



universität  
wien

# DIPLOMARBEIT / DIPLOMA THESIS

Titel der Diplomarbeit / Title of the Diploma Thesis

„Kompetenzen vorprogrammiert? –  
Evaluation von Computerspielprojekten mit Unity an  
höheren Schulen mithilfe interviewbasierter Codereviews“

verfasst von / submitted by

David Unterweger

angestrebter akademischer Grad / in partial fulfilment of the requirements for the degree of

Magister der Naturwissenschaften (Mag. rer. nat.)

Wien, 2020 / Vienna 2020

Studienkennzahl lt. Studienblatt /  
degree programme code as it appears on  
the student record sheet:

UA 190 406 884

Studienrichtung lt. Studienblatt /  
degree programme as it appears on  
the student record sheet:

Lehramtsstudium UniStG  
UF Mathematik UniStG  
UF Informatik und Informatikmanagement UniStG

Betreut von / Supervisor:

Univ.-Prof. Dipl.-Ing. Dr. Renate Motschnig



## Abstract

Im Rahmen des Projekts *Learn to proGrAME* erstellten Schüler\*innen dreier Klassen an zwei allgemeinbildenden, höheren Schulen Computerspiele in freien Projektarbeiten mithilfe des Game Development Tools *Unity*. Ziel dieser Arbeit ist, zu ergründen, welche Kompetenzen diese im Rahmen dieser Projektarbeit erwarben. Die Forschung wird nach dem Modell der *Participatory Action Research* durchgeführt. Basierend auf zwei bestehenden Kompetenzmodellen wird ein Kompetenzmodell für die Objektorientierte Spielentwicklung erarbeitet. Um die individuell erworbenen Kompetenzen der Schüler\*innen möglichst genau zu erfassen, wird ein eigenes Werkzeug – das *Interviewbasierte Codereview* – entwickelt, welches auf der Qualitativen Inhaltsanalyse nach Mayring basiert. Mithilfe der Codereviews werden fünf Projektarbeiten analysiert und es wird gezeigt, dass durch die Spielentwicklung ein großer Bereich an Kompetenzen abgedeckt werden kann, dies aber nicht notwendigerweise der Fall ist. Die geringe Stichprobengröße lässt eine Verallgemeinerung dieser Ergebnisse jedoch nicht zu. Allerdings werden dennoch wertvolle Erkenntnisse über den Kompetenzerwerb durch die Entwicklung von Computerspielen im Unterricht erlangt.

In the context of the project *Learn to proGrAME* students of general high schools (AHS) created video games with the game development tool *Unity* within the setting of an open project. The goal of this thesis is to find out which competencies those students acquire in said setting. The research is based on the *Participatory Action Research* model. Based on two existing competency models a model specific to object-oriented game development is created. In order to apprehend the individual learning outcomes of the students, a tool – the *Interview-Based Code Review* – is created, which is based on Mayring's Qualitative Content Analysis. With this tool five projects get analyzed and it is shown that game development has the potential to cover a broad field of competencies, but not necessarily does so. However, because of the small sample size a generalization of the results is not possible. Yet some important findings about the acquisition of competencies in the context of game development can be derived.

## Eidesstattliche Erklärung

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbstständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht.

Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

Wien, am 18.02.2020



David Unterweger

## **Danksagung**

Ohne die Unterstützung vieler wertvoller Menschen wäre diese Arbeit nicht zustande gekommen. Vielen Dank Renate und Oswald für die großartige Betreuung, insbesondere für das viele zeitnahe und konstruktive Feedback. Meinen Eltern möchte ich danken, dass sie mich über die Jahre hinweg immer unterstützten, an mich glaubten und mir ein finanziell sorgenfreies Studium ermöglichten.

Ganz besonderer Dank gilt Anna Maria und Viki für das Lektorieren dieser Arbeit.

Dank gebührt zu guter Letzt auch Klaus Teuber und dem Känguru, die es mir ermöglichten nach diversen Arbeitsmarathons den rauchenden Kopf zu entlüften sowie all den Mitstudierenden, mit denen ich in den letzten Jahren zusammenarbeitete und die mich durch mein Studium begleiteten.

