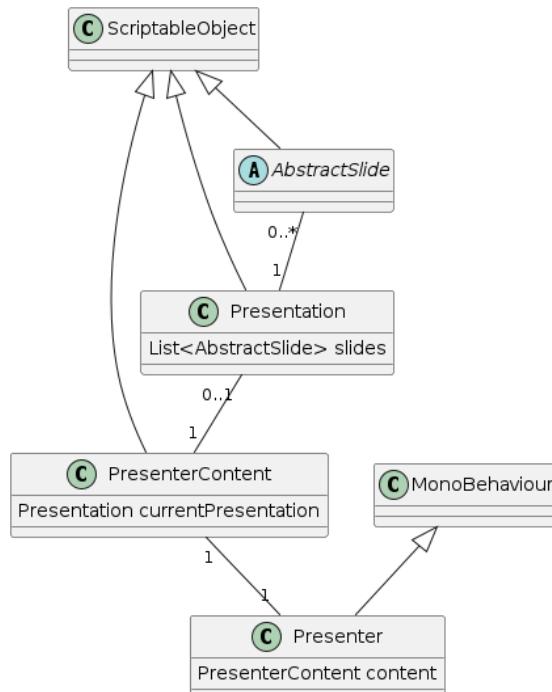


Abbildung 34: Klassendiagramm der Präsentation

Instanz	Inhalt
PresenterContent	Entweder keine oder eine Instanz einer Präsentation.
Presentation	Eine beliebige Anzahl von Folien (AbstractSlides) in einer definierten Reihenfolge.
AbstractSlide	Beliebiger Inhalt. Normalerweise Titel, Text und / oder ein Bild.

Tabelle 109: Inhalt einer Instanz (Präsentationskontext)

In der folgend *Abbildung 35* werden die Instanzen der Hint-Präsentationen, welche in der PresenterContent Instanz verwendet werden, aufgelistet.

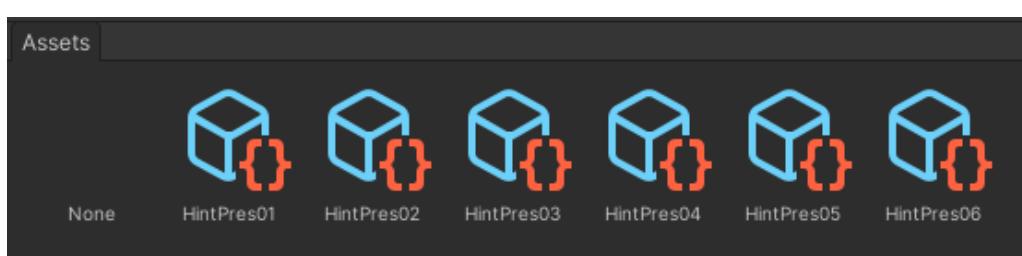


Abbildung 35: Liste der Hint-Präsentationen

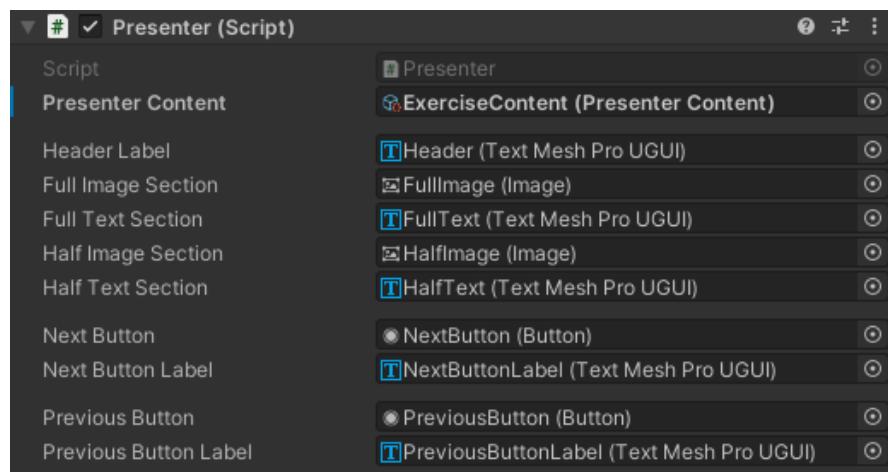
#### 7.7.4 Presenter

Der Presenter wird vom Entwickler mit diversen Text- und Bildelementen (Platzhalter) verknüpft. Er projiziert dann die Inhalte der aktuellen Folie auf eine gewählte Oberfläche, im Falle des Serious Games auf die Wandtafel. In der folgenden *Tabelle 110* werden diese Elemente stichwortartig aufgeführt.

Element	Beschreibung
Header Label	Titel der Folie.
Full Image Section	Bild einer Bild-Folie.
Full Text Section	Text einer Text-Folie.
Half Image Section	Bild einer Allzweck-Folie.
Half Text Section	Text einer Allzweck-Folie.
Next Button Label	Label des «Next»-Button.
Previous Button Label	Label des «Previous»-Button.

*Tabelle 110: Verknüpfte Presenter-Elemente*

In der nachfolgenden *Abbildung 36* ist ein Beispiel eines konfigurierten Presenters innerhalb der Spielszene ersichtlich.



*Abbildung 36: Konfiguration eines Presenters*

### 7.7.5 Use-Cases

Die Präsentation (resp. Das ganze Framework um die Präsentation) wurde basierend auf den folgenden Use-Cases und funktionalen Anforderungen entwickelt (*Tabelle 111: Use-Cases des Präsentations-Framework*).

<b>Use-Case</b>	<b>Funktionale Anforderung</b>
Aufgabe ablesen	Aktuelle Aufgabe wird auf der Wandtafel angezeigt
Aufgabe überprüfen	Check-Button wird angezeigt
Aufgabe überprüfen	Falsch gelöste Aufgabe
Aufgabe überprüfen	Korrekt gelöste Aufgabe
Navigieren	Weiter navigieren
Navigieren	Zurück navigieren
Hinweis / Lösung beziehen	Kosten von Score abziehen
Hinweis / Lösung beziehen	Nächsten Hinweis / Lösung anzeigen
Hinweis / Lösung beziehen	Bezahlte Hinweise / Lösungen nicht erneut bezahlen

*Tabelle 111: Use-Cases des Präsentations-Framework*

## 7.8 Stromfluss

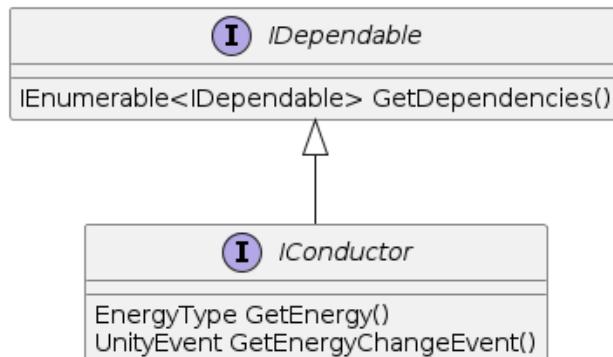
Der Stromfluss bildet eine zentrale Funktion in dem Serious Game. Der Stromfluss muss über diverse Objekte zuverlässig funktionieren, sowie auch für den Spieler nachvollziehbar sein.

Um technisch den Stromwert behandeln zu können, wurde ein Enum mit drei Werten definiert. Dies ist aus physikalischer Sicht nicht korrekt, jedoch für die Behandlung des Stroms praktischer. Diese drei Werte werden in der nachfolgenden *Tabelle 112* aufgelistet.

<b>Wert</b>	<b>Beschreibung</b>
EnergyType.TRUE	Leitet Strom.
EnergyType.FALSE	Leitet keinen Strom.
EnergyType.INVALID	Ungültiger Strom. Beispielsweise wenn ein Gatter nicht auf das Board gesnappt wurde oder nicht alle Eingänge eines Gatters mit einem Kabel verbunden wurden.

*Tabelle 112: Beschreibung der Werte des EnergyType*

Jedes Objekt im Stromfluss implementiert IConductor (Leiter). Der IConductor definiert von wem ein Objekt Strom bezieht und was der ausgehende Strom des Objekts ist.



*Abbildung 37: Klassendiagramm der leitenden Objekte*

Die folgenden Elemente ersichtlich in *Tabelle 113* implementieren IConductor.

<b>Leiter</b>	<b>Abhängigkeiten</b>	<b>Beschreibung</b>
EnergySource	-	Die EnergySource leitet je nachdem ob der dazugehörige Switch umgelegt ist.
Cable	EnergySource Gatter	Das Kabel nimmt den Strom unverändert entweder von der EnergySource oder vom Gatter.
CableOutputConnector	Cable	Der CableOutputConnector nimmt den Strom unverändert vom Kabel.
CableOutputSnapZone	CableOutputConnector	Die CableOutputSnapZone nimmt den Strom unverändert vom CableOutputConnector sofern die SnapZone gesnappt ist. Ansonsten wird EnergyType.INVALID weitergeleitet.
Gatter	CableOutputSnapZone	Das Gatter nimmt den Strom von allen dazugehörigen CableOutputSnapZonen und bestimmt den ausgehenden Strom.

EnergyDestination	CableOutputSnapZone	Die EnergyDestination nimmt den Strom unverändert von der CableOutputSnapZone.
-------------------	---------------------	--

*Tabelle 113: Abhängigkeiten der Leiter*

Damit der Strom nicht in jedem Frame erneut berechnet wird, besitzt jeder Leiter einen Cache, welcher den aktuellen Strom zwischenspeichert. Sobald ein eingehender Event eintritt, wird der Strom automatisch neu berechnet und der Cache wird aktualisiert.

### 7.8.1 Stromfluss des Gatters

Im folgenden Abschnitt wird der Stromfluss des Gatters beschrieben.

#### 7.8.1.1 Eingehende Events

Die folgenden eingehenden Events in *Tabelle 114* führen dazu, dass der Strom auf dem Gatter neu berechnet wird.

Event	Sender
Snap Gatter auf Board	Gatter
Un-Snap Gatter von Board	Gatter
Snap Kabel auf Gatter	CableOutputSnapZone
Un-Snap Kabel von Gatter	CableOutputSnapZone
Stromwechsel	CableOutputSnapZone

*Tabelle 114: Eingehende Events für das Gatter*

### 7.8.1.2 Swimlane-Diagramm

Im folgenden Swimlane-Diagramm wird die Berechnung des Stromflusses im Zusammenhang mit den eingehenden Events ersichtlich.

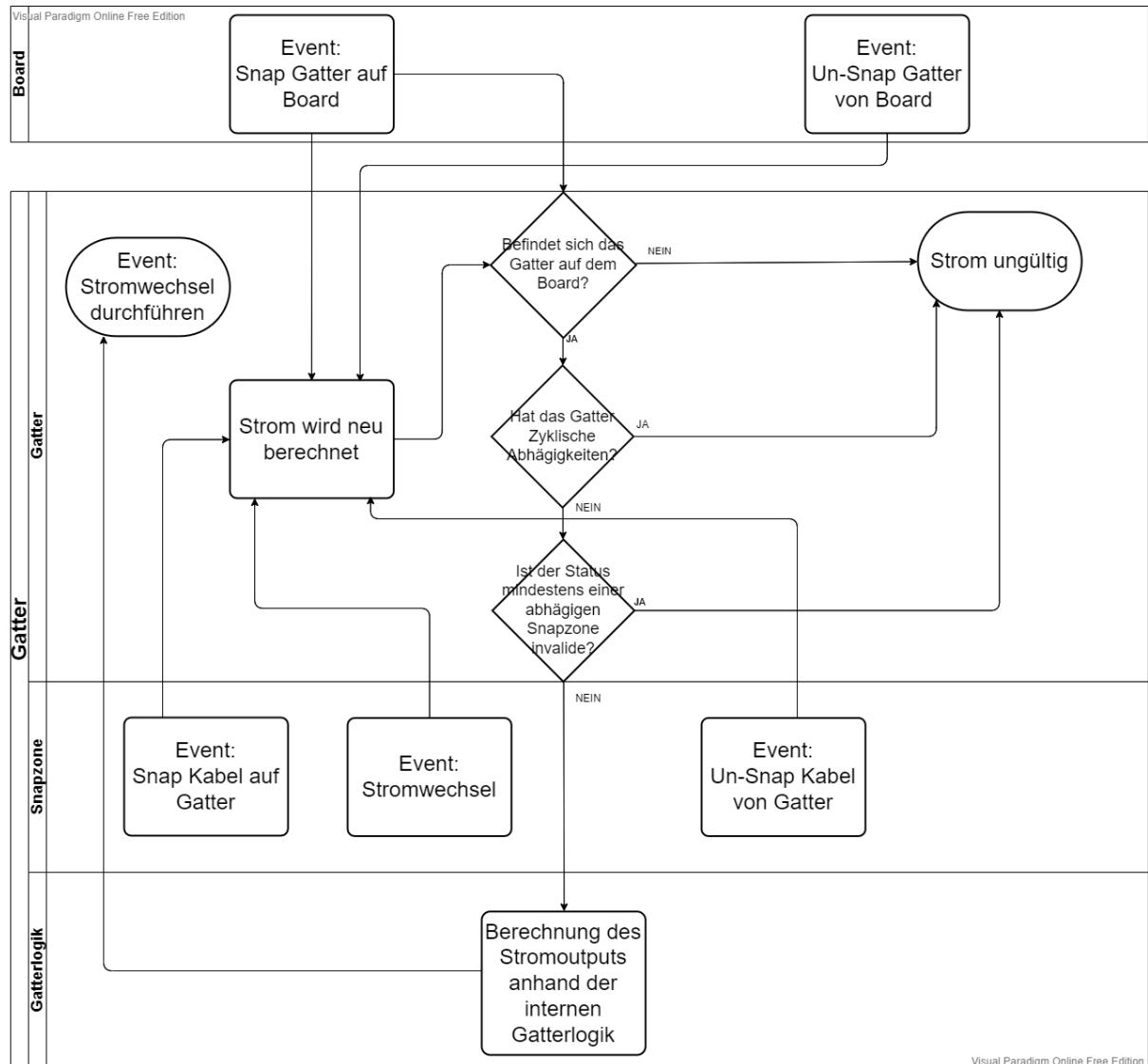


Abbildung 38: Swimlane Diagramm Stromfluss im Gatter

Visual Paradigm Online Free Edition

## 7.8.2 Kabel

Im folgenden Abschnitt wird der Stromfluss des Kabels beschrieben.

### 7.8.2.1 Eingehende Events

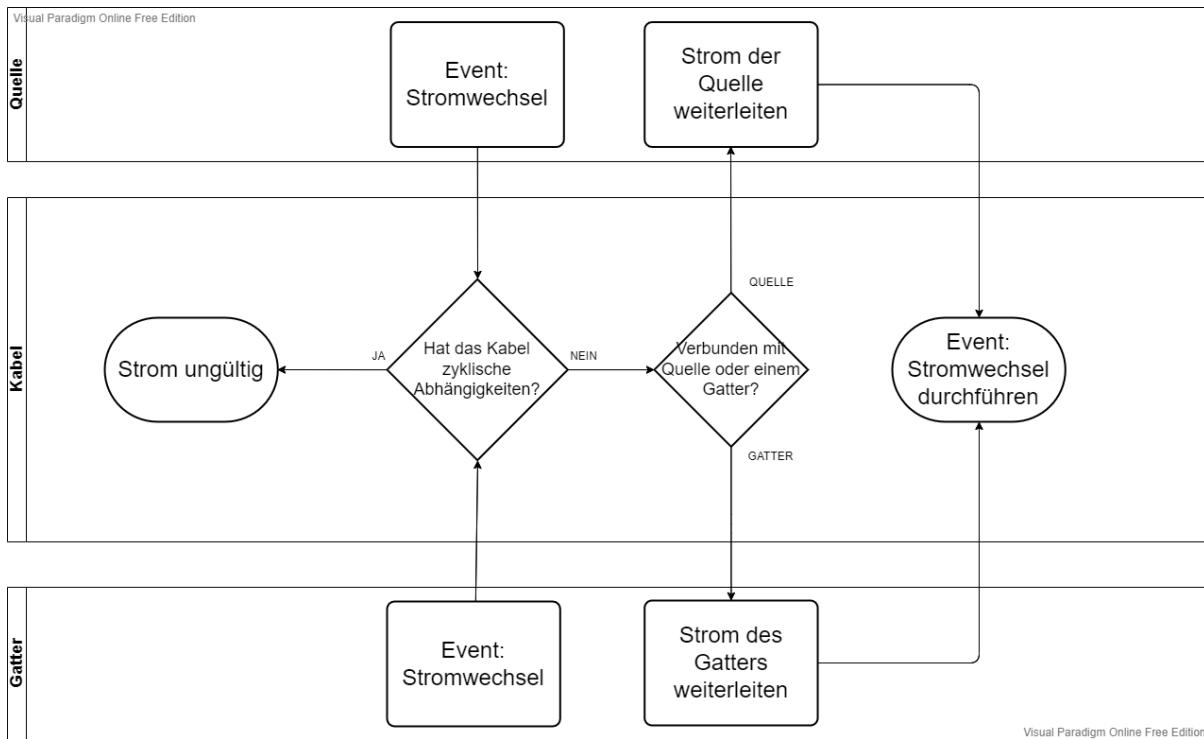
Die folgenden eingehenden Events in *Tabelle 115* führen dazu, dass der Strom auf dem Kabel neu berechnet wird.

Event	Sender
Stromwechsel	Gatter
Stromwechsel	EnergySource

*Tabelle 115: Eingehende Events für das Kabel*

### 7.8.2.2 Swimlane-Diagramm

Im folgenden Swimlane-Diagramm wird die Berechnung des Stromflusses im Zusammenhang mit den eingehenden Events ersichtlich.



*Abbildung 39: Swimlane Diagramm zum Stromfluss im Kabel*

### 7.8.3 EnergySource

Im folgenden Abschnitt wird der Stromfluss der EnergySource beschrieben.

#### 7.8.3.1 Eingehende Events

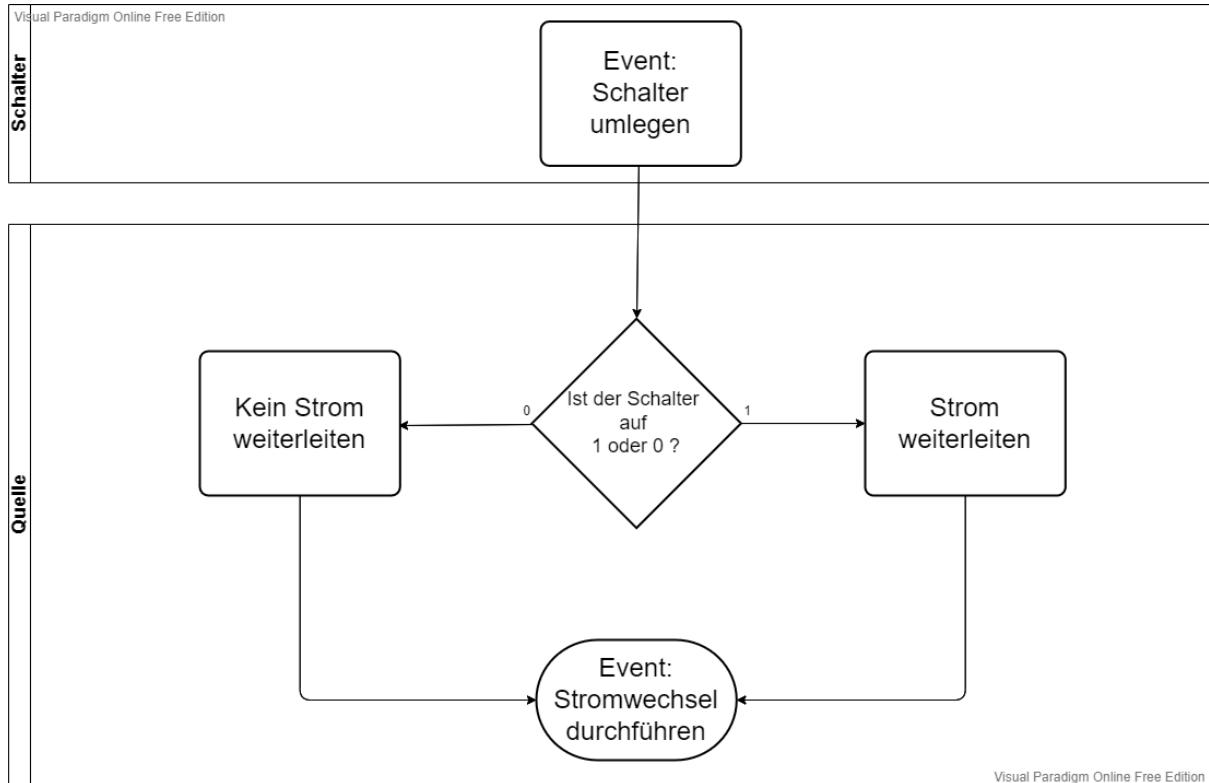
Die folgenden eingehenden Events in *Tabelle 116* führen dazu, dass der Strom auf der EnergySource neu berechnet wird.