# Inhaltsverzeichnis

<b>1. Einleitung.....</b>	<b>9</b>
<b>1.1. Das Projekt Learn to ProGrAME</b>	<b>9</b>
1.2. <i>Forschungsfragen</i> .....	10
1.3. <i>Vorgehen und Methodik</i> .....	11
1.3.1. Forschungssetting .....	11
1.3.2. Participatory Action Research (PAR) .....	12
1.3.3. Qualitative Inhaltsanalyse.....	12
1.3.4. Ablauf.....	14
<b>2. Game Development in der IT-Education .....</b>	<b>16</b>
2.1. <i>Einbettung von Game Development im Lehrplan</i> .....	16
2.2. <i>Unity in der IT-Education</i> .....	18
<b>3. Digitale Kompetenzen.....</b>	<b>20</b>
3.1. <i>DigiKomp12</i> .....	20
3.2. <i>Kompetenzmodell zur Objektorientierten Programmierung</i> .....	21
3.3. <i>Das Kompetenzmodell zur Objektorientierten Spielentwicklung</i> .....	23
<b>4. Interviewbasierte Code Reviews .....</b>	<b>25</b>
4.1. <i>Überblick</i> .....	25
4.2. <i>Vom Kompetenzmodell zum Interviewleitfaden</i> .....	26
4.3. <i>Ablauf der Analyse</i> .....	29
4.3.1. Strukturierende Inhaltsanalyse.....	29
4.3.2. Kategorienbildung und Durchführung .....	30
<b>5. Ergebnisse.....</b>	<b>32</b>
5.1. <i>Beschreibung der Projekte</i> .....	32
5.2. <i>Codeabdeckung</i> .....	33
5.3. <i>Analyse der kodierten Segmente</i> .....	35
5.3.1. 1a Datenstrukturen und Operatoren .....	35
5.3.2. 1b Kontrollstrukturen .....	38
5.3.3. 1c Objekte und Klassen .....	39
5.3.4. 1d Zustandsbasierte Automaten .....	40

5.3.5.	2bc Code Stil und Dateiorganisation .....	41
5.3.6.	3a Problemlösungsorientierung .....	42
5.3.7.	3b Interpretieren von Code .....	43
5.3.8.	4a Team Work .....	44
5.3.9.	4b Selbstbestimmtes Arbeiten .....	45
5.3.10.	5a Erstellung und Verarbeitung Digitaler Medien .....	46
5.3.11.	5b Copyright & Lizenzierung.....	47
5.3.12.	Unityspezifisches Wissen und Kompetenzen .....	48
<b>6.</b>	<b>Diskussion der Ergebnisse .....</b>	<b>50</b>
6.1.	<i>Folgerungen.....</i>	<i>50</i>
6.1.1.	Forschungsfrage F1.....	50
6.1.2.	Forschungsfrage F2.....	52
6.2.	<i>Einschränkungen der Studie .....</i>	<i>54</i>
6.3.	<i>Zukunftsausblick.....</i>	<i>54</i>
<b>7.</b>	<b>Literaturverzeichnis .....</b>	<b>56</b>
<b>8.</b>	<b>Tabellen .....</b>	<b>58</b>
<b>9.</b>	<b>Abbildungen.....</b>	<b>59</b>
<b>10.</b>	<b>Anhang .....</b>	<b>60</b>
A.	<i>Kodierleitfaden: .....</i>	<i>60</i>
B.	<i>Kategorien zur Inhaltsanalyse.....</i>	<i>61</i>
C.	<i>Transkriptionsrichtlinien.....</i>	<i>64</i>
D.	<i>Kodierte Interviews .....</i>	<i>65</i>



# 1. Einleitung

## 1.1. Das Projekt Learn to ProGrAME

Von August 2017 bis Oktober 2019 lief an der Universität Wien das Projekt Learn to proGrAME der Forschungsgruppe CSLEARN-Educational Technologies. Leitfrage des Projektes war „Unter welchen Bedingungen kann Computerspielentwicklung das Programmierenlernen verbessern?“ (Universität Wien, 2019).

Die Kernidee im Projekt war, durch enge Zusammenarbeit mit Schüler\*innen zweier Partnerschulen<sup>1</sup>, welche im Informatikunterricht ihr eigenes Programmierenlernen anhand der Entwicklung von Computerspielen erforschen, zu Erkenntnissen zu gelangen, die für Forschende, welche bereits über ein breites Spektrum an Programmierkompetenzen verfügen oder gar nicht programmieren lernen wollen, verborgen blieben. Das heißt, die Schüler\*innen nehmen nicht nur die Rolle der Lernenden und Forschungsobjekte ein, sondern gestalten den Forschungsprozess aktiv mit. Hierfür arbeiteten die lernenden Schüler\*innen der neunten Schulstufe gemeinsam mit Lehrkräften, Wissenschaftler\*innen sowie Schüler\*innen höherer Schulstufen der Wahlpflichtfächer Psychologie/Philosophie und Informatik zusammen. (vgl. Universität Wien, 2019)

Das Programmierenlernen selbst ist für viele Menschen mit großen Schwierigkeiten verbunden.

*Nicht-ProgrammiererInnen erscheint Programmieren oft wie Magie: seltsame Symbole, unverständliche Konstrukte und mysteriöse Zusammenhänge führen für Nicht-Eingeweihte auf undurchschaubare Weise zu erwarteten und manchmal auch unerwarteten Ergebnissen. (Comber, MC learn to proGrAME - Vorwort und Motivation, 2019)*

Um diesen Hürden entgegenzuwirken, wurde das Programmierenlernen mit Computerspielen in Verbindung gesetzt, da Computerspiele unter Jugendlichen sehr beliebt sind und die Aussicht, solche selbst entwickeln zu können, von vielen als sehr motivierend empfunden wird (Comber, 2016, S. 94).

Im Zuge des Projekts wurden in verschiedenen Forschungsarbeiten (darunter viele Bachelor- und Diplomarbeiten sowie eine Dissertation) zahlreiche Daten, unter anderem in Form von Fragebögen, Interviews, Lerntagebüchern und IT-Skills-Tests, erhoben und ausgewertet. Außerdem wurden Unterrichtsmaterialien in Form eines Moodlekurses sowie text- und videobasierten Tutorials erarbeitet. Diese wurden unter laufender Anwendung im Unterricht verbessert und weiterentwickelt. Alle im Zuge des Projekts erarbeiteten Unterrichtsmaterialien

---

<sup>1</sup> Gymnasium und Realgymnasium Wien 16 - Maroltingergasse & Evangelisches Realgymnasium Donaustadt

sind online und frei verfügbar<sup>2</sup>. Die Tutorials umfassen ein gut strukturiertes Grundlagentutorial über die Basiskonzepte des Programmierens in C# und Unity, ein detailliertes Tutorial zur Erstellung des Spiels *Bouncyfant* und Tutorials zur Modifizierung des speziell für das Projekt entwickelten Mobile-Games *Space Asteroids*.

## 1.2. Forschungsfragen

Um ein Computerspiel zu erstellen, benötigt man Wissen und Kompetenzen in einer Vielzahl verschiedener Bereiche. Für ein Spiel braucht man zum Beispiel unter anderem Grafiken, 3D-Modelle, Texturen, Soundeffekte, Musik, ein Narrativ, Spielmechaniken und natürlich den Programmcode, der all das in ein, auf einem Computer ausführbares, Programm packt. Zusätzlich zu dem für diese Aspekte nötigem kreativen und technischen Knowhow, benötigen die Schaffenden noch mehr. Denn, soll ein Spiel auch richtig gut werden, muss man sich zusätzlich Gedanken darüber machen, wie man Spieler\*innen überhaupt dazu motiviert in ein Spiel einzutauchen und dessen mitunter komplexe Spielmechaniken und Regeln zu lernen und zu meistern. Die Schaffenden nehmen hier also auch eine Rolle ähnlich der einer Regisseurin oder eines Regisseurs ein.

Natürlich werden nicht alle der hier genannten Bereiche benötigt, um ein simples Computerspiel im Rahmen des Unterrichts zu erstellen. In eigenen persönlichen Hobbyprojekten und durch die Arbeit im Projekt Learn to proGrAME zeigte sich mir jedoch, dass das breite Spektrum an benötigten Kompetenzen auch in kleinen Projekten eine relevante Rolle spielt.

Aufgrund dieser Herausforderung stellte sich mir die Frage, welchen konkreten Nutzen Computerspielentwicklung im Unterricht hat und wie sich diese Komplexität des Themas auf das Arbeiten und Lernen im Unterricht auswirkt. Aus diesen Überlegungen resultiert die folgende Forschungsfrage:

*F1: Welche Fertigkeiten und digitalen Kompetenzen werden durch die Entwicklung von Computerspielen in Unity im Informatikunterricht gefördert?*

Es liegt auf der Hand, dass Computerspielentwicklung ein breites Spektrum an Kompetenzen abdeckt. Interessant ist vor allem, welche Kompetenzbereiche stärker und welche eventuell nicht oder fast nicht gefördert werden. Es liegt außerdem nahe, dass nicht alle betroffenen Kompetenzbereiche gleichermaßen gefördert werden und das Maß des Lernfortschritts in den verschiedenen Bereichen stark individuell variiert. Weiters lässt sich vermuten, dass