Event	Sender
Schalter umlegen	Schalter

*Tabelle 116: Eingehende Events für die EnergySource*

#### 7.8.3.2 Swimlane-Diagramm

Im folgenden Swimlane-Diagramm wird die Berechnung des Stromflusses im Zusammenhang mit den eingehenden Events ersichtlich.



*Abbildung 40: Swimlane Diagramm Stromfluss der Quelle*

### 7.8.4 EnergyDestination

Im folgenden Abschnitt wird der Stromfluss der EnergyDestination beschrieben.

#### 7.8.4.1 Eingehende Events

Die folgenden eingehenden Events in *Tabelle 118* führen dazu, dass der Strom auf der EnergyDestination neu berechnet wird.

Event	Sender
Snap Kabel auf EnergyDestination	CableOutputSnapZone
Un-Snap Kabel von EnergyDestination	CableOutputSnapZone
Stromwechsel	CableOutputSnapZone

Tabelle 117: Eingehende Events der EnergyDestination

#### 7.8.4.2 Swimlane-Diagramm

Im folgenden Swimlane-Diagramm wird die Berechnung des Stromflusses im Zusammenhang mit den eingehenden Events ersichtlich.

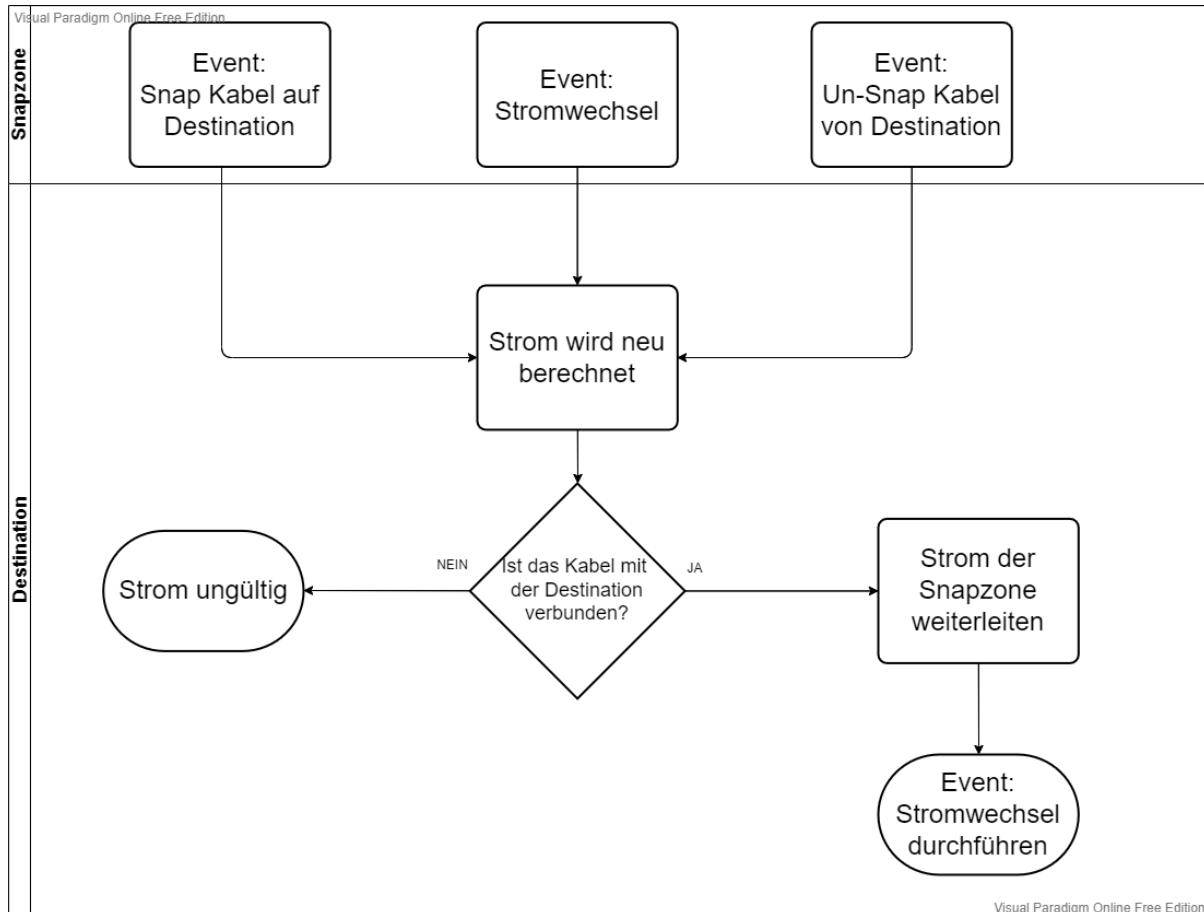


Abbildung 41: Swimlane Diagramm zum Stromfluss der Destination

#### 7.8.5 Ungültige Konstellationen

Eine ungültige Konstellation wird auf dem Board mit grauen Kabeln angezeigt. Dies tritt normalerweise ein, sobald eine Vorbedingung eines Leiters nicht erfüllt wurde.

##### 7.8.5.1 Unplatziertes Gatter

Damit ein Gatter Strom oder keinen Strom leiten kann, muss es auf dem Board platziert sein. Sofern ein Gatter beispielsweise auf dem Tisch liegt und trotzdem mit Strom aus einer EnergySource verbunden ist, wird der EnergyType.Invalid aus dem Gatter weitergegeben und die verbundenen Kabel bleiben grau.

### 7.8.5.2 Zyklus

Eine ungültige Konstellation der Gatter und Kabel tritt auf, sobald ein Zyklus verbunden wird. Beispielsweise wenn der Ausgang eines Gatters indirekt mit dem Eingang des gleichen Gatters verbunden wird. Direkt kann ein Gatter nicht mit sich selbst verbunden werden. In der Abbildung 42 kann man so einen Zyklus erkennen.

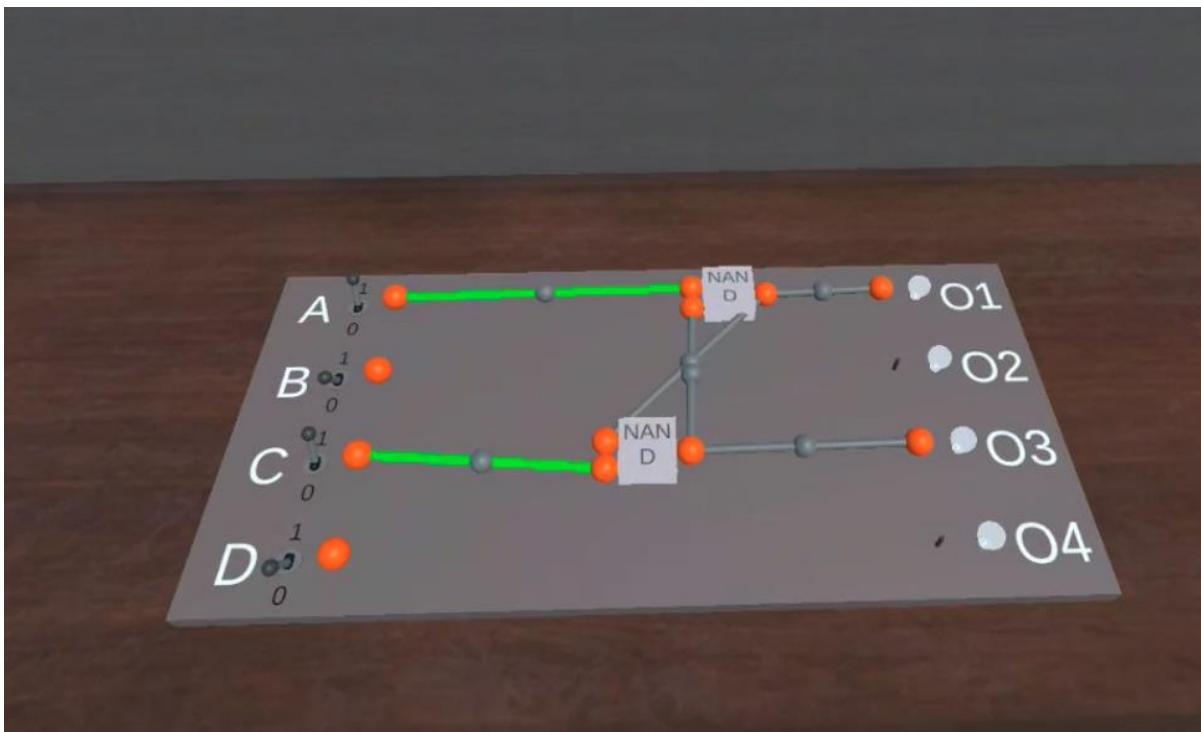


Abbildung 42: Zyklische Konstellation auf Board

Damit ein Leiter bestimmen kann, ob eine zyklische Konstellation gelegt wurde, muss rekursiv durch alle verbundenen Leiter iteriert werden. Das IConductor Interface ermöglicht dies, indem die gelegte Konstellation als Graph abgebildet wird. Damit die aktuell besuchten Leiter identifiziert werden können, wird ein Cache<sup>9</sup> (Technisch: HashSet) dafür initialisiert und der Leiter vor jeder Iteration der Abhängigkeiten darin abgelegt.

### 7.8.6 Use-Cases

Der Stromfluss wurde basierend auf den folgenden Use-Cases und funktionalen Anforderungen entwickelt (*Tabelle 118: Use-Cases des Stromfluss*).

Use-Case	Funktionale Anforderung
Strom einschalten	Schalter wird von 0 auf 1 gekippt
Strom einschalten	Strom auf verbundene Objekte weitergegeben
Strom ausschalten	Schalter wird von 1 auf 0 gekippt
Strom ausschalten	Strom auf verbundene Objekte nicht weitergegeben

Tabelle 118: Use-Cases des Stromfluss

<sup>9</sup> Mehr zu Cache: <https://www.techtarget.com/searchstorage/definition/cache>

## 7.9 Gatter Dispenser

Der Gatter-Dispenser (Kurz Dispenser) spawnt bei Bedarf ein beliebiges Gatter für den Spieler. Der Dispenser führt hierzu eine interne Liste in der zur Laufzeit beliebige Gatter hinzugefügt werden können. Im Hauptspiel wird davon Gebrauch gemacht, indem Anfangs nur das NAND-Gatter in der Liste vorhanden ist und nach erfolgreichem Lösen einer Aufgabe das neue Gatter hinzugefügt wird.

Der Gatter-Dispenser besteht aus insgesamt vier Objekten, welche die Interaktionen mit dem Spieler ermöglichen. Diese Objekte werden in untenstehender Tabelle abgebildet.

Objekt	Beschreibung
Left Button	Wählt das vorherige Gatter in der Gatterliste aus. Falls aktuell das erste Gatter in der Liste ausgewählt ist, wird das letzte Gatter ausgewählt.
Right Button	Wählt das nächste Gatter in der Gatterliste aus. Falls aktuell das letzte Gatter in der Liste ausgewählt ist, wird das erste Gatter ausgewählt.
Dispense Button	Spawn eine Instanz des aktuell ausgewählten Gatters.
Gatter Label	Zeigt das Label des aktuell ausgewählten Gatters an.

Tabelle 119: Objekte innerhalb des Gatter-Dispenser

Damit durch die EDA beliebig Gatter zu einem in der Szene lebenden Dispenser hinzugefügt werden können, wurden die Daten (GatterList) vom Dispenser getrennt. Da die GatterList von ScriptableObject erbt, kann darauf von ausserhalb der Szene zugegriffen werden. (Abbildung 43)

### 7.9.1 Klassendiagramm

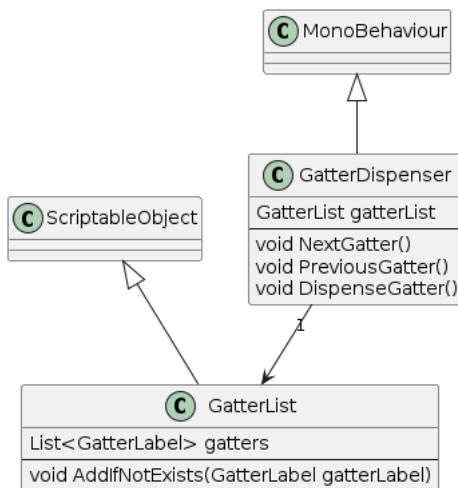


Abbildung 43: Klassendiagramm des Gatter-Dispenser

## 7.9.2 Toggle

Der Toggle (Schalter) kann aus der Sicht des Spielers in zwei unterschiedliche Status versetzt werden. Technisch existieren jedoch insgesamt vier Status. Ein- und Ausgeschaltet sowie zweimal ein Cooldown-Status<sup>10</sup>. Dies wird in der *Abbildung 44* visuell beschrieben.

Ein Statuswechsel wird durch eine Kollision mit einem beliebigen Collider durchgeführt. Dadurch kann beispielsweise auch ein anderes Objekt als eine Hand einen Statuswechsel auslösen.

Nachdem ein Statuswechsel durchgeführt wurde, wird, während einer zuvor festgelegten Cooldown-Zeit, keine weiteren Statuswechsel erlaubt, bis die Zeit abgelaufen ist. Dies hilft zu verhindern, dass zwei Collider kurz hintereinander mit dem Schalter kollidieren und dadurch zwei Statuswechsel auslösen. In diesem Fall würde der Schalter wieder in den initialen Status übergehen, obwohl nur ein einzelner Statuswechsel vom Spieler beabsichtigt war.

In der folgenden *Abbildung 44* wird die Funktion visuell in einem Status-Diagramm dargestellt.

Der Schalter wurde explizit unabhängig von anderen Modulen entwickelt. Dadurch ist der Schalter universell einsetzbar.

### 7.9.2.1 Statusdiagramm

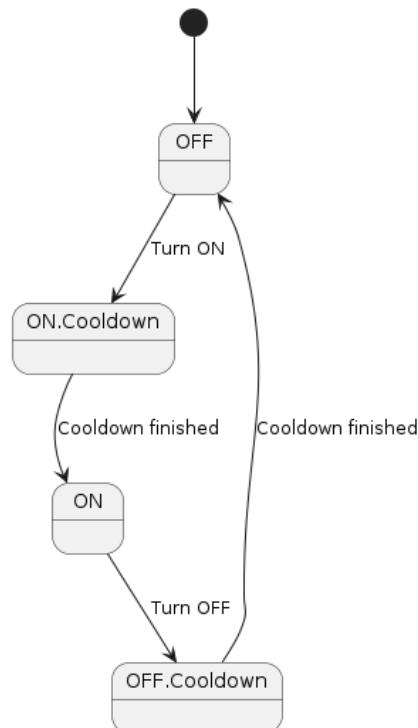


Abbildung 44: Simples Statusdiagramm des Schalters

<sup>10</sup> Ein Cooldown-Status blockiert jegliche statusändernden Aktionen und wird nach einer gewissen Zeit wieder freigeschaltet.

### 7.9.2.2 Klassendiagramm

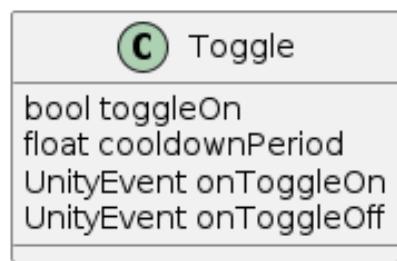


Abbildung 45: Simples Klassendiagramm des Toggle

### 7.9.3 Button

Der Button repräsentiert eine vereinfachte Form des Schalters (ohne den Kontext ein- oder ausgeschaltet zu sein). Auch der Button ist universell einsetzbar. Analog zum Schalter existiert auch beim Button eine konfigurierbare Cooldown-Zeit.

In der folgenden *Abbildung 46* wird die Funktion visuell in einem Status-Diagramm dargestellt.

#### 7.9.3.1 Statusdiagramm

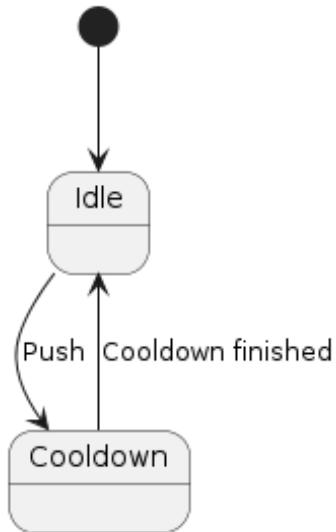


Abbildung 46: Simples Statusdiagramm des Buttons.

#### 7.9.3.2 Klassendiagramm

Da die Oculus-Integration Library bereits eine Klasse mit dem Namen «Button» im gleichen Scope zur Verfügung stellt, wurde entschieden das in diesem Fall einfachheitshalber ein Prefix «BA» verwendet wird. (*Abbildung 47*)

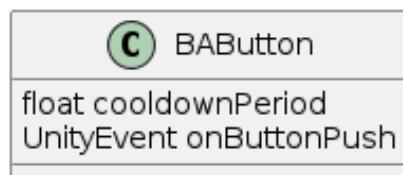


Abbildung 47: Simples Klassendiagramm des Buttons.

## 8 Entwicklungsentscheidungen

In diesem Kapitel werden die grundlegenden Fragen, welche vor der Realisierung beantwortet werden mussten, beschrieben und deren Ergebnis begründet.

### 8.1 Controller vs. Handtracking?

Eine erste Entscheidung, die am Anfang der Entwicklung getroffen werden musste, ist, ob das Spiel mit dem eingebauten Handtracking von Oculus oder mit den mitgelieferten Controllern bedient wird.

Das Serious Game basiert auf mehreren feinen, handlichen Tätigkeiten, wie zum Beispiel kleine Objekte greifen, platzieren und wieder lösen. Das angebotene Handtracking der Oculus Quest hat in einem Demospiel eher grob und schwerfällig gewirkt, darum wurde sich im Sinne der Gamemechanik gegen das Handtracking und für die auf feinere Tätigkeiten ausgelegte Bewegungsmechanik mit den Controllern entschieden.

### 8.2 Spielsprache?

Da das Spiel als Serious Game im Internet auf der Open Source XR Plattform<sup>11</sup> der ZHAW School of Engineering publiziert wird, hat man sich in einem ersten Schritt, für die Spielsprache Englisch entschieden. Im Rahmen der Bachelorarbeit wurde die Entwicklung nicht auf das Bereitstellen von Sprachpaketen ausgelegt, sondern rein auf Englisch konzentriert. Die Aufgaben werden anhand einer Präsentation im Spiel angezeigt, diese Präsentationsfolien kann man ggf. auf andere Sprachen in einem zukünftigen Schritt direkt im Unity Editor anpassen. Auf der unten folgenden *Abbildung 48* sieht man ein Beispiel einer solchen Folie, die man direkt im Unity Editor bearbeiten kann.