---

<sup>2</sup><https://www4.lernplattform.schule.at/g16slsz/course/view.php?id=183>  
<https://learn2progame.github.io/learn2proGrAME-Tutorial/>  
<https://www.youtube.com/channel/UC4e5WvpfeHccA4HSPpt2Njg>

Schüler\*innen sich mehr mit Aspekten des Spiels beschäftigen, die ihnen für das Spiel besonders wichtig erscheinen, ihnen leichter fallen oder sie besonders interessieren. Das kann beispielsweise bedeuten, dass jemand der oder die gerne zeichnet, viel Arbeitszeit damit verbringt, Spielfiguren zu zeichnen, was wiederum vielleicht weniger Lernerfolg bei Programmierkompetenzen mit sich zieht. Dieser Effekt dürfte durch das Arbeiten in Teams mit konkreter Arbeitsteilung begünstigt werden. Die Vermutung ist, dass sich Schüler\*innen eher intensiv mit bestimmten Teilbereichen der Spielentwicklung auseinandersetzen, als sich gleichermaßen mit all den nötigen Arbeiten zu beschäftigen, sofern ihnen die dazu nötigen Freiräume gewährt werden.

Aus diesen Überlegungen resultieren folgende Hypothesen:

*H1: Durch die Entwicklung von Computerspielen kann ein großer Teil der im Lehrplan geforderten Kompetenzbereiche abgedeckt werden.*

*H2: Der tatsächliche Lernerfolg in bestimmten Kompetenzbereichen variiert stark zwischen einzelnen Schüler\*innen.*

*H3: Schüler\*innen setzen sich bei Arbeiten an einem Computerspielprojekt stärker mit Themen auseinander, auf die einer der folgenden Punkte zutrifft:*

*H3a: Das Thema fällt ihnen nicht schwer.*

*H3b: Sie interessieren sich stark für das Thema.*

*H3c: Das Thema ist für das Spiel sehr wichtig.*

Um die Forschungsfrage F1 zu beantworten und die Hypothesen zu überprüfen, wurde ein eigenes Werkzeug – die Interviewbasierten Codereviews – entwickelt. Dazu ergibt sich die zweite Forschungsfrage:

*F2: Inwieweit sind Interviewbasierte Codereviews als Mittel zur Kompetenzmessung geeignet?*

## 1.3. Vorgehen und Methodik

### 1.3.1. Forschungssetting

Im Zuge dieser Arbeit wurde der Informatikunterricht zweier Klassen<sup>3</sup> des GRG16 – Maroltingergasse Wien innerhalb eines Schuljahres begleitet. In beiden Klassen erstellten die Schüler\*innen zunächst unter Anleitung des unterrichtenden Lehrers, sowie mithilfe speziell angefertigter Unterrichtsmaterialien ein vorgegebenes Spiel. Im Anschluss arbeiteten sie allein oder in Gruppen von bis zu drei Personen entweder an einer Modifizierung des Spiels oder an

---

<sup>3</sup> 5. Klasse (9. Schulstufe) im dritten Jahr Informatik & Wahlpflichtfach Informatik der 7. Klassen (11. Schulstufe) einer AHS

einem komplett eigenen Projekt. Parallel zu den begleiteten Klassen wurde ein ähnlicher Unterricht auch am ERG Donaustadt, Wien 22 durchgeführt<sup>4</sup>. Für die empirische Forschungsarbeit wurde dann mit Schüler\*innen aus diesen drei Klassen zusammengearbeitet.

### 1.3.2. Participatory Action Research (PAR)

Die Durchführung der Forschung wurde nach dem Modell der Participatory Action Research (Baum, MacDougall, & Smith, 2006) durchgeführt. Dementsprechend wurde nicht versucht, die Schüler\*innen bzw. deren Lernerfahrungen unter „Laborbedingungen von außen“ zu beobachten. Stattdessen nahm ich aktiv am Unterricht der beiden Klassen am GRG16 teil. Dafür arbeitete ich eng mit deren Informatiklehrer zusammen, indem ich den Unterricht mitgestaltete, beobachtete und die Schüler\*innen bei ihrem Arbeiten unterstützte. Dies ermöglichte es, individuelle Lernerfahrungen nicht isoliert, sondern in ihrem jeweiligen Kontext zu erfassen und basierend darauf, weitere Schritte zu planen. Dieser dynamische Ansatz ist einer der drei wesentlichen Unterscheidungsmerkmale der PAR zu einem klassisch positivistisch geprägten Forschungsparadigma. Ein weiterer wichtiger Punkt von PAR ist, dass die Forschungssubjekte aktiv in den Prozess involviert werden und diesen ebenfalls mitgestalten, wodurch diese im Forschungsprozess die Rolle von Partner\*innen einnehmen. (vgl. Baum, MacDougall, & Smith, 2006)