---

<sup>11</sup> Auffindbar unter: <https://xr.zhaw.ch/>

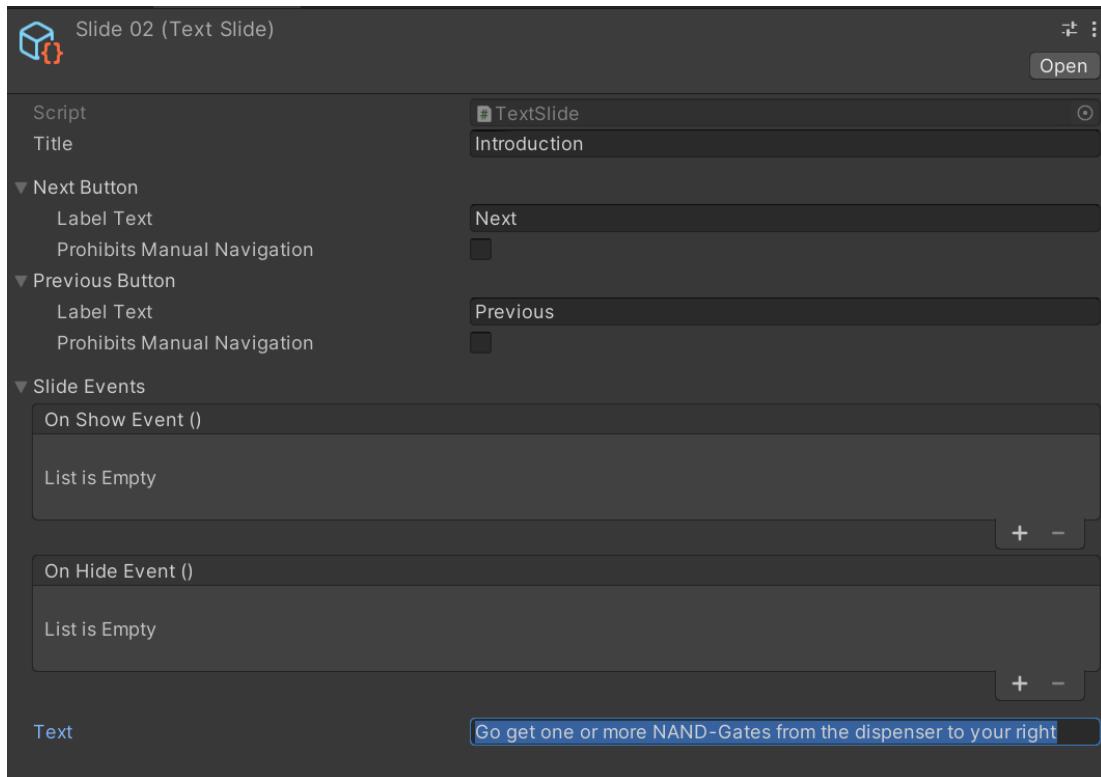


Abbildung 48: Unity Editor zum Erstellen einer Aufgabenstellungs-Folie

### 8.3 XR Interaction Toolkit vs. Oculus Integration SDK?

Während dem Einarbeiten und den ersten Schritten in Unity, musste sich für ein XR Toolkit entschieden werden, um die Controller und Brille richtig anzusteuern. Von Unity selbst gibt es das XR Interaction Toolkit, welches allgemein für alle VR und XR Systeme ausgelegt ist [32]. Für das Serious Game hat man sich, nach Nachforschungen im Internet, explizit für das Oculus Integration SDK [14] entschieden, um die volle Bandbreite der Oculus Quest 2 nutzen zu können.

### 8.4 Single-Player vs. Multi-Player?

Das Serious Game ist in einem ersten Entwurf, im Rahmen der Bachelorarbeit auf Single-Player Modus ausgerichtet. Ein Multiplayer Modus für VR hätte den Umfang der Bachelorarbeit in Zusammenhang mit der komplexen Spiellogik, überschritten. Da es sich um ein Serious Game handelt, wäre die Zusammenarbeit mit anderen Studenten/Schülern wünschenswert, es besteht aber die Möglichkeit das Geschehen über die Oculus Brille auf die Oculus App oder auf der Oculus Plattform<sup>12</sup> live zu streamen, so kann jemand in Echtzeit Anweisungen oder eine Hilfestellung geben.

---

<sup>12</sup> Auffindbar unter: <https://www.oculus.com/casting/>

## 9 Resultate

Im folgenden Kapitel werden die Resultate der Arbeit bestehend aus Spielkomponenten, Szenen, Akustik und Aufgaben genau aufgezeigt und vorgestellt.

### 9.1 Spielkomponenten

Im folgenden Abschnitt werden alle entwickelten Spielkomponenten beschrieben und erklärt.

#### 9.1.1 Kabel



Abbildung 49: Kabel mit unterschiedlichem Stromfluss

Jedes Kabel besteht aus einem Input, Output, Kabelhandle und zwei Verbindungsstücken.

- (1) Der Kabelinput ist fest und lässt sich nicht bewegen. Er verankert das Kabel entweder an der Quelle oder am Gatter.
- (2) Der Kabelhandle wird benutzt, um ein bereits verbundenes Kabel wegzuziehen. In Abbildung 50 erkennbar.
- (3) Der Kabeloutput ist greifbar und kann beliebig weit gezogen werden. Man kann das Kabel mit einem Gatter oder mit einer Glühbirne verbinden.
- (4) Die Kabelverbindungsstücke sind:
  - a. Grau wenn der Stromfluss nicht geschlossen ist, d.h. kein valider Stromkreis gebildet wurde.
  - b. Grün wenn eine 1 geschickt wird.
  - c. Rot wenn eine 0 geschickt wird.

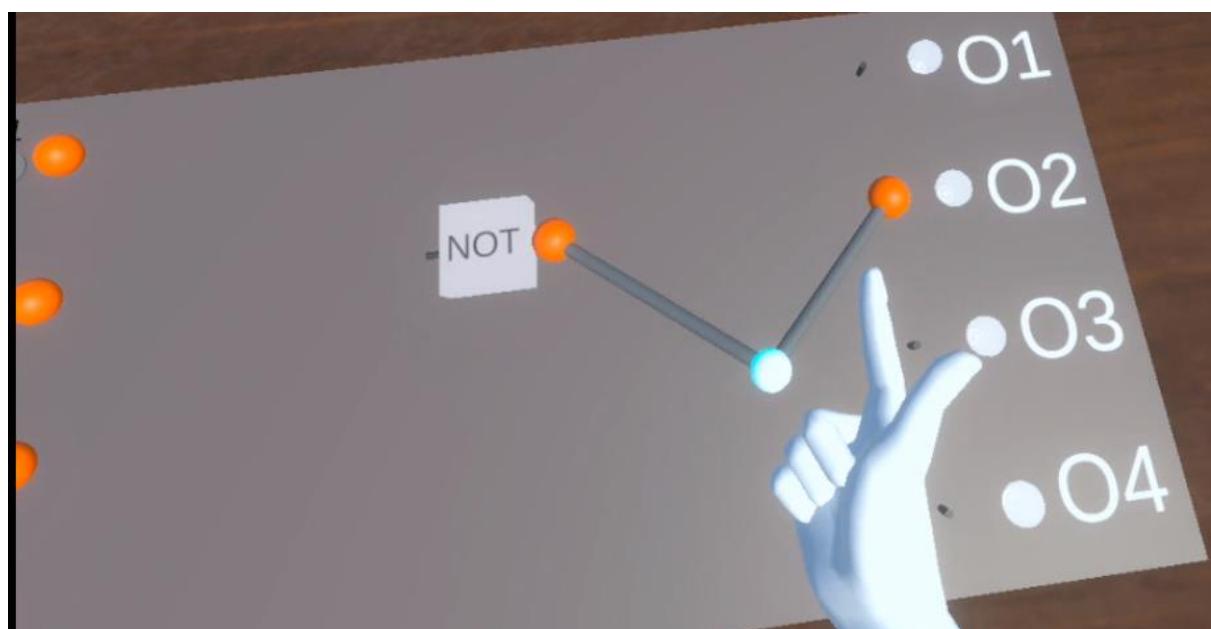


Abbildung 50: Kabel am Kabelhandle wegziehen

### 9.1.2 Gatter

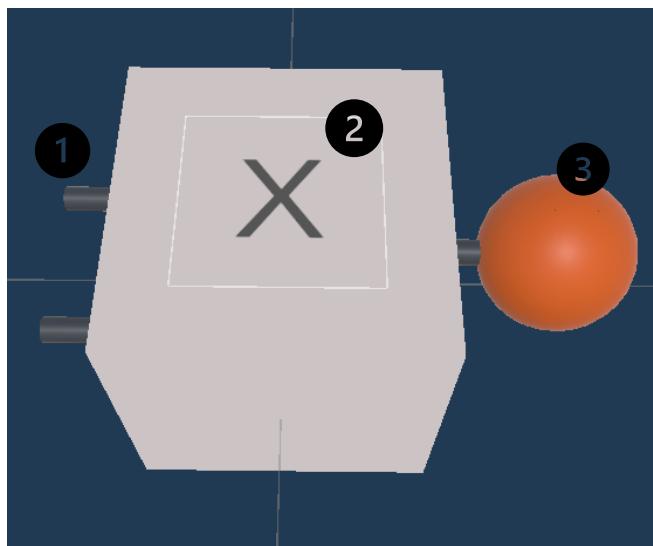


Abbildung 52: Beispiel Gatter mit einem Kabel Ausgang

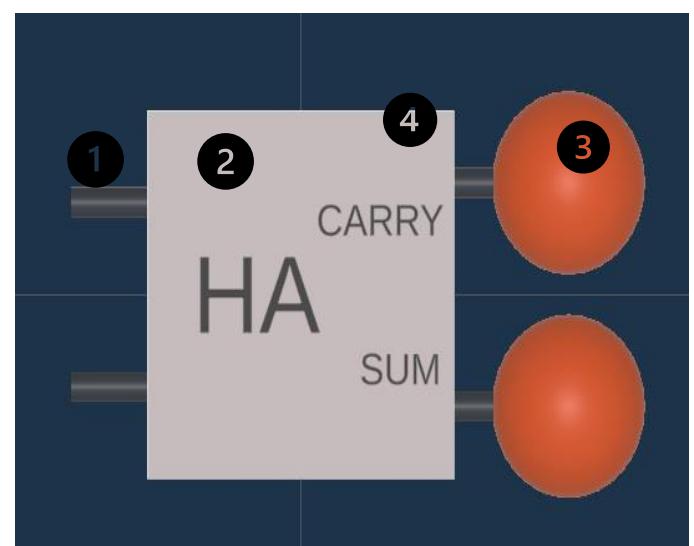


Abbildung 51: Beispiel Gatter mit 2 Kabel Ausgängen

Jedes Gatter besteht aus einem Würfel als Körper, Kabeleingänge, Kabelausgänge und Labels.

- (1) Je nach Gatterlogik, kann ein Gatter ein bis drei Kabeleingänge besitzen.
- (2) Jedes Gatter hat ein Gatterlabel, welches das Gatter beschreibt.
- (3) Je nach Gatterlogik kann ein Gatter ein oder zwei Kabelausgänge besitzen. Aus jedem Kabelausgang können fünf Kabel gezogen werden. (Siehe Abbildung 55)
- (4) Halbaddierer und Volladdierer besitzen zusätzliche Labels (Carry & Sum) für die Ausgänge.

Beim Aufheben eines Gatters leuchtet dieses Blau. Das Kabel kann aus dem Gatter mit der Hand gezogen werden. Dies ist in den folgenden drei Abbildungen ersichtlich.



Abbildung 54: Leuchtendes Gatter

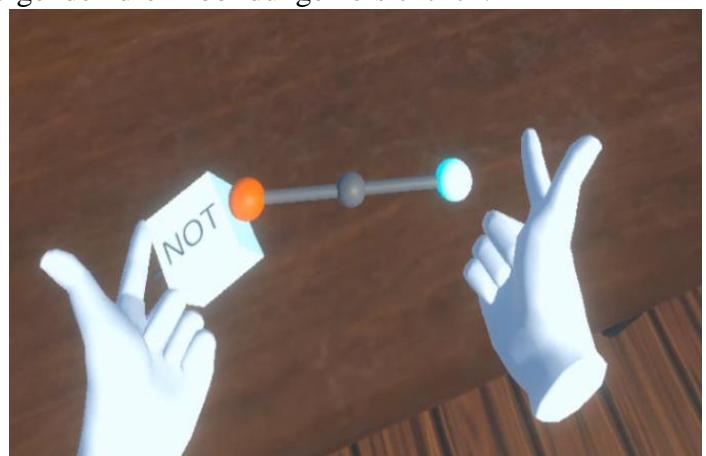


Abbildung 53: Kabel aus dem Gatter ziehen

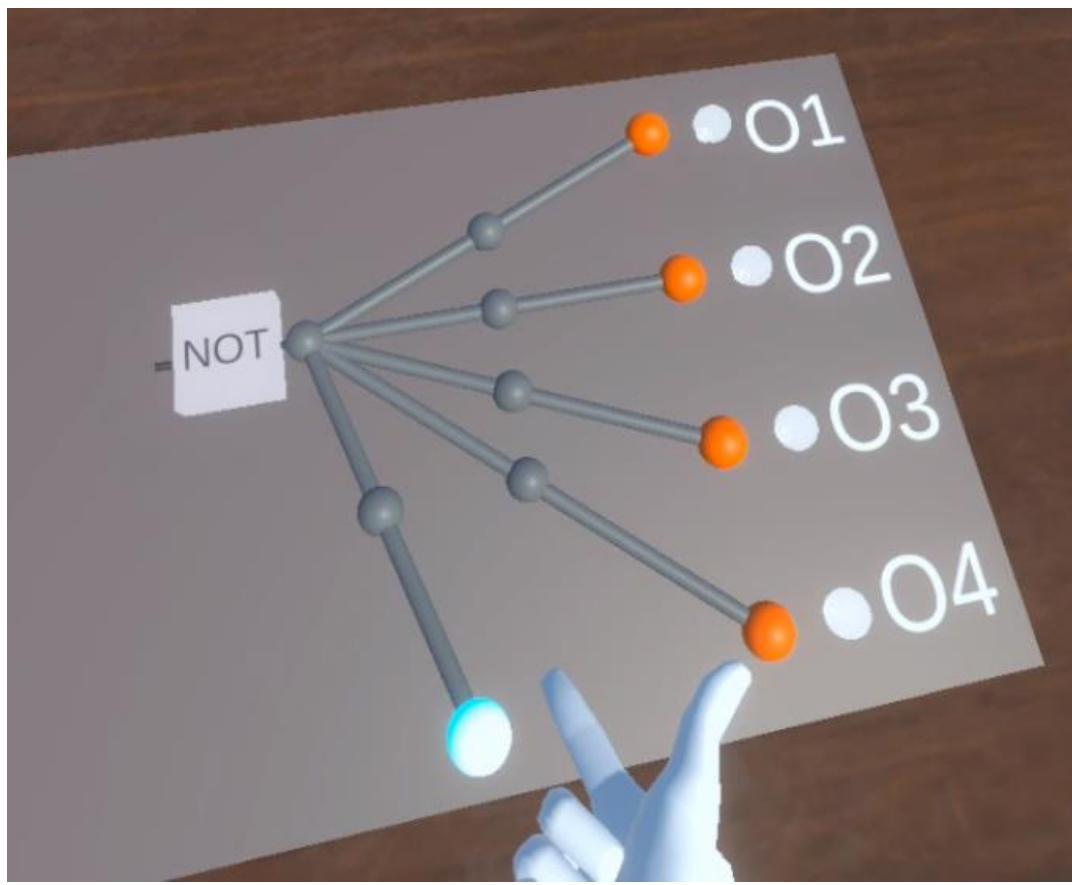


Abbildung 55: 5 Kabel aus dem Gatter ziehen

### 9.1.3 Board

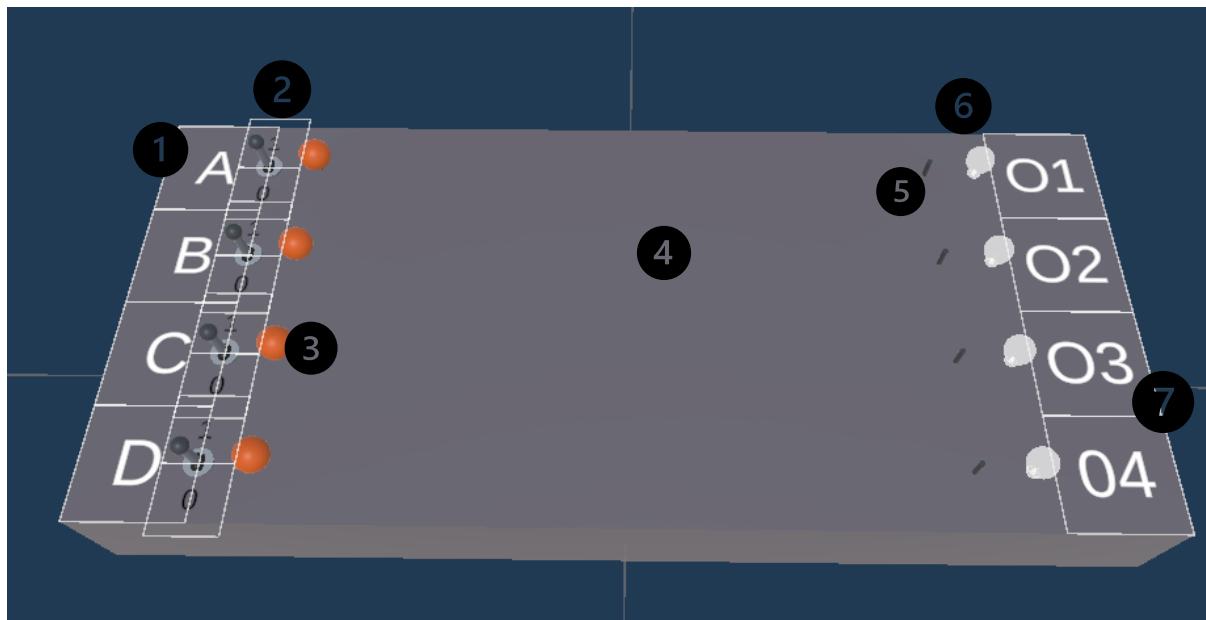


Abbildung 56: Übersicht über das Board

Das Board besteht aus vier Schaltern mit je fünf Kabeln und Label von A-D, vier Glühbirnen mit Kabeleingang und Label von O1 – O4, dazwischen auf dem Board ist Platz für 24 Gatter. Es befindet sich auf dem Spieltisch in der Mitte.

- (1) Labels von A-D für die Schalter.
- (2) Schalter der entweder auf 0 oder 1 gekippt werden kann.
- (3) Zu jedem Schalter gehören fünf Kabel.
- (4) Platz für 24 Gatter. Wenn ein Gatter auf das Board gesteckt wird, sieht man anhand einer Vorschau, wo das Gatter hin platziert wird. (Siehe Abbildung 60)
- (5) Um ein Kabel mit einer Glühbirne zu verbinden, benutzt man den Kabelankerpunkt neben der Glühbirne. Auch hier gibt es eine Vorschau. (Siehe Abbildung 59)
- (6) Das Board besitzt vier Glühbirnen. Die Glühbirnen leuchten nur dann, wenn ein valider Stromkreis angeschlossen ist und eine 1 geschickt wird. Ersichtlich in Abbildung 57 und Abbildung 58.
- (7) Label für die Glühbirnen (Output) O1-O4.