### 1.3.3. Qualitative Inhaltsanalyse

Einen wesentlichen Teil der in Kapitel 4 beschriebenen Interviewbasierten Code Reviews bildet die Qualitative Inhaltsanalyse nach Mayring (Mayring, 2014). Qualitative Inhaltsanalyse ist eine Auswertungsmethode für Datenerhebungen (meist in Form von Texten) in sozialwissenschaftlichen Forschungsprojekten. Sie bildet ein Werkzeug mit dem auch große Mengen an Daten analysiert werden können. Der erste Analyseschritt ist hierbei immer ein qualitativ-interpretativer Akt, wodurch auch die Erfassung verdeckter Inhalte ermöglicht wird. Bei der Analyse geht man allerdings streng regelgeleitet vor, was eine hohe intersubjektive Überprüfbarkeit bedingt. (vgl. Mayring & Fenzl, 2014, S. 543)

Anders als der Begriff Qualitative Inhaltsanalyse vermuten lässt, handelt es sich jedoch keineswegs um ein rein qualitatives Verfahren. Qualitative Inhaltsanalyse lässt sich in die sogenannten Mixed-Methods-Ansätze einordnen, da sie auch eine generalisierend-quantitative Vorgehensweise erlaubt. (vgl. Mayring & Fenzl, 2014, S. 551)

Den Kern bildet die Kategorisierung von einzelnen Textstellen anhand eines speziell erstellten Kategoriensystems, was die Qualitative Inhaltsanalyse von anderen gängigen

---

<sup>4</sup> 5. Klasse (9. Schulstufe) AHS

Textanalyseansätzen unterscheidet. (vgl. Mayring & Fenzl, 2014, S. 544) Die Erstellung dieser Kategorien kann entweder induktiv am Material oder vorab deduktiv erfolgen. Wesentlich ist, dass dieser Vorgang aber nicht einmal ausgeführt wird, sondern sich zirkulär wiederholt, wodurch die Regeln angepasst und verfeinert werden können. Im endgültigen Materialdurchgang müssen die Regeln jedoch unverändert bleiben.