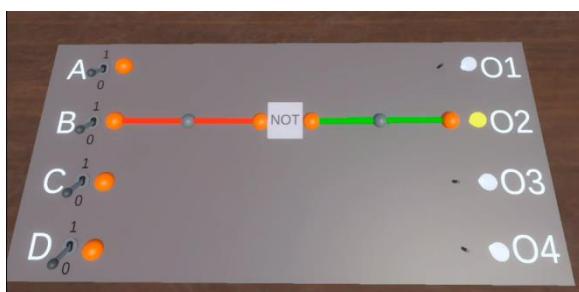


Abbildung 57: Glühbirne leuchtet

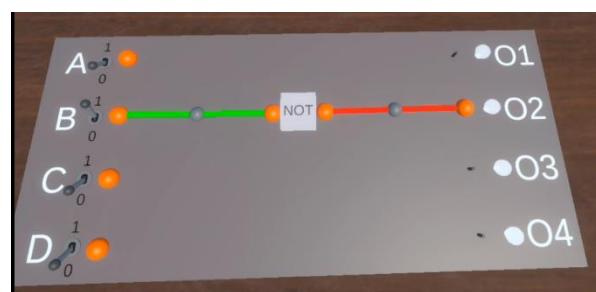


Abbildung 58: Glühbirne leuchtet nicht



Abbildung 59: Vorschau für Kabel

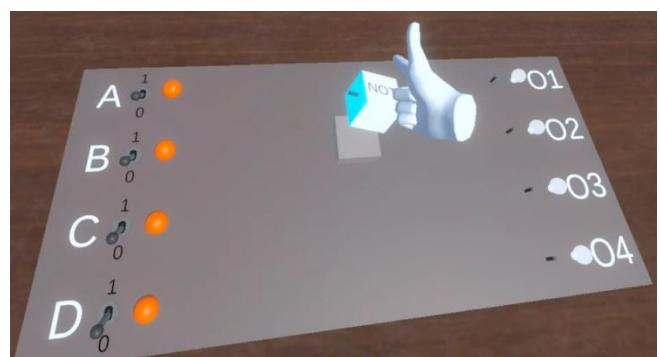


Abbildung 60: Vorschau für Gatter

#### 9.1.4 Automat

Der Automat besteht aus einem Bildschirm, zwei Pfeiltasten, ein Knopf und einem Auswurf. Er befindet sich rechts neben dem Spieltisch.

- (1) Das Anzeigefenster, zeigt den Namen des aktuellen Gatters an.
- (2) Mit den Pfeiltasten kann man zwischen den Gattern hin und her wählen.
- (3) Mit dem «GET»-Knopf kann das ausgewählte Gatter, ausgegeben werden.
- (4) Aus dem Auswurf kommt das Gatter nach Betätigen des «GET»-Knopfs raus. Es fällt auf die Platte unter dem Auswurf und kann von da aus in die Hand genommen werden.

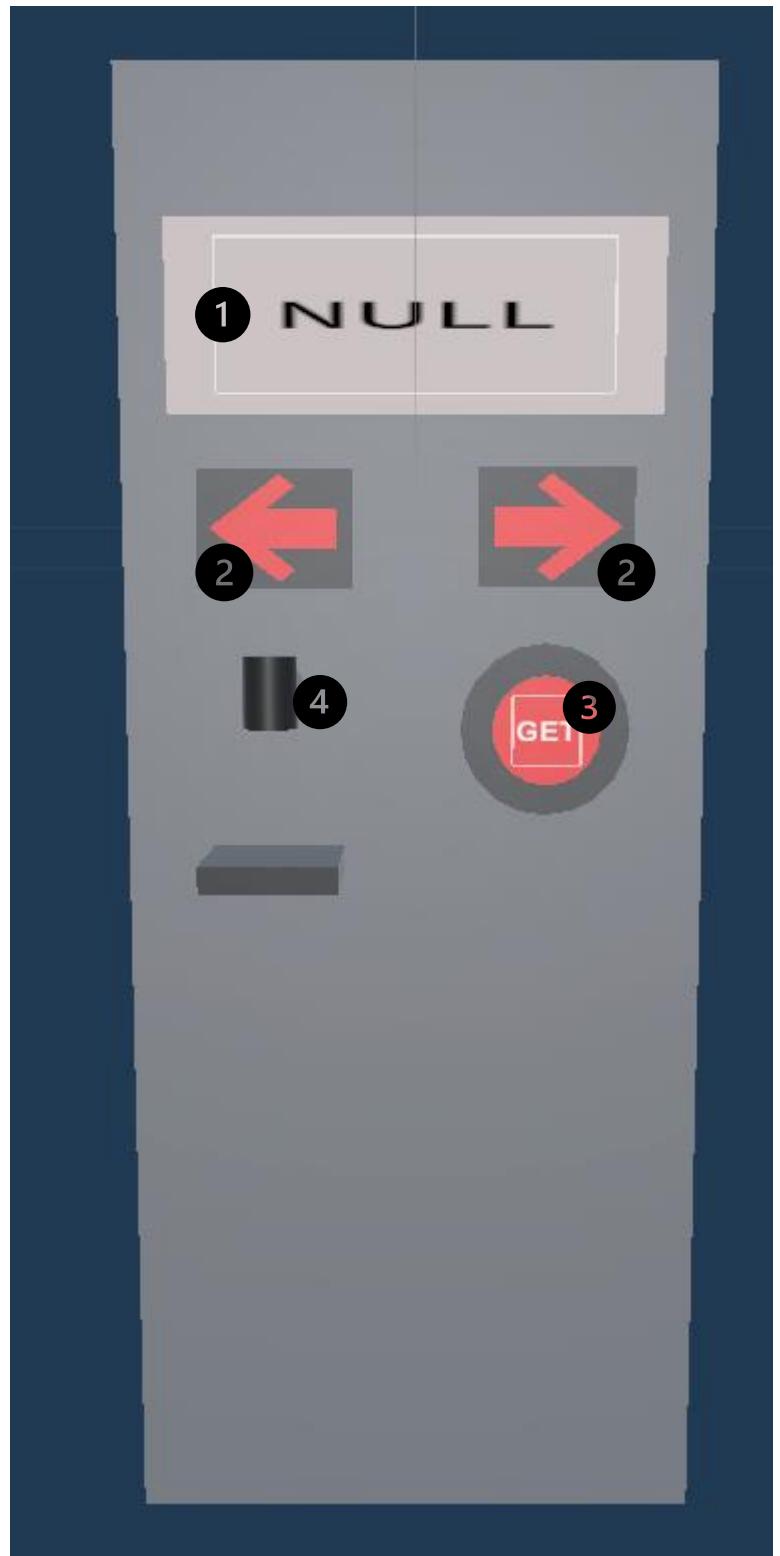


Abbildung 61: Automat

### 9.1.5 Wandtafel



Abbildung 62: Wandtafel Beispiel

Die Wandtafel (*Abbildung 62*) besteht aus verschiedenen Navigations-Knöpfen und einem Überprüf-Knopf. Auf der Wandtafel werden die vorab in Unity angelegten Präsentationsfolien in der vordefinierten Reihenfolge angezeigt. Die Wandtafel wird einmal als Aufgabensteller und einmal als Hinweisanzeiger gebraucht. Die Aufgaben-Wandtafel befindet sich vor dem Tisch und ist während dem Bedienen des Boards allzeit einsehbar. Die Hinweis-Wandtafel befindet sich an der linken Wand neben dem Tisch. Um die Hinweise einzusehen, muss sich der Spieler zur Hinweis-Wandtafel ausrichten.

### 9.1.6 In-Game Menü

Das In-Game Menü besteht aus drei Knöpfen, die über Ray-Tracing mit dem rechten Controller bedient werden können. Der „Continue“ Knopf schliesst das Menü wieder und das Spiel wird normal fortgeführt. Der „Reload“ Knopf lädt die Spielszene neu, d.h. der Spielstand wird zurückgesetzt und das Spiel beginnt von vorne. Der „Exit“ Knopf verlässt das Spiel und der Spieler befindet sich zurück in der Hauptmenü-Szene (Abschnitt 9.2.2).

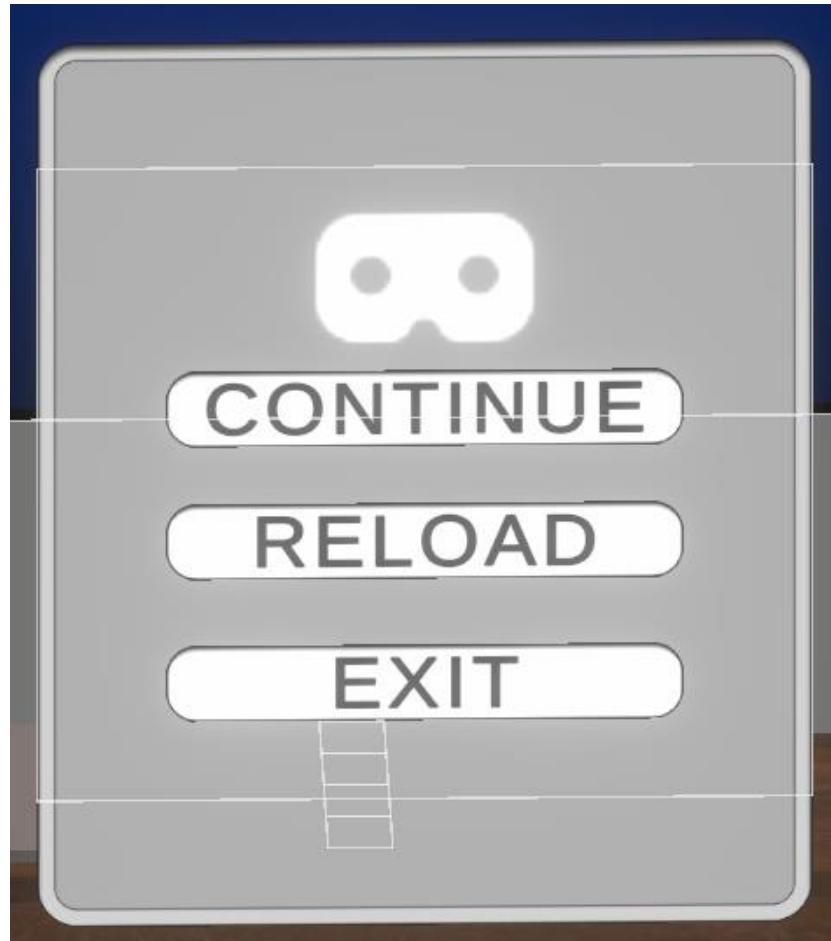


Abbildung 63: In-Game Menü

## 9.2 Szenen

Im Rahmen der Bachelorarbeit wurden 4 Szenen in Unity kreiert. Diese werden im folgenden Abschnitt vorgestellt und beschrieben.

### 9.2.1 Design der Szenen

Alle Szenen bis auf die Hauptmenüszenen sehen identisch aus. Die Spielszene ähnelt einem Tüftler-Büro. Schreibtische, Computer und Technische Apparaturen verleihen der Szene das gewünschte Flair. Die Büro-Objekte<sup>13</sup> dienen einem rein dekorativen Zweck, und haben nichts mit dem Spielablauf zu tun. An den Wänden hängen Poster, welche die einzelnen Wahrheitstabellen zu den Logikgattern anzeigen. Dies soll dem Spieler als freiwillige Hilfestellung dienen. Der Tisch mit dem Spielboard ist dominant in der Mitte platziert, so dass man leicht erkennen kann, wo das Spiel beginnt. Die Deko-Objekte sind dagegen nahe an den Wänden platziert, um den Kontrast zu den interaktiven Gameobjekten zu verstärken. In den untenstehenden Bildern erkennt man Beispieldausschnitte der Szene.



Abbildung 64: Design-Objekte in der Szene



Abbildung 65: Design-Objekte in der Szene

### 9.2.2 Hauptmenü-Szene

Die Hauptmenü-Szene dient der Navigation zwischen den verschiedenen Spielszenen. Beim Spielstart wird diese Szene als Start-Szene initialisiert. Hier kann nun der Spieler zwischen den folgenden Szenen wählen: Spielszene, Tutorial-Szene und Sandbox-Szene.

---

<sup>13</sup> Alle Deko-Objekte kommen aus dem Unity Asset Store:  
<https://assetstore.unity.com/packages/3d/environments/sci-fi/free-sci-fi-office-pack-195067>

Das Menü zur Auswahl der verschiedenen Szenen wird über den VR-Controller mit Ray-Tracing bedient. Ersichtlich in folgender *Abbildung 66*.

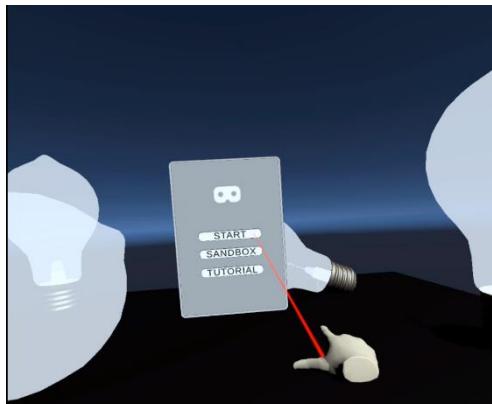


Abbildung 66: Ausschnitt im Hauptmenü

### 9.2.3 Tutorial-Szene

Die Tutorial-Szene übernimmt die Aufgabe dem Benutzer die verschiedenen Spielmechaniken Schritt für Schritt beizubringen. Die einzelnen Schritte werden auf der Wandtafel in Textform beschrieben. Zu jedem Schritt leuchtet das zu benutzende oder zu manipulierende Objekt gelb auf (Ersichtlich in *Abbildung 67*). Sobald der Spieler den angezeigten Schritt erfolgreich abgeschlossen hat, hört das Spielobjekt auf zu leuchten und auf der Wandtafel wird der nächste Schritt angezeigt. Das Tutorial endet damit, dass der Spieler aufgefordert wird über den Menübutton auf dem Controller, das Menü aufzurufen und die Tutorial-Szene über den Exit-Button zu verlassen.



Abbildung 67: Tutorial mit hervorgehobenem Spielobjekt

### 9.2.4 Spiel-Szene

Bei der Spiel-Szene handelt es sich um die Hauptszene des Serious Games. Hier werden dem Spieler die Aufgaben anhand des didaktischen Konzepts Schritt für Schritt gestellt. Es werden dem Spieler insgesamt sechs Aufgaben gestellt. Beim Laden der Hauptszene wird der Spieler in der Mitte des Raumes platziert, mit Ausrichtung Richtung Wandtafel und Tisch (Ersichtlich in *Abbildung 68*). Er kann sich von hier frei im ganzen Raum bewegen.

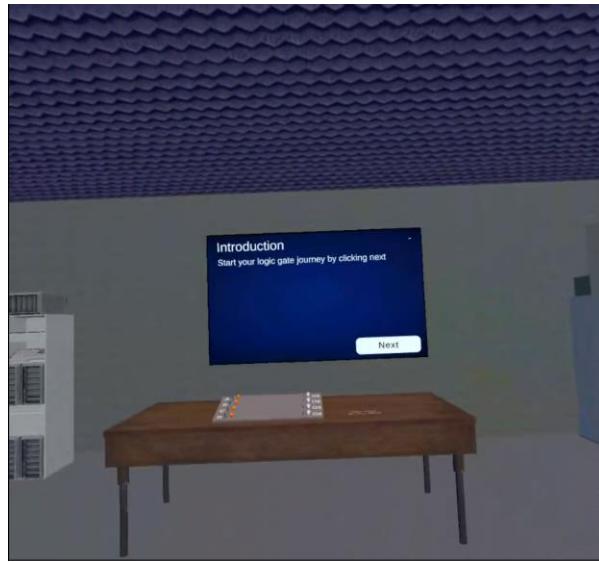


Abbildung 68: Startpunkt in der Hauptszene

#### 9.2.4.1 Spiel Ablauf in der Spiel-Szene

Am Anfang des Spiels, wird der Spieler anhand von Text auf der Wandtafel begrüßt und aufgefordert, das Spiel zu starten. Die Aufgabe wird auf der Wandtafel anhand Text und jeweiligem Bild zur Wahrheitstabelle des zu erstellenden Gatters angezeigt (Ersichtlich in Abbildung 70).

Der Spieler baut dann mit den bereits vorhanden / erspielten Gattern das neue Gatter auf dem Board. Zu jedem Zeitpunkt können Gatter aus dem Automaten bezogen werden (Ersichtlich in Abbildung 69). Nicht gebrauchte Gatter können gelöscht werden, indem man sie auf den Boden fallen lässt.



Abbildung 70: Beispiel Aufgabe



Abbildung 69: Gatter-Automat

Mit einem Überprüfen Knopf, der sich auf der Wandtafel mit den Aufgaben befindet, kann festgestellt werden, ob der Spieler die Aufgabe auf dem Board richtig gelöst hat. Anhand von Audiofeedback wird der Spieler über den Erfolg oder Misserfolg seiner Aufgabe informiert. Bei erfolgreichem Lösen der Aufgabe, wird das erstellte Gatter im Automaten für weitere Aufgaben zur Verfügung gestellt.

Das Spiel ist erfolgreich beendet, wenn alle sechs Aufgaben gelöst sind und alle sieben Gatter (NAND, NOT, AND, OR, XOR, Halbaddierer und Volladdierer) im Automaten zur Verfügung stehen (Ersichtlich in Abbildung 71). Das Spiel kann zu jedem Zeitpunkt über das Spielmenü beendet oder neu gestartet werden. Der Fortschritt wird **nicht** gespeichert und das Spiel beginnt vom Anfang an.



Abbildung 71: Erfolgreiches Abschliessen einer Aufgabe

#### 9.2.4.2 Punkte / Highscore in der Spielszene

Sobald die erste Aufgabe gestartet wird, beginnt ein Score herunterzählen. Jede Sekunde wird ein Punkt abgezogen. Der Spieler kann sich mit Hilfe von Punkten Hinweise zu den einzelnen Aufgaben auf einer anderen Wandtafel anzeigen lassen (Ersichtlich in Abbildung 73 und Abbildung 74). Jede Aufgabe hat zwei Hinweise und eine Lösung, die bezogen werden können. Jeder Hinweis kostet Punkte, nach zwei Hinweisen kann sich der Spieler eine Lösung zur Aufgabe mit Punkten kaufen (Ersichtlich in Abbildung 72).

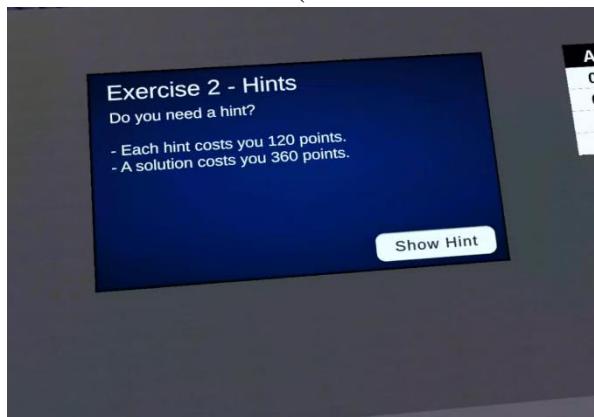


Abbildung 72: Hinweise zur aktuellen Aufgabe

Das Ziel ist es, das Spiel mit möglichst vielen Punkten zu beenden, d.h. je schneller der Spieler ist und je weniger Hinweise und Lösungen der Spieler bezieht, desto mehr Punkte hat er beim Abschliessen aller Aufgaben.