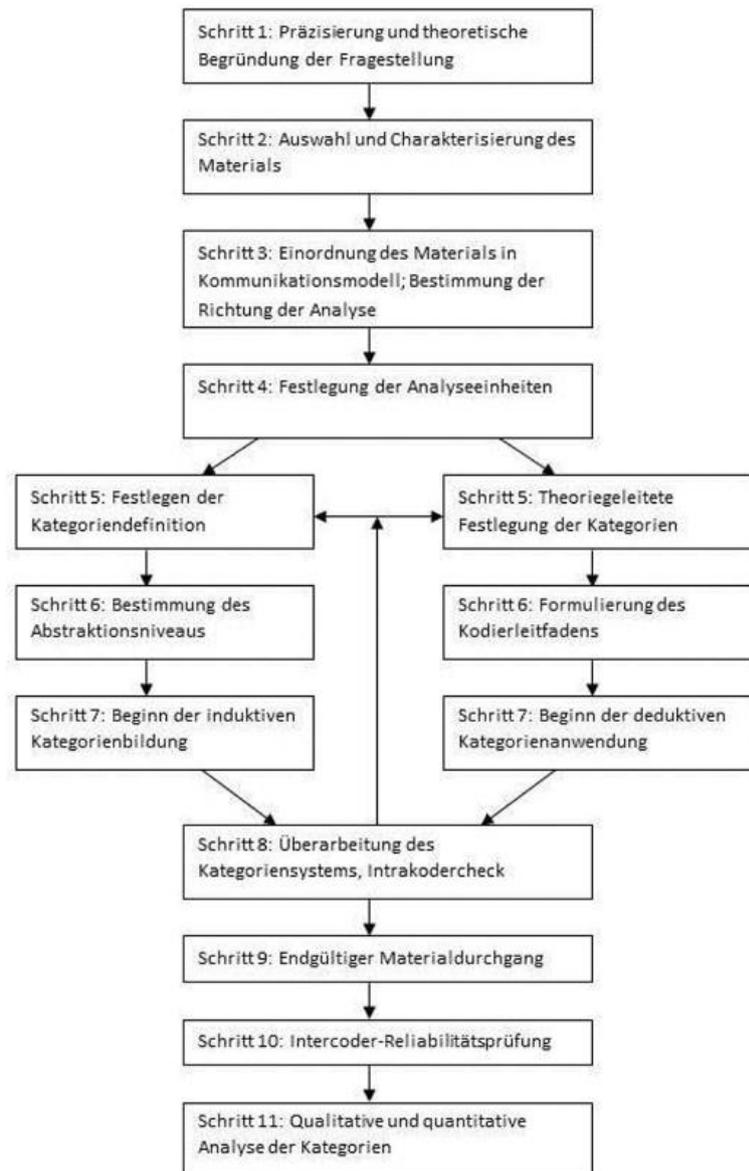


Abbildung 1 Ablauf Induktiver Kategorienbildung und Deduktiver Kategorienanwendung (Mayring und Brunner 2006 zitiert nach Mayring und Fenzl, 2014, S. 550)

Um ein hohes Maß an Objektivität sicherzustellen, sollte die Kodierung von einer zweiten Person wiederholt werden, wobei das Ziel nicht eine vollständige Übereinstimmung sein muss. (vgl. Mayring & Fenzl, 2014, S. 546f) Da die Durchführung immer interpretative Elemente hat, wird ein hohes Maß an Übereinstimmung nur schwer erreicht werden. Hierfür nennt Mayring drei Stufen der Intercoder-Überprüfung. Stufe 1 bildet die strengste Überprüfung. Hier wird der

zweiten Person lediglich das zu analysierende Material gegeben. In Stufe 2 werden, neben dem zu untersuchenden Material, auch Kodierleitfaden und Kategoriensystem zur deduktiven Kategorieanwendung weitergegeben. In der dritten Stufe erhält die zweite Person das fertig kategorisierte Material, das Kategoriensystem sowie den Kodierleitfaden und überprüft, ob sie die Kodierung bestätigen kann. Je nach Art des Forschungsprojektes sollte man sich für eine dieser drei Stufen entscheiden. Gibt es nach der Überprüfung Diskrepanzen, sollten diese diskutiert werden, um sich auf eine „richtige“ Kodierung festzulegen. (vgl. Mayring, 2014, S. 114)

#### 1.3.4. Ablauf

Wie oben erwähnt, ist Spieleentwicklung mit Unity ein größeres Thema im Informatikunterricht der betrachteten Schüler\*innen im Schuljahr 2018/2019. In allen drei Klassen wurden zuvor gesondert die Grundlagen des Programmierens mit C# in Visual Studio behandelt. Der Unterricht zur Spieleentwicklung fand dann in zwei Phasen statt. In der ersten erstellten die Schüler\*innen, wie schon erwähnt, ein Spiel nach Anleitung und in der zweiten Phase arbeiteten sie an einem eigenen Projekt. Während dieser Phase erstellte ich im Unterricht stichwortartige Notizen zu individuellen Lernerfahrungen und auftretenden Problemen der Schüler\*innen. Anhand dieser Notizen und der Erfahrungen im Unterricht wurde dann im Sinne des Participatory Action Research (Baum, MacDougall, & Smith, 2006) das weitere Vorgehen geplant. In Folge wurden am Ende des Schuljahres fünf Projekte der Schüler\*innen im Hinblick auf die Forschungsfrage F1 analysiert.

Für die Analyse wurden fünf Projekte herangezogen. Diese geringe Anzahl hat mehrere Gründe. Einerseits war der Zeitplan in allen Klassen sehr eng gesteckt, sodass nicht alle ein zumindest annähernd fertiges Projekt abliefern konnten. Hinzu kam, dass die Anzahl der Projekte selbst sehr gering war, denn den Schüler\*innen der Klasse des ERG – Donaustadt stand frei, ob sie sich in ihrer Projektarbeit der Spielentwicklung oder einem anderen Thema widmen und die 7. Klasse des Wahlpflichtfachs Informatik am GRG16 bestand aus lediglich fünf Schülern. Diese sehr kleine Menge an Projekten schränkt die Validität der im Zuge der Analyse erfassten Ergebnisse stark ein. Trotzdem können wertvolle Erkenntnisse gewonnen werden, welche dann in mögliche weitere Iterationen im Sinn der Participatory Action Research (Baum, MacDougall, & Smith, 2006) einfließen können.