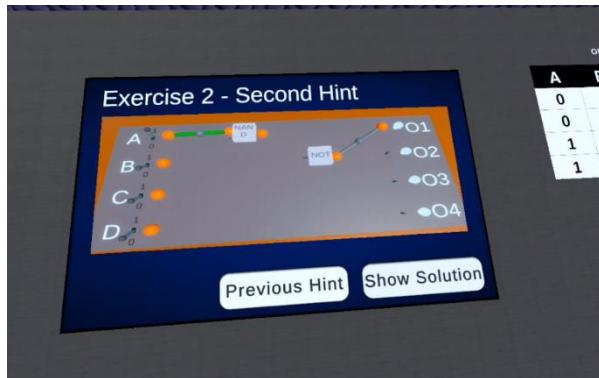


Abbildung 73: 2. Hinweis

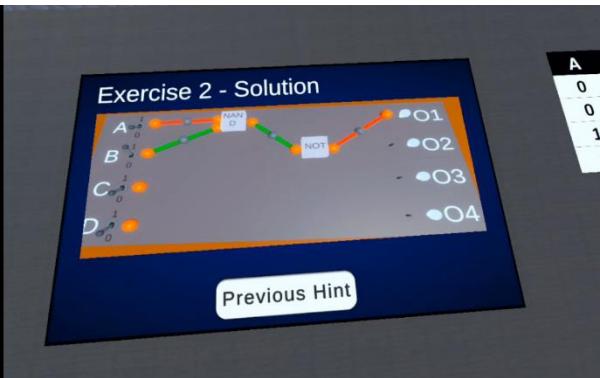


Abbildung 74: Lösung



Abbildung 75: High Score

Der Highscore wird persistent gespeichert und neben dem Board auf dem Tisch wie auf **Error! Reference source not found.** angezeigt. Die Hauptaufgabe des Scores ist es, den Spieler zu motivieren ohne Hinweise und Lösungen zu arbeiten und das Spiel mehrmals zu spielen, um sich zu verbessern.

### 9.2.5 Sandbox-Szene

Bei der Sandbox-Szene handelt es sich um die experimentelle Szene des Serious Games. In der Sandbox-Szene stehen von Anfang an alle Gatter (NAND, NOT, AND, OR, XOR, Halbaddierer und Volladdierer) im Automaten zur Verfügung. Die Hauptaufgabe der Sandboxszene ist es den Spieler in seinem eigenen Tempo, ohne Zeit- oder Punktedruck, lernen und entdecken zu lassen. Hier können nach Lust und Laune alle Kombinationen der verschiedenen Gatter ausprobiert werden, um die Gatter als einzelne Komponenten besser zu verstehen.

### 9.3 Akustik

Das Spiel ist mit Soundeffekten und Hintergrundgeräuschen ausgestattet, um eine höhere Immersion zu erzeugen. In der folgenden Tabelle werden aller Aktionen aufgezeigt, welche einen Soundeffekt ausführen. Alle aufgeführten Soundeffekte sind gratis und können lizenzenfrei gebraucht werden.

Aktion	Soundeffekt	Quelle
Gatter auf den Boden schmeissen	Lösch-Geräusch	<a href="https://www.fesliyanstudios.com/royalty-free-sound-effects-download/glass-shattering-and-breaking-124">https://www.fesliyanstudios.com/royalty-free-sound-effects-download/glass-shattering-and-breaking-124</a>
Gatter auf das Board snappen	Snap-Geräusch	<a href="https://mixkit.co/free-sound-effects/click/-&gt; hard click">https://mixkit.co/free-sound-effects/click/-&gt; hard click</a>
Kabel auf ein Kabelanker snappen	Snap-Geräusch	<a href="https://mixkit.co/free-sound-effects/click/-&gt; hard click">https://mixkit.co/free-sound-effects/click/-&gt; hard click</a>
Kabel wegziehen	Snap-Geräusch	<a href="https://mixkit.co/free-sound-effects/click/-&gt; hard click">https://mixkit.co/free-sound-effects/click/-&gt; hard click</a>
Automaten Knopf bedienen	Knopf-Geräusch	<a href="https://mixkit.co/free-sound-effects/click/">https://mixkit.co/free-sound-effects/click/</a>
Schalter auf dem Board umlegen	Schalter-Geräusch	<a href="https://mixkit.co/free-sound-effects/discover/switch/">https://mixkit.co/free-sound-effects/discover/switch/</a>
Aufgabe überprüfen – Erfolg	Erfolgs-Geräusch	<a href="https://mixkit.co/free-sound-effects/win/">https://mixkit.co/free-sound-effects/win/</a>
Aufgabe überprüfen – Misserfolg	Misserfolgs-Geräusch	<a href="https://mixkit.co/free-sound-effects/buzzer/">https://mixkit.co/free-sound-effects/buzzer/</a>
Spiel starten	Stadt-Hintergrund-Geräusch (Dauerschleife)	<a href="https://www.sfxbuzz.com/download/16-background-noise-sound-effects/217-city-traffic-close-background-noise-sound-effects">https://www.sfxbuzz.com/download/16-background-noise-sound-effects/217-city-traffic-close-background-noise-sound-effects</a>

Tabelle 120: Akustik

## 9.4 Gestellte Aufgaben und Lösungen

Im folgenden Abschnitt werden Musterlösungen zu den gestellten Aufgaben aufgeführt. Diese Musterlösungen sind immer nur eine von mehreren Möglichkeiten die Aufgabe zu lösen. Die Aufgaben werden wie in den unteren Abbildungen als Wahrheitstabelle an der Wandtafel im Spiel angezeigt.

### 9.4.1 NOT Aufgabe

In der untenstehenden Abbildung sieht man die NOT Aufgabe mit zugehöriger Musterlösung.

Aufgabe	Lösung						
<table border="1"> <thead> <tr> <th colspan="2">A      O1</th> </tr> </thead> <tbody> <tr> <td>0</td><td>1</td></tr> <tr> <td>1</td><td>0</td></tr> </tbody> </table>	A      O1		0	1	1	0	
A      O1							
0	1						
1	0						

Abbildung 76: NOT Aufgabe mit dazugehöriger Musterlösung

### 9.4.2 AND Aufgabe

In der untenstehenden Abbildung sieht man die AND Aufgabe mit zugehöriger Musterlösung.

Aufgabe	Lösung															
<table border="1"> <thead> <tr> <th>A</th> <th>B</th> <th>O1</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>0</td></tr> <tr> <td>0</td> <td>1</td> <td>0</td></tr> <tr> <td>1</td> <td>0</td> <td>0</td></tr> <tr> <td>1</td> <td>1</td> <td>1</td></tr> </tbody> </table>	A	B	O1	0	0	0	0	1	0	1	0	0	1	1	1	
A	B	O1														
0	0	0														
0	1	0														
1	0	0														
1	1	1														

Abbildung 77: AND Aufgabe mit dazugehöriger Musterlösung

#### 9.4.3 OR Aufgabe

In der untenstehenden Abbildung sieht man die OR Aufgabe mit zugehöriger Musterlösung.

Aufgabe			Lösung
A	B	O1	
0	0	0	
0	1	1	
1	0	1	
1	1	1	

Abbildung 78: OR Aufgabe mit dazugehöriger Musterlösung

#### 9.4.4 XOR Aufgabe

In der untenstehenden Abbildung sieht man die XOR Aufgabe mit zugehöriger Musterlösung.

Aufgabe			Lösung
A	B	O1	
0	0	0	
0	1	1	
1	0	1	
1	1	0	

Abbildung 79: XOR Aufgabe mit dazugehöriger Musterlösung

#### 9.4.5 Halbaddierer Aufgabe

In der untenstehenden Abbildung sieht man die Halbaddierer Aufgabe mit zugehöriger Musterlösung.

Aufgabe				Lösung
HALFADDER		CARRY	SUM	
A	B	O1	O2	
0	0	0	0	
0	1	0	1	
1	0	0	1	
1	1	1	0	

Abbildung 80: Halbaddierer Aufgabe mit dazugehöriger Musterlösung

#### 9.4.6 Volladdierer Aufgabe

In der untenstehenden Abbildung sieht man die Volladdierer Aufgabe mit zugehöriger Musterlösung.

Aufgabe					Lösung
FULLADDER			CARRY	SUM	
A	B	C	O1	O2	
0	0	0	0	0	
0	0	1	0	1	
0	1	0	0	1	
0	1	1	1	0	
1	0	0	0	1	
1	0	1	1	0	
1	1	0	1	0	
1	1	1	1	1	

Abbildung 81: Volladdierer Aufgabe mit dazugehöriger Musterlösung

## 10 Diskussion

Um einen besseren Einblick in die Entwicklung dieser Arbeit zu gewährleisten, werden in diesem Kapitel die Probleme und deren Lösungen, und erfolgreiche Lösungsansätze genauer besprochen. Danach wird einen kurzen Einblick in die wichtigsten Punkte des Projektablaufs gegeben.

### 10.1 Oculus Integration

Unity stellt Entwicklern das XR Interaction Toolkit zur Verfügung, um verschiedene VR Headsets von unterschiedlichen Herstellern gleichwertig ansprechen zu können. Alternativ bietet Oculus eine eigene API, um exklusiv mit den Oculus Headsets und deren Features interagieren zu können. Dadurch werden Grundfunktionalitäten wie der Blickwinkel des Headsets, Teleportation und Grabbing bereits zur Verfügung gestellt und müssen nicht selbst entwickelt werden.

Das XR Interaction Toolkit ist nicht mit der Oculus Integration kompatibel. Das bedeutet, dass verschiedene Funktionalitäten unter anderem die Snapzone, welche von XR bereits «Out-Of-The-Box» angeboten werden, nicht verwendet werden können. Für das Serious Game musste dann eine eigene Snapzone entwickelt werden, da die Oculus Integration diese nicht anbietet.

Der Verwendung des Oculus Integration SDK's ist anhand des heutigen Standes (Juni 22) eher abzuraten.

#### 10.1.1 Oculus Integration Bugs

Während der Entwicklung des Serious Game ist ein besonders tückischer Bug aufgefunden worden.

Falls der Spieler mit einer Hand zwischen zwei Grabbables greift (Grabbables nah beieinander: Bsp. Zwischen Gatter und Kabel), kann es passieren, dass eine Hand durch eine Exception in einen ungültigen Zustand fällt. Dieser Zustand führt dazu, dass die Hand keine Grabbables mehr halten kann. Der Zustand kann zur Laufzeit des Serious Games nicht Rückgängig gemacht werden und führt dazu, dass die Hand des Spielers unbrauchbar wird. Die Hand bewegt sich weiterhin relativ zum Controller und öffnet und schliesst sich noch immer durch das Interagieren mit dem Hand-Trigger, doch kann nichts mehr aufgenommen oder festgehalten werden. Beim Eintreten des Fehlers verändert sich optisch nichts, dem Spieler fällt das Problem also erst auf, wenn er probiert ein neues Grabbable zu greifen.

Durch das Auslesen aller Logs mit Logcat, konnte man den Nullpointer grob auf ein initiales Skript von Oculus zurückführen.

Nach mehreren Versuchen das initiale Skript von Oculus anzupassen, musste man sich mit einem Workaround zufriedengeben. Zum jetzigen Stand kann das Problem nur durch Neuladen des Serious Games behoben werden.

## 10.2 Debugging

Oculus bietet den Entwicklern die Möglichkeit durch Oculus Link ein Spiel durch Unity direkt zu starten und zu debuggen. Die Voraussetzung ist, dass der PC über eine der von Meta vorgeschriebenen Grafikkarten<sup>14</sup> verfügt. Sofern keine dieser Grafikkarten verwendet wird, besteht keine Möglichkeit ein Spiel direkt auf dem Headset zu debuggen.

Im Verlauf der Projektarbeit wurde das Plugin Logcat<sup>15</sup> eingebunden. Dieses Plugin ermöglicht es die Logmeldungen auf einem, mit dem PC verbundenen Android Gerät auszulesen. Durch das Platzieren von Logmeldungen im Code konnten diverse Bugs danach schneller identifiziert und behoben werden.

## 10.3 Snapzone

Die Snapzone ist eine der wichtigsten Komponenten in dem Serious Game, da diese als Dreh- und Angelpunkt der wichtigsten Interaktionen (Gatter und Kabel platzieren) fungiert. Die Snapzone muss diverse Spezialfälle abdecken. Aufgrund der erschwerten Debugging-Situation konnten fehlerhafte Interaktionen nur sehr aufwendig analysiert und korrigiert werden.

Aufgrund der vielen Abhängigkeiten, welche für eine Snapzone erfüllt werden müssen, wurden manuelle Test gegenüber den automatisierten Tests vorgezogen. Dies führte im Gegenzug dazu, dass für jede Änderung ein neuer, vollständiger Build auf das Headset geladen werden musste. Pro Build belief sich der Prozess auf ca. 5 Minuten, was die ganze Prozedur zeitlich ziemlich aufblähte.

## 10.4 Löschen eines Gatters

Aufgrund der Objekthierarchie eines Gatters, konnte das Löschen nicht so einfach implementiert werden, wie initial angenommen. Das Problem lag daran, dass OVRGrabbable<sup>16</sup> ein Objekt, nachdem es in die Hand genommen wurde, nicht mehr in die ursprüngliche Hierarchie zurückgesetzt hat. Das bedeutet, dass einzelne Elemente, welche ursprünglich innerhalb der Gatter Hierarchie (z.B. Kabel, Würfel) lebten, im Verlauf des Spiels plötzlich ausserhalb der Gatterhierarchie existierten. Dies hatte zur Folge, dass sobald das Gatter gelöscht werden sollte (Löschen des Root-Element in der Gatter Hierarchie), nicht mehr alle ursprünglichen Kinder-Elemente automatisch mitgelöscht wurden. Auf den untenstehenden Abbildungen: *Abbildung 83* und *Abbildung 84* kann man die Hierarchieänderung nach dem Aufnehmen mit der Hand nachvollziehen.

---

<sup>14</sup> <https://store.facebook.com/de-de/help/quest/articles/headsets-and-accessories/oculus-link/oculus-link-compatibility/>

<sup>15</sup> <https://developer.android.com/studio/command-line/logcat>

<sup>16</sup> OVRGrabbable ist eine Komponente, welche Interaktionen mit der Hand ermöglicht und vom Oculus Integration Framework zur Verfügung gestellt wird.

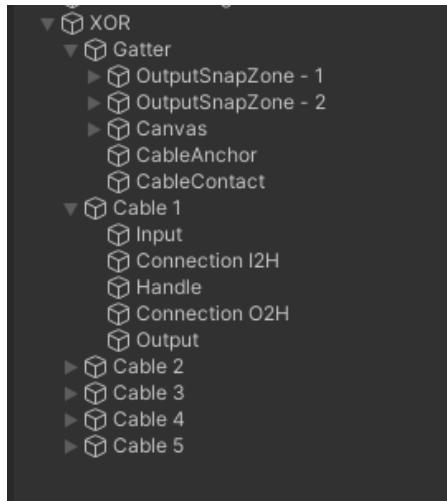


Abbildung 83: Hierarchie, bevor Gatter in die Hand genommen wurde

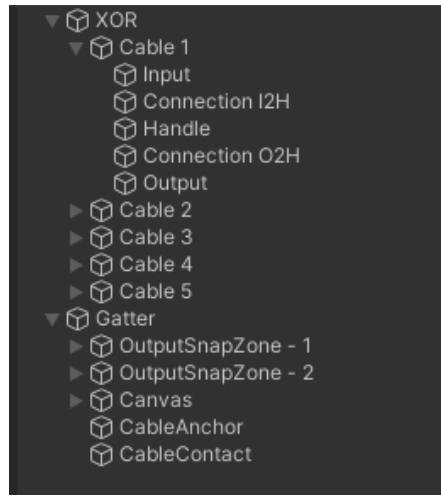


Abbildung 82: Hierarchie, nachdem Gatter in die Hand genommen wurde

In der Abbildung 84 und Abbildung 85 kann man erkennen, wie sich der Löschfehler in der Spielszene ersichtlich gemacht hat.

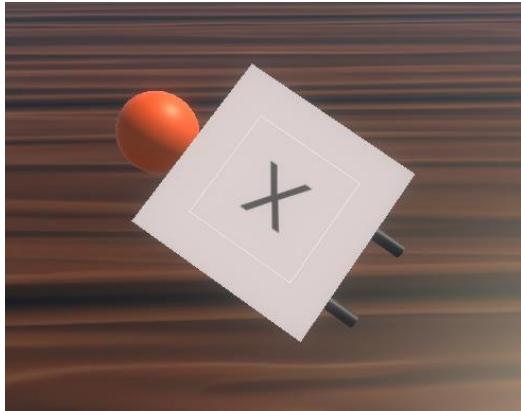


Abbildung 85: Gatter vor dem Löschen.

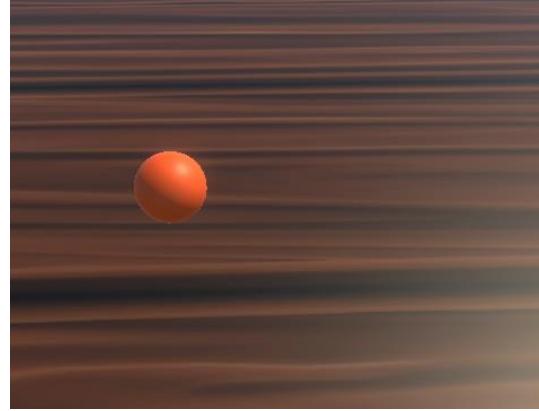


Abbildung 84: Gatter nach dem Löschen.

## 10.5 Strom Rekursiv berechnen

Zu Beginn war die Idee, dass der Strom live für jede Komponente rekursiv berechnet wird. Das bedeutete jedoch, dass jede nachfolgende Komponente doppelten Rechenaufwand betreiben muss. Beim Durchführen von Play-Tests wurde ersichtlich, dass bereits bei wenigen Komponenten in der Szene, die Performance stark leidet. Ersichtlich wurde dies durch eine niedrige Frame-Rate und durch Zittern der Szene.

Eine Lösung für dieses Problem war das Einführen eines Cache in Kombination mit der Event Driven Architecture. Jedes Element besitzt nun einen Cache, welcher den aktuellen Strom zwischenspeichert. Sobald ein stromändernder Event auftritt, wird dieser neu berechnet und der Cache aktualisiert.

## 10.6 Event Driven Architecture

Wie im *Abschnitt 7.1.6* beschrieben, basiert das Serious Game auf einer Event Driven Architecture. Im Falle des Serious Games bedeutet das, dass jede Komponente, die im Stande ist, selbst Folgeinteraktion (Events) auszulösen, ein UnityEvent zugeschrieben bekommt.

Durch die daraus entstehende Invertierung der Abhängigkeit (Technisch: Dependency Inversion<sup>17</sup>) müssen Komponenten keine hart-codierten Logik-Abläufe implementieren und können dafür im Unity-Editor konfigurativ beliebig erweitert werden.

### 10.6.1 Sound

Anstatt dass jedes Element sich selbst um Soundeffekte kümmert, übernimmt das eine eigenständige Komponente SFXManager<sup>18</sup>. Durch UnityEvents kann der SFXManager angesprochen werden und einzelne Soundeffekte abspielen.

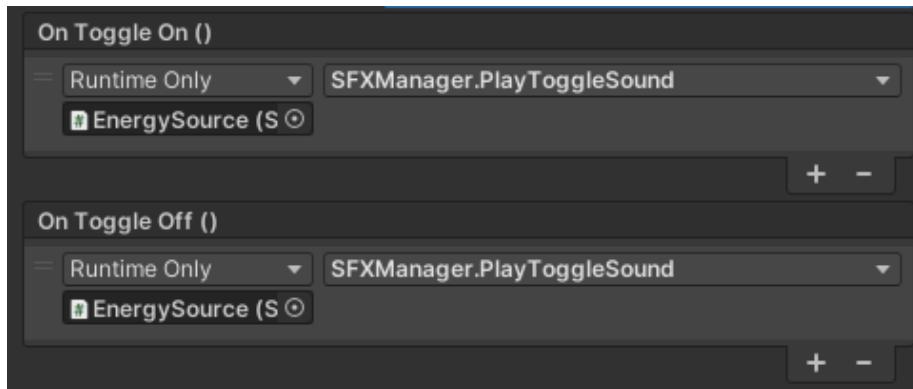


Abbildung 86: Soundeffekte durch SFXManager abspielen.

Der SFXManager hält alle Soundeffekte, welche in dem Spiel verwendet werden. Neue Soundeffekte können einfach hinzugefügt werden.

<sup>17</sup> Mehr zur Dependency-Inversion: <https://deviq.com/principles/dependency-inversion-principle>

<sup>18</sup> SFXManager ist eine selbst entwickelte Klasse zum Abspielen von Soundeffekten.

	SFX Manager (Script)	
Script	SFXManager	(○)
Snap	None (Audio Source)	(○)
Button Click	None (Audio Source)	(○)
Toggle	None (Audio Source)	(○)
Cable Snap	None (Audio Source)	(○)
Cable Un Snap	None (Audio Source)	(○)
Ambient	None (Audio Source)	(○)
Shatter	None (Audio Source)	(○)
Right Answer	None (Audio Source)	(○)
Wrong Answer	None (Audio Source)	(○)

Abbildung 87: Übersicht mit allen verfügbaren Soundeffekten.

### 10.6.2 Tutorial

Durch die Event Driven Architecture war es möglich das Tutorial modular und einfach erweiterbar aufzubauen.

Zum Beispiel kann eine Interaktion „Ein Gatter in die Hand nehmen“ oder „Ein Kabel mit einem Gatter verbinden“ einfach durch einen Event abgebildet werden. Generell konnte das Tutorial dadurch wie eine State-Machine aufgebaut werden, welche sehr einfach veränder- und erweiterbar ist. Ein Game-Designer hat dadurch die Flexibilität das Tutorial beliebig nach seinen Wünschen anzupassen.

### 10.6.3 Präsentation

Die ScriptableObjects bieten dem Entwickler eine einfache Möglichkeit, konfigurationslastige Strukturen vom Code, in den Editor zu verlagern. Mit wenigen Zeilen Code wurde es möglich, dass der Entwickler komplexe Präsentationen mit Text und Bildern, nur noch durch Mausklicks erstellen kann. Durch diese einfach Editierbarkeit können auch spätere Modifikationen leicht vorgenommen werden. Beispiele dafür sind die Reihenfolge der Folien oder sogar ganzen Aufgaben im Nachhinein anzupassen.

In einem professionellen Spielentwickler-Umfeld würde diese Implementation eine Trennung zwischen den Rollen Entwickler und Game-Designer ermöglichen.

### 10.6.4 Animationen als Eventhelfer

Der Animator bietet dem Entwickler die Möglichkeit einzelne Eigenschaften von Objekten durch eine Zeitsteuerung zu bearbeiten. Ein Beispiel dafür wäre, dass ein Objekt über eine bestimmte Zeitspanne wächst. Ein anderes Beispiel wäre, dass sich ein Objekt von Position A nach Position B bewegt. Dies kann durch das Verändern der grundlegenden Eigenschaften wie Grösse und Position eines Objekts umgesetzt werden.

Um jedoch komplexere Animationen abbilden zu können, kann der Animator sogar auf Domain-spezifische Eigenschaften; wie das Material eines Shaders oder den Statuswert eines Switches zugreifen. Somit konnten die Animationen für die Glühbirnen ohne spezifischen Code für das Layout realisiert werden.

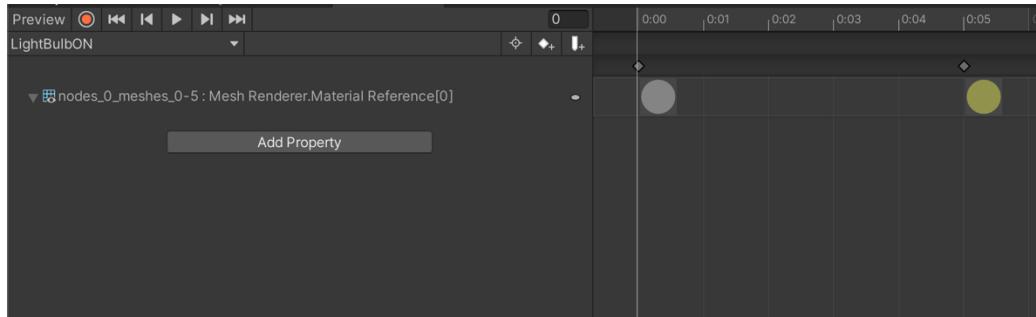


Abbildung 88: Animation der Glühbirne welche das Material nach einem kurzen Moment austauscht.

In der obenstehenden *Abbildung 88* sieht man den Animationseditor in Unity für die Glühbirne. Zum Zeitpunkt [0:00] wird das graue Material verwendet. Nach 5 Frames wird (Zeitpunkt [0:05]) das gelbe Material gesetzt.

Dank der Animation hat das «Lightbulb» Skript keine Abhängigkeit zum Material. Die Lightbulb teilt jediglich dem Glühbirnen Animator mit, dass die «LightBulbON» oder «LightBulbOFF» Animation abgespielt werden soll.

Auch hier würde diese Umsetzung eine Trennung zwischen den Rollen Entwickler und Visual-Designer ermöglichen.

## 10.7 Projektablauf

Im folgenden Abschnitt werden die verschiedenen Themen zum Projektablauf näher erläutert.

### 10.7.1 Pairprogramming

Durch das erschwerete Debugging hat sich im Verlauf des Projektes das Pairprogramming als effizienteste Methode für die Entwicklung des Codes entpuppt. Dadurch wurden ca.  $\frac{3}{4}$  des Codes im Pairprogramming entwickelt und getestet. Das Pairprogramming wurde vor Ort am selben Computer durchgeführt, dies erleichterte die Zusammenarbeit immens.

### 10.7.2 Spiel-Tests

Als der erste spielbare Prototype und explizit das Tutorial Form annahm, wurde wöchentlich das Serious Game externen Personen zum Testen bereitgestellt. Dies half vor allem Probleme in der Handhabung der Interaktionen zu entdecken, an die man sich als Entwickler bereits gewöhnt hatte. Durch das Einführen des Tutorials und Verbesserung anhand des Feedbacks konnten im Nachhinein Spielabläufe und Aktionen angepasst werden, sodass das Serious Game intuitiver zu bedienen wurde. Ohne die regelmässigen Spieltests wären weniger Fehler und Kuriositäten aufgefallen.

### 10.7.3 Nicht entwickelte Use-Cases

In der untenstehenden Abbildung kann man die sieben Use-Cases einsehen, die es anhand von Zeitgründen nicht bis in die Entwicklung geschafft haben.

Backlog 7 issues		Create sprint	...
<input checked="" type="checkbox"/>	Label für Pseudo-Gatter bestimmen	BA22-77	= 2
<input checked="" type="checkbox"/>	Pseudo-Gatter erstellen	BA22-76	= 8
<input checked="" type="checkbox"/>	Kreide halten	BA22-71	= -
<input checked="" type="checkbox"/>	Striche mit Kreide platzieren	BA22-72	= -
<input checked="" type="checkbox"/>	Schwamm halten	BA22-73	= -
<input checked="" type="checkbox"/>	Striche mit Schwamm entfernen	BA22-74	= -
<input checked="" type="checkbox"/>	Board leeren	BA22-75	= -

Abbildung 89: Nicht entwickelte Use-Cases im Backlog

#### 10.7.3.1 Notizen

Use Cases: UC-20, UC-21, UC-22 und UC-23

Ein geplanter Usecase ist Notizen an einer Wandtafel zu erstellen. Der Spieler hat Kreide und einen Schwamm zu Verfügung, um freihand Notizen an der Wandtafel zu erstellen. Mit dem Schwamm kann er seine Notizen wieder entfernen. Dieser Usecase konnte aus Zeitgründen nicht umgesetzt werden.

#### *10.7.3.2 Board leeren*

Use Case: UC-24

Ein weiteres Ziel war es, dass man anhand eines Knopfes oder Button das ganze Board mit allen Verbindungen leeren kann. Dies wurde aber nicht so umgesetzt, sondern man kann nun an jedem einzelnen Gatter ziehen und alle verbundenen Kabel werden vom Board entfernt.

#### *10.7.3.3 Pseudo Gatter*

Use Cases: UC-25 und UC-26

Das Erstellen der Pseudo-Gatter ist ein Use-Case, der wegen seiner Komplexität nicht mehr umgesetzt werden konnte. Das Ziel war es, dass der Spieler anhand seiner Board Konfiguration (alle Gatter auf dem Board) ein eigenes Gatter mit eigenem ausgewähltem Label erstellen kann.

# 11 Ausblick

In diesem letzten Kapitel wird sich genauer mit dem Ausblick und der Zukunft dieses Projektes auseinandergesetzt.

## 11.1 Nicht entwickelte Use-Cases

Als unmittelbar nächsten Schritt für das Serious Game würde die Entwicklung der nicht umgesetzten Use-Cases anstehen.

## 11.2 Multiplayer

Ein klarer nächster grösserer Schritt für das Serious Game ist das Einführen eines Multiplayer-Modus. Da das Serious Game als didaktische Hilfestellung für den Hochschulunterricht entwickelt wurde, macht es mehr Sinn, dass Spiel für mehrere Studenten oder Spieler zu öffnen. In Zusammenarbeit mit Kollegen, können die doch eher komplexeren Aufgaben leichter gemeistert werden.

## 11.3 Erweiterung Aufgaben & Komponenten

Zum jetzigen Stand spielt man das Spiel vom NAND-Gatter bis und mit Volladdierer durch. Dies könnte bis zu Prozessoren ausgebaut werden. Der Ablauf würde wie folgt aussehen:

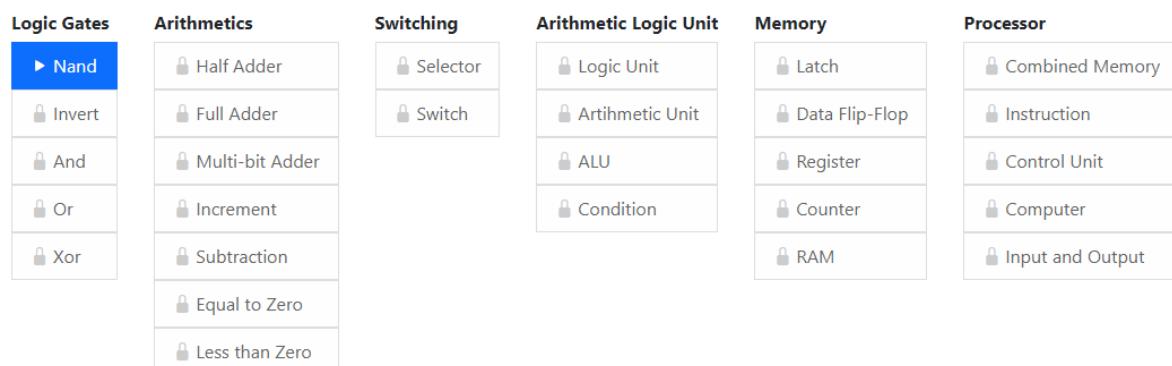


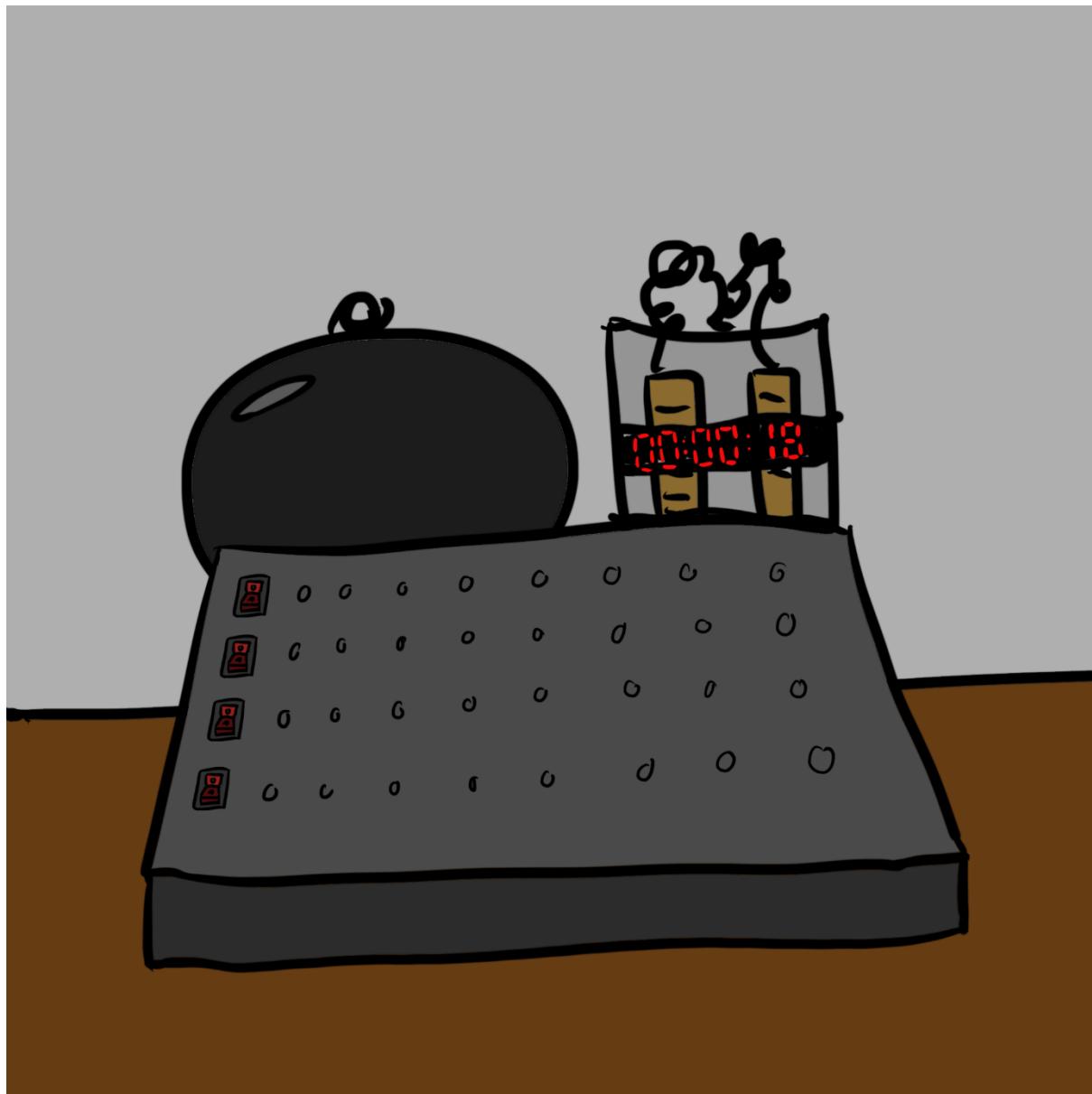
Abbildung 90: NAND Game Ablauf [18]

Je nach Erweiterung müssen auch neue Komponenten im Spiel hinzugefügt werden. Diese kann man dann ohne grossen Aufwand aus den jetzigen Prefabs erstellen.

## 11.4 Gamification

Gamification bedeutet, in einer spielfremden Umgebung Spielelemente einzubauen. Konkret heisst das für unser Serious Game, dass das Spiel in einem nächsten Schritt in eine Spielumgebung eingebettet werden soll. Anstatt die Gatterlogiken in Vordergrund zu haben und linear die Aufgaben durchzuarbeiten, würde die Gatterlogiken als Voraussetzung oder Nebenbedingung gelten, und der Hauptfokus liegt auf einem Puzzle-Spiel, wie zum Beispiel aus einem Labor auszubrechen, indem man eine elektrische Tür mit Schalterlogik knackt, oder eine Bombe zu entschärfen. Wie im Abschnitt 7.1.6 beschrieben, basiert das Serious Game auf einer Event Driven Architecture, im Falle der Gamification heisst das, dass das Spiel in alle vorstellbaren Szenarien eingebettet werden kann, ohne etwas am jetzigen Stand

des Codes zu ändern. In der untenstehenden *Abbildung 91* kann man eine erste Idee zur Gamification erkennen.



*Abbildung 91: Beispiel für Gamification – Bombe entschärfen.*

## 12 Literaturverzeichnis

- [1] Wirtschaftslexikon, „Agile Softwareentwicklung,“ [Online]. Available: <https://wirtschaftslexikon.gabler.de/definition/agile-softwareentwicklung-53460>. [Zugriff am 06 03 2022].
- [2] Microsoft, „C Sharp,“ [Online]. Available: <https://docs.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/>. [Zugriff am 06 03 2022].
- [3] Unity, „Game Engine,“ [Online]. Available: <https://unity.com/how-to/beginner/game-development-terms>. [Zugriff am 06 03 2022].
- [4] Wikipedia, „Abhängigkeitnotation,“ [Online]. Available: <https://de.wikipedia.org/wiki/Abh%C3%A4ngigkeitsnotation>. [Zugriff am 06 03 2022].
- [5] Atlassian, „Jira Software,“ [Online]. Available: <https://www.atlassian.com/software/jira>. [Zugriff am 06 03 2022].
- [6] Unity3d, „Unity Documentation,“ [Online]. Available: <https://docs.unity3d.com/Manual/index.html>. [Zugriff am 06 03 2022].
- [7] sitejabber, „Unity Technologies,“ [Online]. Available: <https://www.sitejabber.com/reviews/unity3d.com>. [Zugriff am 06 03 2022].
- [8] Wired, „Virtual reality,“ [Online]. Available: <https://www.wired.com/story/what-is-xr/>. [Zugriff am 06 03 2022].
- [9] fulldome.pro, „VR PROJECTION ROOM VS VR PROJECTION DOME,“ [Online]. Available: <https://fulldome.pro/blog/vr-projection-room-vs-vr-projection-dome/>. [Zugriff am 06 03 2022].
- [10] Polygon, „THE COMPLETE GUIDE TO VIRTUAL REALITY IN 2016 (SO FAR),“ [Online]. Available: <https://www.polygon.com/2016/1/15/10772026/virtual-reality-guide-oculus-google-cardboard-gear-vr>. [Zugriff am 06 03 2022].
- [11] J. Mütterlein, mmersion, Presence, Interactivity: Towards a Joint Understanding of Factors Influencing Virtual Reality Acceptance and Use, 2007.
- [12] IGI Global, „IGI Global,“ IGI Global, [Online]. Available: <https://www.igi-global.com/dictionary/serious-games/26549>. [Zugriff am 31 05 2022].
- [13] Oculus, „Oculus for buisness,“ [Online]. Available: <https://business.oculus.com/products/specs/>. [Zugriff am 12 03 2022].
- [14] Oculus, „Oculus For Developers,“ Oculus, [Online]. Available: <https://developer.oculus.com/documentation/unity/unity-utilities-overview/>. [Zugriff am 22 05 2022].
- [15] Dataprot, „Virtual Reality Statistics,“ [Online]. Available: <https://dataprot.net/statistics/virtual-reality-statistics/>. [Zugriff am 12 03 2022].
- [16] Metacritic, „Please, Don't Touch Anything,“ [Online]. Available: <https://www.metacritic.com/game/pc/please-dont-touch-anything>. [Zugriff am 05 03 2022].
- [17] Four Quarters, „Please, Don't Touch Anything 3D,“ [Online]. Available: [https://store.steampowered.com/app/529590/Please\\_Dont\\_TouchAnything\\_3D/](https://store.steampowered.com/app/529590/Please_Dont_TouchAnything_3D/). [Zugriff am 05 03 2022].
- [18] O. J. Kjær. [Online]. Available: <https://nandgame.com/>. [Zugriff am 05 03 2022].
- [19] Mouse Hat Games, „Youtube,“ [Online]. Available: <https://youtu.be/OpTZ1geIECU>. [Zugriff am 12 03 2022].