Für die Analyse, welche den zentralen empirischen Teil dieser Forschungsarbeit darstellt, wurde ein eigenes Instrument, basierend auf der Qualitativen Inhaltsanalyse nach Mayring (Mayring, 2014), entwickelt. Die Wahl für eine qualitative Methode hat mehrere Gründe. Einerseits hätte eine quantitative Forschungsmethode in diesem Setting, aufgrund der geringen Anzahl an Schüler\*innen, welche im Rahmen des Unterrichts an eigenen Projekten arbeiteten, wenig

Aussagekraft. Andererseits arbeitete der Kollege Michael Molnar im Rahmen seiner Diplomarbeit an einem computergestützten, fragenbasierten IT-Skills-Test. Mit diesem Test wurden Schüler\*innen, die im Rahmen von Learn to proGrAME Spielentwicklung betrieben, mit Schüler\*innen, welche programmieren in einem herkömmlichen Setting ohne Spielentwicklung lernten, verglichen (Molnar, 2020).

Eine direkte Analyse der Projektabgaben würde nur insofern Sinn machen, als dass man davon ausgeht, dass Inhalte in dem abgegebenen Material zum einen von den Schüler\*innen eigenständig erstellt sind und zum anderen auch tatsächlich verstanden wurden. Gerade beim Programmieren kommt es aber häufig vor, dass Lösungen durch Trial & Error, Kopieren/Nachahmen von Lösungen anderer, bzw. einer Kombination dessen entstehen. Um in diesem Bezug keine falschen Schlüsse zu ziehen, wurden für die Analyse der Projektarbeiten die Interviewbasierten Codereviews entwickelt, welche eine Analyse des abgegebenen Materials mit einem Gespräch mit dem Projektteam verbinden. Die Interviewbasierten Codereviews werden in Kapitel 4 im Detail erklärt. Die Eignung der Interviewbasierten Codereviews zur Kompetenzmessung<sup>5</sup> wird im Anschluss an deren Anwendung, anhand einer Reflexion über die erhaltenen Resultate, welche in Bezug auf die Erfahrungen im Unterricht mit den einzelnen Schüler\*innen kontextualisiert werden, evaluiert.

---

<sup>5</sup> Forschungsfrage F2

## 2. Game Development in der IT-Education

### 2.1. Einbettung von Game Development im Lehrplan

Im Lehrplan Informatik für die fünfte Klasse AHS (die einzige Schulstufe mit Informatik als Pflichtfach) in der momentan gültigen Fassung<sup>6</sup> wird die Entwicklung von Computerspielen wörtlich nicht erwähnt. Dies bedeutet aber keineswegs, dass diese keine Daseinsberechtigung im Unterricht hat. Die Entwicklung von Computerspielen kann eine Vielzahl von Tätigkeitsbereichen in der Informatik abdecken. Die im Unterricht zu behandelnden Themengebiete sind Informatik, Mensch und Gesellschaft, Informatiksysteme, Angewandte Informatik und Praktische Informatik. Im Folgenden wird kurz erörtert, inwiefern Computerspielentwicklung mit jedem dieser Bereiche Berührungspunkte aufweist.

#### *Informatik, Mensch und Gesellschaft<sup>6</sup>*

- *Die Bedeutung von Informatik in der Gesellschaft beschreiben, die Auswirkungen auf die Einzelnen und die Gesellschaft einschätzen und Vor- und Nachteile an konkreten Beispielen abwägen können*
- *Maßnahmen und rechtliche Grundlagen im Zusammenhang mit Datensicherheit, Datenschutz und Urheberrecht kennen und anwenden können*
- *Die Entwicklung der Informatik beschreiben und bewerten können*
- *Informatikberufe und Einsatzmöglichkeiten der Informatik in verschiedenen Berufsfeldern benennen und einschätzen können*

Computerspiele stellen einen wesentlichen Bezug von Jugendlichen zur Welt der Informatik dar. Die Beschäftigung mit Computerspielen stellt hier also einen natürlichen Anknüpfungspunkt an die Lebenswelt der Jugendlichen dar, der viel Raum bietet, über die Bedeutung von Spielen in unserer Gesellschaft und deren Auswirkungen zu reflektieren. Für die Erstellung eines Computerspiels benötigt man Grafiken, 3D-Modelle, Soundeffekte, Musik etc. Sofern man diese nicht zur Gänze selbst erstellt, sondern von Drittquellen bezieht, bietet dies eine gute Gelegenheit mit direktem Praxisbezug das Thema Urheberrecht im Unterricht zu behandeln.