- [20] Mouse Hat Games, „Logic World,“ [Online]. Available: <https://mousehatgames.itch.io/logic-world>. [Zugriff am 12 03 2022].
- [21] B. K., „Hostingtribunal,“ [Online]. Available: <https://hostingtribunal.com/blog/minecraft-statistics/#gref>. [Zugriff am 12 03 2022].
- [22] Torb, „YouTube,“ [Online]. Available: <https://youtu.be/tDxKhiJfgYk>.
- [23] GAME.GUIDE, „HOW TO CREATE REDSTONE LOGIC GATES IN MINECRAFT,“ 26 06 2012. [Online]. Available: <https://www.game.guide/minecraft-logic-gates>. [Zugriff am 12 03 2022].
- [24] Tomorrow Corporation, „About the Game,“ [Online]. Available: <https://tomorrowcorporation.com/humanresourcemachine>. [Zugriff am 05 03 2022].
- [25] Tomorrow Corporation, „Screenshots,“ [Online]. Available: [https://tomorrowcorporation.com/blog/wp-content/themes/tcTheme2/images/hrm/screenshots/hrm\\_06.png](https://tomorrowcorporation.com/blog/wp-content/themes/tcTheme2/images/hrm/screenshots/hrm_06.png). [Zugriff am 05 03 2022].
- [26] Wikipedia, „NAND gate,“ [Online]. Available: [https://en.wikipedia.org/wiki/NAND\\_gate](https://en.wikipedia.org/wiki/NAND_gate). [Accessed 20 05 2022].
- [27] Wikipedia, „Konjunktion,“ [Online]. Available: [https://de.wikipedia.org/wiki/Konjunktion\\_\(Logik\)](https://de.wikipedia.org/wiki/Konjunktion_(Logik)). [Zugriff am 06 03 2022].
- [28] Wikipedia, „Disjunktion,“ [Online]. Available: <https://de.wikipedia.org/wiki/Disjunktion>. [Zugriff am 06 03 2022].
- [29] Wikipedia, „XOR gate,“ [Online]. Available: [https://en.wikipedia.org/wiki/XOR\\_gate](https://en.wikipedia.org/wiki/XOR_gate). [Zugriff am 20 05 2022].
- [30] Electronic Tutorials, „Electronic Tutorials,“ [Online]. Available: [https://www.electronics-tutorials.ws/combination/comb\\_7.html#:~:text=So%20when%20adding%20binary%20numbers,for%20addition%20and%20so%20on..](https://www.electronics-tutorials.ws/combination/comb_7.html#:~:text=So%20when%20adding%20binary%20numbers,for%20addition%20and%20so%20on..) [Zugriff am 26 05 2022].
- [31] Wrike, „Wrike,“ Wrike, [Online]. Available: <https://www.wrike.com/scrum-guide/scrum-sprints/>. [Zugriff am 04 06 2022].
- [32] Unity, „Unity,“ Unity, [Online]. Available: <https://docs.unity3d.com/Packages/com.unity.xr.interaction.toolkit@0.9/manual/index.html>. [Zugriff am 22 05 2022].
- [33] Minecraft Wiki, „Redstone,“ [Online]. Available: <https://minecraft.fandom.com/de/wiki/Redstone>. [Zugriff am 12 03 2022].
- [34] Acer, „Acer,“ Acer, [Online]. Available: <https://static.acer.com/up/Resource/Acer/Wearable/WMR/Design/Hotspot/20170922/Inside-Out%20Tracking.jpg>. [Zugriff am 12 03 2022].

## Abbildungsverzeichnis

Abbildung 1: Indigel Board aus dem CTIT Unterricht.....	14
Abbildung 2: Prototyp des Boards.....	15
Abbildung 3: Prototyp des Raumes .....	15
Abbildung 4: Sketch des Raumes aus der Perspektive des Spielers .....	15
Abbildung 5: Oculus Quest 2 Headset [13] .....	17
Abbildung 6: Tracking Modi eines VR Headsets [10] .....	18
Abbildung 7: Controller Mapping [14].....	18
Abbildung 8: Ausschnitt der Interaktionsmöglichkeiten [17] .....	19
Abbildung 9: Beschreibung und Aufgabe (Links) sowie das Platzieren und Verbinden der Gatter (Mitte). [18].....	20
Abbildung 10: Ausschnitt welcher das Baukastensystem sowie die Verbindungen aufzeigt. [20].....	21
Abbildung 11: Beispiel einer AND Schaltung mit Red-Stone. [23].....	22
Abbildung 12: Perspektive des Spielers mit den Instruktionen an den Mitarbeiter (Rechts). [25] .....	23
Abbildung 13: Binäre Addition [30].....	26
Abbildung 14: Use-Case Diagramm.....	30
Abbildung 15: Grober Ablauf des Projekts ohne explizite Deadlines.....	71
Abbildung 16: Sprint Ablauf [31].....	72
Abbildung 17: Hierarchie des AND-Gatter Prefab.....	73
Abbildung 18: Gatter-Logiken abgespeichert als Asset .....	74
Abbildung 19: Beispiel eines UnityEvent Editor welcher öffentliche Methoden auf anderen Objekten ausführen kann .....	76
Abbildung 20: Vogelperspektive eines einfachen Gatters mit eingefärbten Snap-Zonen .....	79
Abbildung 21: Seitenansicht eines Socket für Gatter .....	79
Abbildung 22: Klassendiagramm der Snapzone .....	80
Abbildung 23: Gatter mit mehreren Gatter Komponenten .....	82
Abbildung 24: Zuweisung einer Gatter-Logik (Volladdierer Output 1) zu einer Gatter-Komponenten .....	83
Abbildung 25: Klassendiagramm der Gatter-Logiken.....	83
Abbildung 26: Hierarchie des Kabel Prefab .....	84
Abbildung 27: Entfernen eines Kabels durch ziehen am Handle .....	85
Abbildung 28: Klassendiagramm des Kabels .....	86
Abbildung 29: Überschriebenes Label bei Exercise-Folien .....	87
Abbildung 30: Default Label für Next-Button.....	87
Abbildung 31: Konfigurationsansicht einer Tutorial-Folie .....	88
Abbildung 32: Klassendiagramm der Folien .....	89
Abbildung 33: Aufbau einer Präsentation.....	89
Abbildung 34: Klassendiagramm der Präsentation.....	90
Abbildung 35: Liste der Hint-Präsentationen .....	90
Abbildung 36: Konfiguration eines Presenters .....	91
Abbildung 37: Klassendiagramm der leitenden Objekte .....	93
Abbildung 38: Swimlane Diagramm Stromfluss im Gatter.....	95
Abbildung 39: Swimlane Diagramm zum Stromfluss im Kabel .....	96
Abbildung 40: Swimlane Diagramm Stromfluss der Quelle .....	97
Abbildung 41: Swimlane Diagramm zum Stromfluss der Destination .....	98
Abbildung 42: Zyklische Konstellation auf Board .....	99
Abbildung 43: Klassendiagramm des Gatter-Dispenser .....	100
Abbildung 44: Simples Statusdiagramm des Schalters .....	101

Abbildung 45: Simples Klassendiagramm des Toggle .....	102
Abbildung 46: Simples Statusdiagramm des Buttons.....	103
Abbildung 47: Simples Klassendiagramm des Buttons.....	103
Abbildung 48: Unity Editor zum Erstellen einer Aufgabenstellungs-Folie.....	105
Abbildung 49: Kabel mit unterschiedlichem Stromfluss .....	106
Abbildung 50: Kabel am Kabelhandle wegziehen.....	106
Abbildung 51: Beispiel Gatter mit 2 Kabel Ausgängen .....	107
Abbildung 52: Beispiel Gatter mit einem Kabel Ausgang .....	107
Abbildung 54: Leuchtendes Gatter .....	107
Abbildung 53: Kabel aus dem Gatter ziehen .....	107
Abbildung 55: 5 Kabel aus dem Gatter ziehen .....	108
Abbildung 56: Übersicht über das Board.....	109
Abbildung 57: Glühbrine leuchtet .....	109
Abbildung 58: Glühbirne leuchtet nicht .....	109
Abbildung 59: Vorschau für Kabel.....	110
Abbildung 60: Vorschau für Gatter .....	110
Abbildung 61: Automat .....	111
Abbildung 62: Wandtafel Beispiel.....	112
Abbildung 63: In-Game Menü .....	113
Abbildung 64: Design-Objekte in der Szene .....	114
Abbildung 65: Design-Objekte in der Szene .....	114
Abbildung 66: Ausschnitt im Hauptmenü .....	115
Abbildung 67: Tutorial mit hervorgehobenem Spielobjekt .....	115
Abbildung 68: Startpunkt in der Hauptszene.....	116
Abbildung 69: Gatter-Automat .....	116
Abbildung 70: Beispiel Aufgabe .....	116
Abbildung 71: Erfolgreiches Abschliessen einer Aufgabe .....	117
Abbildung 72: Hinweise zur aktuellen Aufgabe.....	117
Abbildung 73: 2. Hinweis .....	118
Abbildung 74: Lösung .....	118
Abbildung 75: High Score .....	118
Abbildung 76: NOT Aufgabe mit dazugehöriger Musterlösung .....	121
Abbildung 77: AND Aufgabe mit dazugehöriger Musterlösung.....	121
Abbildung 78: OR Aufgabe mit dazugehöriger Musterlösung.....	122
Abbildung 79: XOR Aufgabe mit dazugehöriger Musterlösung .....	122
Abbildung 80: Halbaddierer Aufgabe mit dazugehöriger Musterlösung .....	122
Abbildung 81: Volladdierer Aufgabe mit dazugehöriger Musterlösung .....	123
Abbildung 82: Hierarchie, nachdem Gatter in die Hand genommen wurde.....	126
Abbildung 83: Hierarchie, bevor Gatter in die Hand genommen wurde .....	126
Abbildung 84: Gatter nach dem Löschen. ....	126
Abbildung 85: Gatter vor dem Löschen.....	126
Abbildung 86: Soundeffekte durch SFXManager abspielen. ....	127
Abbildung 87: Übersicht mit allen verfügbaren Soundeffekten. ....	128
Abbildung 88: Animation der Glühbrine welche das Material nach einem kurzen Moment austauscht.....	129
Abbildung 89: Nicht entwickelte Use-Cases im Backlog.....	130
Abbildung 90: NAND Game Ablauf [18] .....	132
Abbildung 91: Beispiel für Gamification – Bombe entschärfen. ....	133
Abbildung 92: Konzeptionsphase in Jira .....	142
Abbildung 93: Realisierte Stories in der Konzeptionsphase.....	142

Abbildung 94: Einarbeitungsphase in Jira .....	143
Abbildung 95: Realisierte Stories in Einarbeitungsphase.....	143
Abbildung 96: Aufwand für Infrastruktur in Jira.....	144
Abbildung 97: Realisierte Stories der Infrastruktur.....	144
Abbildung 98: Prototypingphase in Jira .....	145
Abbildung 99: Realisierte Stories in der Prototypingphase .....	145
Abbildung 100: Feinschliffphase in Jira .....	146
Abbildung 101: Realisierte Stories in der Feinschliffphase .....	146
Abbildung 102: Dokumentationsphase in Jira.....	147
Abbildung 103: Realisierte Stories in der Dokumentationsphase .....	147

## Tabellenverzeichnis

Tabelle 1: Tabelle mit Abkürzungen .....	4
Tabelle 2: Tabelle mit Symbolen.....	4
Tabelle 3: Tabelle mit Glossar.....	6
Tabelle 4: Oculus Quest 2 Spezifikationen [13] .....	17
Tabelle 5: Wahrheitstabelle des NAND .....	24
Tabelle 6: Wahrheitstabelle des NOT .....	24
Tabelle 7: Wahrheitstabelle des AND .....	25
Tabelle 8: Wahrheitstabelle des OR .....	25
Tabelle 9: Wahrheitstabelle des XOR.....	25
Tabelle 10: Wahrheitstabelle zum Halbaddierer .....	26
Tabelle 11: Wahrheitstabelle zum Volladdierer .....	27
Tabelle 12: Vorlage eines Use Cases.....	31
Tabelle 13: Vorlage funktionale Anforderung.....	31
Tabelle 14: Use-Case - Spiel Starten .....	32
Tabelle 15: Funktionale Anforderung - Spielszene ist geladen .....	32
Tabelle 16: Use-Case - Tutorial Starten.....	33
Tabelle 17: Funktionale Anforderung - Tutorial Szene ist geladen.....	33
Tabelle 18: Use-Case - Sandbox starten .....	33
Tabelle 19: Funktionale Anforderung - Sandboxszene ist geladen .....	34
Tabelle 20: Use-Case - Menü aufrufen.....	35
Tabelle 21: Funktionale Anforderung - In-Game Menü wird angezeigt .....	35
Tabelle 22: Funktionale Anforderung - In-Game Menü kann bedient werden .....	36
Tabelle 23: Use-Case - Menü verstecken .....	36
Tabelle 24: Funktionale Anforderung - In-Game Menü wird nicht mehr angezeigt .....	36
Tabelle 25: Use-Case - Spielszene neu laden .....	37
Tabelle 26: Funktionale Anforderung - Spielszene wird neu geladen.....	37
Tabelle 27: Use-Case - Spielszene verlassen.....	37
Tabelle 28: Funktionale Anforderung - Spielszene wird verlassen .....	38
Tabelle 29: Use-Case - Headset bewegen.....	39
Tabelle 30: Funktionale Anforderung - Perspektive wird ausgerichtet .....	39
Tabelle 31: Use-Case - Controller bewegen .....	40
Tabelle 32: Funktionale Anforderung - Virtuelle Hände werden relativ zum Controller bewegt.....	40
Tabelle 33: Funktionale Anforderung - Virtuelle Hände werden durch Controller geschlossen .....	40
Tabelle 34: Funktionale Anforderung - Hände werden durch Controller geöffnet .....	41
Tabelle 35: Use-Case - Im Raum bewegen.....	41
Tabelle 36: Funktionale Anforderung - Spieler an beliebigen freien Ort im Raum teleportieren .....	42
Tabelle 37: Use-Case - Gatter halten .....	43
Tabelle 38: Funktionale Anforderung - Gatter wird gehalten .....	43
Tabelle 39: Funktionale Anforderung - Gatter wird losgelassen .....	43
Tabelle 40: Use-Case - Gatter platzieren .....	44
Tabelle 41: Funktionale Anforderung - Gatter wird auf freiem Socket platziert .....	44
Tabelle 42: Use-Case - Gatter entfernen.....	45
Tabelle 43: Funktionale Anforderung - Gatter wird vom Socket entfernt.....	45
Tabelle 44: Use-Case - Gatter löschen.....	45
Tabelle 45: Funktionale Anforderung - Gatter wird gelöscht.....	46
Tabelle 46: Use-Case - Kabel halten .....	47

Tabelle 47: Funktionale Anforderung - Kabel wird am Kabel-Handle gehalten.....	47
Tabelle 48: Funktionale Anforderung - Kabel loslassen .....	48
Tabelle 49: Use-Case - Kabel verbinden .....	48
Tabelle 50: Funktionale Anforderung - Kabel wird auf freiem Kabel-Eingang platziert.....	49
Tabelle 51: Use-Case - Kabel entfernen .....	49
Tabelle 52: Funktionale Anforderung - Verbundenes Kabel wird entfernt .....	49
Tabelle 53: Use-Case - Strom einschalten .....	50
Tabelle 54: Funktionale Anforderung - Schalter wird von 0 auf 1 gekippt.....	50
Tabelle 55: Funktionale Anforderung - Strom auf verbundene Objekte weitergeben.....	50
Tabelle 56: Use-Case - Strom ausschalten.....	51
Tabelle 57: Funktionale Anforderung - Schalter wird von 1 auf 0 gekippt.....	51
Tabelle 58: Funktionale Anforderung - Strom auf verbundene Objekte nicht weitergeben....	51
Tabelle 59: Use-Case - Kreide halten .....	52
Tabelle 60: Funktionale Anforderung - Kreide wird in der Hand gehalten.....	52
Tabelle 61: Funktionale Anforderung - Kreide wird losgelassen .....	53
Tabelle 62: Use-Case - Striche mit Kreide platzieren .....	53
Tabelle 63: Funktionale Anforderung - Kreide platziert Striche zwischen Berührpunkten mit Wandtafel.....	54
Tabelle 64: Use-Case - Schwamm halten .....	54
Tabelle 65: Funktionale Anforderung - Schwamm wird gehalten.....	55
Tabelle 66: Funktionale Anforderung - Schwamm wird losgelassen .....	55
Tabelle 67: Use-Case - Striche mit Schwamm entfernen .....	56
Tabelle 68: Funktionale Anforderung - Schwamm entfernt Striche zwischen Berührpunkten mit Wandtafel .....	56
Tabelle 69: Use-Case - Board leeren .....	57
Tabelle 70: Funktionale Anforderung - Board wird geleert .....	57
Tabelle 71: Use-Case - Pseudo-Gatter erstellen .....	58
Tabelle 72: Funktionale Anforderung - Pseudo Gatter ist erstellt .....	58
Tabelle 73: Use-Case - Label für Pseudo-Gatter bestimmen.....	59
Tabelle 74: Funktionale Anforderung - Gatter-Label eines Pseudo-Gatter wird durch Pfeiltasten ausgewählt.....	59
Tabelle 75: Use-Case - Aufgabe ablesen .....	60
Tabelle 76: Funktionale Anforderung - Aktuelle Aufgabe wird auf der Wandtafel angezeigt	60
Tabelle 77: Use-Case - Aufgabe überprüfen .....	61
Tabelle 78: Funktionale Anforderung - Check-Button wird angezeigt .....	61
Tabelle 79: Funktionale Anforderung - Falsch gelöste Aufgabe .....	61
Tabelle 80: Funktionale Anforderung - Richtig gelöste Aufgabe .....	62
Tabelle 81: Use-Case - Navigieren .....	63
Tabelle 82: Funktionale Anforderung - Weiter navigieren.....	63
Tabelle 83: Funktionale Anforderung - Zurück navigieren .....	63
Tabelle 84: Use-Case - Hinweise / Lösung beziehen .....	64
Tabelle 85: Funktionale Anforderung - Kosten von Score abziehen.....	64
Tabelle 86: Funktionale Anforderung - Nächsten Hinweis / Lösung anzeigen.....	64
Tabelle 87: Funktionale Anforderung - Bezahlte Hinweise / Lösungen nicht erneut bezahlen .....	65
Tabelle 88: Use-Case - Gatter auswählen .....	66
Tabelle 89: Funktionale Anforderung - Vorwärts navigieren.....	66
Tabelle 90: Funktionale Anforderung - Rückwärts navigieren .....	67
Tabelle 91: Use-Case - Gatter beziehen.....	68
Tabelle 92: Funktional Anforderung - Neue Gatter-Instanz erscheint .....	68

Tabelle 93: Use-Case - High-Score ablesen .....	69
Tabelle 94: Funktionale Anforderung - High-Score wird auf Tisch angezeigt .....	69
Tabelle 95: Use-Case - Aktueller Score ablesen.....	69
Tabelle 96: Funktionale Anforderung - Abziehen der Punkte pro Sekunde.....	70
Tabelle 97: Unity Prefab Symbole.....	73
Tabelle 98: Beschreibung der meistverwendeten Unity-Lifecycle Methoden .....	75
Tabelle 99: Kollisionstypen in Unity .....	75
Tabelle 100: Use-Cases der Headset- und Controllerinteraktionen.....	77
Tabelle 101: Events eines Grabbable.....	78
Tabelle 102: Arten von SnapZonen .....	80
Tabelle 103: Events einer Snapzone .....	80
Tabelle 104: Use-Cases der Snap-Zone .....	81
Tabelle 105: Use-Cases des Gatters.....	83
Tabelle 106: Beschreibung der Objekte innerhalb des Kabels .....	84
Tabelle 107: Use-Cases des Kabels .....	86
Tabelle 108: Spezifische Slides für Spielmodus.....	87
Tabelle 109: Inhalt einer Instanz (Präsentationskontext).....	90
Tabelle 110: Verknüpfte Presenter-Elemente .....	91
Tabelle 111: Use-Cases des Präsentations-Framework.....	92
Tabelle 112: Beschreibung der Werte des EnergyType .....	93
Tabelle 113: Abhängigkeiten der Leiter .....	94
Tabelle 114: Eingehende Events für das Gatter.....	94
Tabelle 115: Eingehende Events für das Kabel .....	96
Tabelle 116: Eingehende Events für die EnergySource .....	97
Tabelle 117: Eingehende Events der EnergyDestination.....	98
Tabelle 118: Use-Cases des Stromfluss .....	99
Tabelle 119: Objekte innerhalb des Gatter-Dispenser .....	100
Tabelle 120: Akustik.....	120

# Anhang

## A. Epic-Reports zum Projektablauf

In den untenstehenden Abbildungen sieht man zu allen sieben Epic (Konzeption, Einarbeitung, Infrastruktur, Entwicklung Prototype, Entwicklung Feinschliff und Dokumentation) die von Jira erstellten Epic Reports. Alle Use-Cases, die zum Epic gehören werden im Status Report aufgelistet.

### a. Konzeption

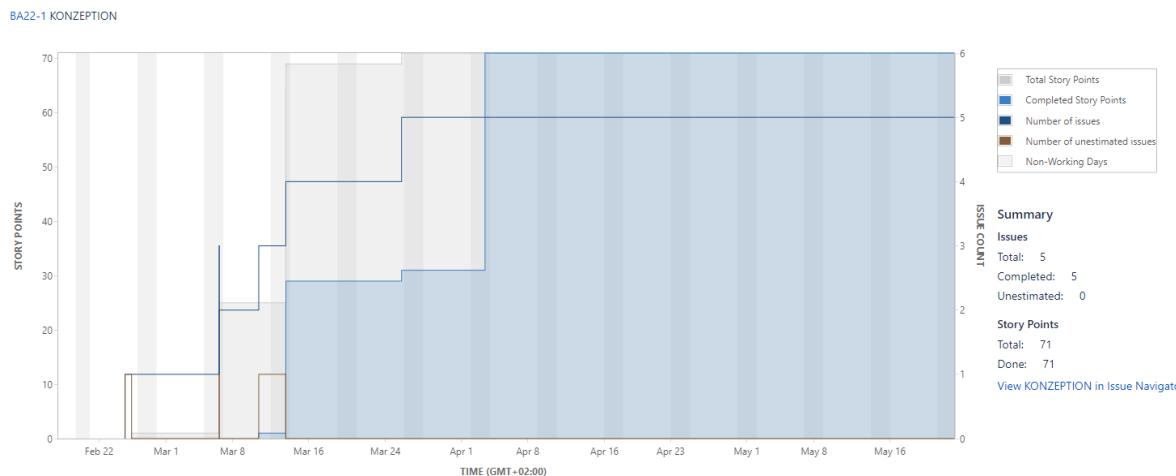


Abbildung 92: Konzeptionsphase in Jira

### Status Report

Completed Issues					View in Issue Navigator	
Key	Summary	Issue Type	Priority	Status	Story Points (71)	
BA22-41	Abklärung der Einsetzbrake im Unterricht	Story	= Medium	DONE	4	
BA22-47	Bestimmung der Use-Cases und funktionalen / nicht-funktionalen Anforderungen	Story	= Medium	DONE	40	
BA22-51	Create Paper Prototype for Scenes	Story	= Medium	DONE	2	
BA22-22	Beschreibung der Spielidee	Story	= Medium	DONE	24	
BA22-26	Didaktisches Konzept mit scia / knaa ausarbeiten	Story	= Medium	DONE	1	

Abbildung 93: Realisierte Stories in der Konzeptionsphase

## b. Einarbeitung

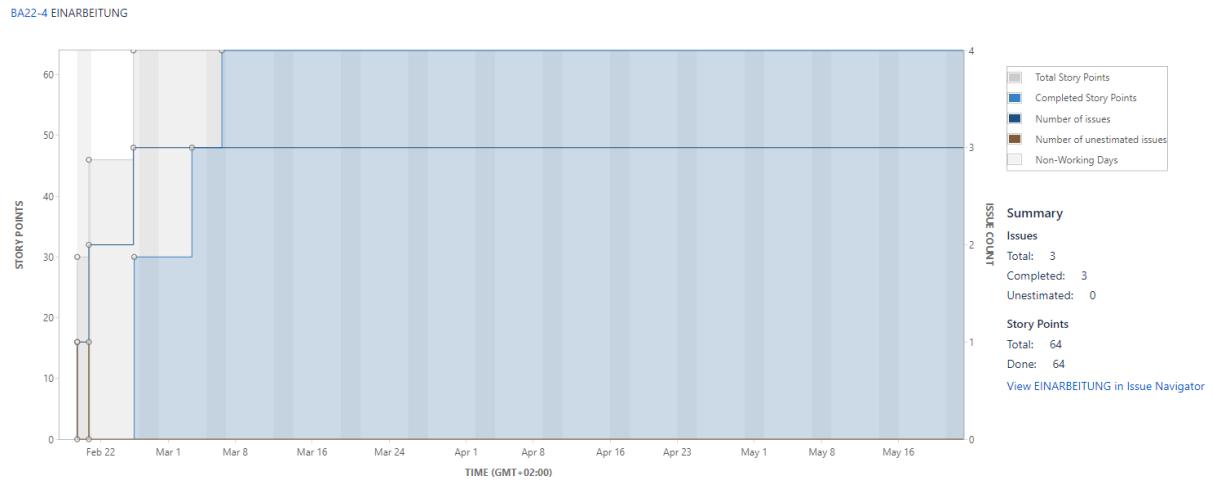


Abbildung 94: Einarbeitungsphase in Jira

Status Report					
Completed Issues			View in Issue Navigator		
Key	Summary	Issue Type	Priority	Status	Story Points (64)
BA22-7	Online C# Tutorial durchspielen	Story	Medium	DONE	30
BA22-27	Eigenes, kleines Unity-Spiel entwickeln	Story	Medium	DONE	18
BA22-21	Mit VR-Framework vertraut machen	Story	Medium	DONE	16

Abbildung 95: Realisierte Stories in Einarbeitungsphase

## c. Infrastruktur

### Epic Report

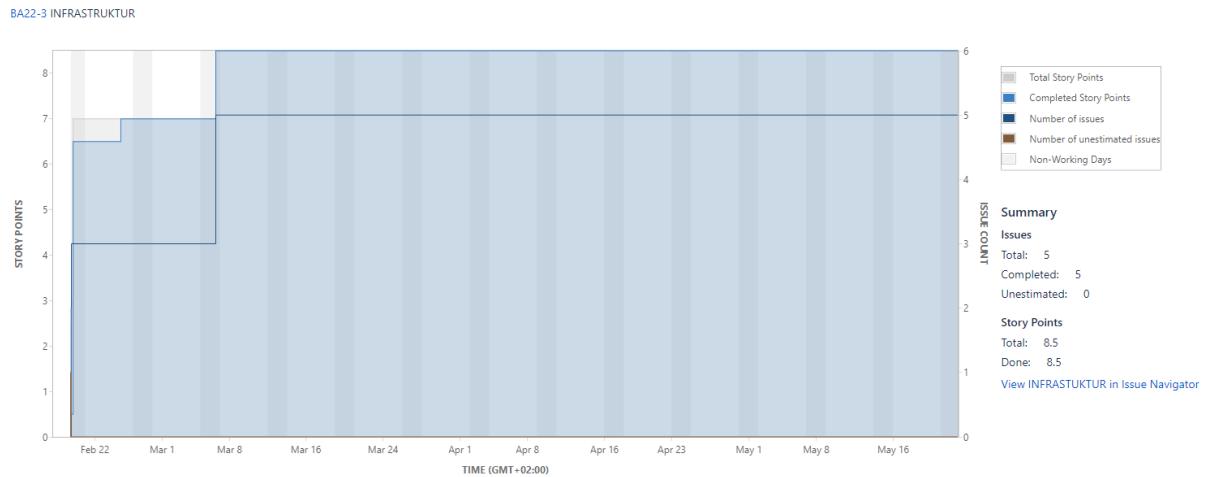


Abbildung 96: Aufwand für Infrastruktur in Jira

### Status Report

Completed Issues						<a href="#">View in Issue Navigator</a>
Key	Summary	Issue Type	Priority	Status	Story Points (8.5)	
BA22-16	Install all programs needed	Story	Medium	<span style="background-color: green; border: 1px solid black; padding: 2px;">DONE</span>	6	
BA22-6	BA Bericht in Word aufsetzen	Story	Medium	<span style="background-color: green; border: 1px solid black; padding: 2px;">DONE</span>	1	
BA22-8	Git-Repository aufsetzen	Story	Medium	<span style="background-color: green; border: 1px solid black; padding: 2px;">DONE</span>	0.5	
BA22-12	Unity Projekt aufsetzen	Story	Medium	<span style="background-color: green; border: 1px solid black; padding: 2px;">DONE</span>	0.5	
BA22-9	Repository zu knaa transferieren	Story	Medium	<span style="background-color: green; border: 1px solid black; padding: 2px;">DONE</span>	0.5	

Abbildung 97: Realisierte Stories der Infrastruktur

## d. Entwicklung Prototype

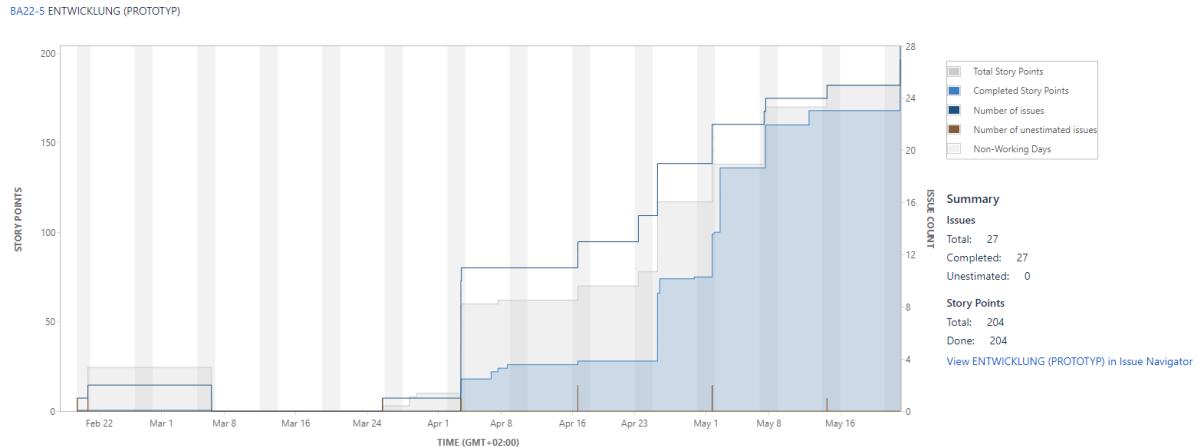


Abbildung 98: Prototypingphase in Jira

Status Report							
Completed Issues		Issues				View in Issue Navigator	
Key	Summary	Issue Type	Priority	Status	Story Points (204)		
BA22-52	Create first VR Scene	Story	Medium	DONE	10		
BA22-58	Headset bewegen	Story	Medium	DONE	1		
BA22-59	Controller bewegen	Story	Medium	DONE	4		
BA22-60	In Raum bewegen	Story	Medium	DONE	2		
BA22-61	Gatter hantieren	Story	Medium	DONE	1		
BA22-62	Gatter platzieren	Story	Medium	DONE	16		
BA22-63	Gatter entfernen	Story	Medium	DONE	6		
BA22-64	Kabe hantieren	Story	Medium	DONE	2		
BA22-65	Kabe platzieren	Story	Medium	DONE	16		
BA22-66	Kabe entfernen	Story	Medium	DONE	2		
BA22-133	Für Teleportation - Feinheit	Story	Medium	DONE	2		
BA22-57	Sandbox starten	Story	Medium	DONE	2		
BA22-148	Recycling	Story	Medium	DONE	20		
BA22-148	Hints	Story	Medium	DONE	12		
BA22-145	Unsnap Cable from Gatter when removed from Board	Story	Medium	DONE	8		
BA22-142	Erstellen der Didaktischen Aufgaben	Story	Medium	DONE	24		
BA22-67	Strom einschalten	Story	Medium	DONE	4		
BA22-81	Aufgabe ablesen	Story	Medium	DONE	8		
BA22-79	Aufgabe lösen	Story	Medium	DONE	24		
BA22-68	Strom ausschalten	Story	Medium	DONE	4		
BA22-69	Stromfluss einbinden	Story	Medium	DONE	4		
BA22-70	Stromfluss austrennen	Story	Medium	DONE	4		
BA22-55	Spiel starten	Story	Medium	DONE	6		
BA22-82	Raum zurücksetzen	Story	Medium	DONE	1		
BA22-56	Tutorial starten	Story	Medium	DONE	1		
BA22-78	Gatter-Dispenser betätigen	Story	Medium	DONE	16		
BA22-80	Gatter im Gatter-Dispenser auswählen	Story	Medium	DONE	4		

Abbildung 99: Realisierte Stories in der Prototypingphase

## e. Entwicklung Feinschliff

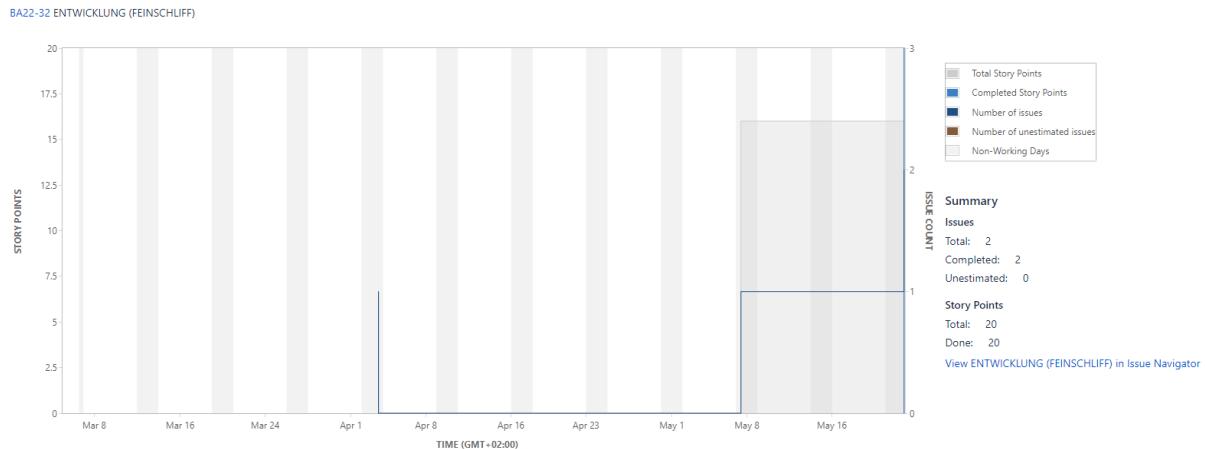


Abbildung 100: Feinschliffphase in Jira

Status Report						
Completed Issues				View in Issue Navigator		
Key	Summary	Issue Type	Priority	Status	Story Points (20)	
BA22-143	Refactor - Feinschliff an Unity Szene	Story	Medium	DONE	16	
BA22-144	Fix Cable Lag	Story	Medium	DONE	4	

Abbildung 101: Realisierte Stories in der Feinschliffphase

## f. Dokumentation

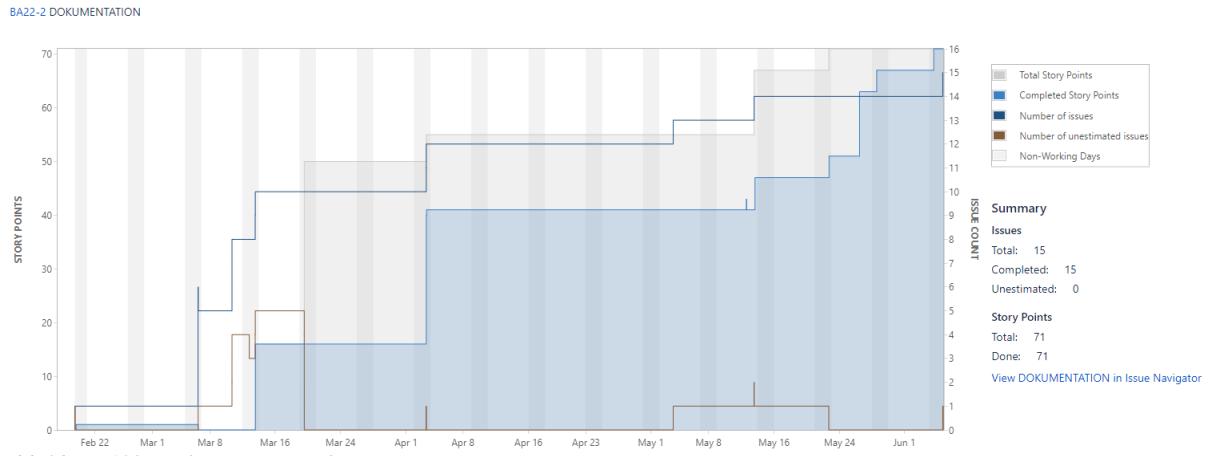


Abbildung 102: Dokumentationsphase in Jira

### Status Report

Completed Issues						View in Issue Navigator	
Key	Summary	Issue Type	Priority	Status	Story Points (71)		
BA22-48	Dokumentation der Use-Cases und funktionalen / nicht-funktionalen Anforderungen	Story	Medium	DONE	24		
BA22-39	Dokumentation Controller vs. Handtracking	Story	Medium	DONE	2		
BA22-40	Dokumentation Spielsprache / Mehrsprachigkeit	Story	Medium	DONE	2		
BA22-38	Dokumentation Single-Player	Story	Medium	DONE	2		
BA22-54	Copy Usecases into Jira	Story	Medium	DONE	1		
BA22-53	Übertragung Use Cases in Bericht, wenn Usecases implementiert	Story	Medium	DONE	4		
BA22-147	Diagram für Architektur	Story	Medium	DONE	12		
BA22-46	Dokumentation der Unterschiede zu den existierenden Lösungen	Story	Medium	DONE	4		
BA22-140	Quellenangaben zu Assets und Sounds	Story	Medium	DONE	4		
BA22-149	Dokumentation	Story	Medium	DONE	-		
BA22-25	Dokumentation der Spielidee	Story	Medium	DONE	16		
BA22-28	Dokumentation der Existierenden Lösungen	Task	Medium	DONE	-		
BA22-29	Dokumentation der Technologie (VR, Oculus)	Task	Medium	DONE	-		
BA22-31	Dokumentation des Projekt Managements	Task	Medium	DONE	-		
BA22-30	Überfliegen der verfügbaren BAs	Task	Medium	DONE	-		

Abbildung 103: Realisierte Stories in der Dokumentationsphase