#### *Informatiksysteme<sup>6</sup>*

- *Den Aufbau von digitalen Endgeräten beschreiben und erklären können*
- *Die Funktionsweise von Informatiksystemen erklären können*
- *Grundlagen von Betriebssystemen erklären, eine graphische Oberfläche und Dienstprogramme bedienen können*
- *Grundlagen der Vernetzung von Computern beschreiben und lokale und globale Computernetzwerke nutzen können*

---

<sup>6</sup> BGBl. II Nr. 395/2019 Artikel IV Anlage A 5. Teil Informatik (Aufgerufen am 06.02.2020)

Der Bereich Informatiksysteme kann im Grunde zur Gänze abgedeckt werden. Um Spiele für digitale Endgeräte zu erstellen, muss man sich zwangsweise mit deren Aufbau und Funktionsweise beschäftigen. Man könnte sogar Spiele erstellen, welche über Computernetzwerke spielbar sind, wodurch sich die Schüler\*innen mit der Funktionsweise von ebendiesen auseinandersetzen müssten.

#### *Angewandte Informatik<sup>6</sup>*

- *Standardsoftware zur Kommunikation und Dokumentation sowie zur Erstellung, Publikation und multimedialen Präsentation eigener Arbeiten einsetzen können*
- *Standardsoftware für Kalkulationen und zum Visualisieren anwenden können*
- *Informationsquellen erschließen, Inhalte systematisieren, strukturieren, bewerten, verarbeiten und unterschiedliche Informationsdarstellungen verwenden können*
- *Digitale Systeme zum Informationsaustausch, zur Unterstützung der Unterrichtsorganisation und zum Lernen auch in kommunikativen und kooperativen Formen verwenden können*

Die im Bereich Angewandte Informatik angeführten Aspekte weisen weniger direkten Bezug zur Spielentwicklung auf. Jedoch ist es auch hier möglich anzuknüpfen, wenn beispielsweise das Dokumentieren und Präsentieren der eigenen Arbeit, das eigenständige Erarbeiten und Recherchieren oder die Benutzung von digitalen Hilfsmitteln zur Teamarbeit in einem Projekt zum Gegenstand gemacht wird.

#### *Praktische Informatik<sup>6</sup>*

- *Begriffe und Konzepte der Informatik verstehen und Methoden und Arbeitsweisen anwenden können*
- *Algorithmen erklären, entwerfen, darstellen und in einer Programmiersprache implementieren können*
- *Grundprinzipien von Automaten, Algorithmen, Datenstrukturen und Programmen erklären können*
- *Datenbanken benutzen und einfache Datenmodelle entwerfen können*

Im Punkt Praktische Informatik weist Spielentwicklung einen direkten Bezug auf. In diesem Bereich finden sich Programmierkompetenzen, Algorithmen, Datenstrukturen, sowie die Anwendung von Methoden und Arbeitsweisen der Informatik.

In den didaktischen Grundsätzen des Lehrplans wird explizit darauf hingewiesen, dass dieser viel Freiraum in der Wahl der Inhalte und deren konkreten Umsetzung lässt. Die Auswahl der Themen und Inhalte soll hierbei insbesondere unter Rücksicht auf die Vorkenntnisse, Interessen und individuellen Stärken der Schüler\*innen angepasst werden. Außerdem soll das gemeinschaftliche Problemlösen in einem projektorientierten Unterricht angeregt werden.<sup>6</sup> Im Rahmen des Unterrichts ist es möglich sehr unterschiedliche Fokussierungen auf verschiedene

Handlungsbereiche zu legen. Beispielsweise kann man den Fokus mehr auf das Zeichnen von Spielegrafiken legen und den Schüler\*innen schon vorgefertigten Programmcode geben. Genauso gut kann man ein Projekt auch umgekehrt angehen, indem man sämtliche Grafiken zur Verfügung stellt oder aus Sammlungen frei verfügbarer Grafiken heranzieht, sodass sich die Schüler\*innen mehr auf das Programmieren oder die Spielmechaniken fokussieren können. Aufgrund dieser vielfältigen Möglichkeiten eignet sich die Entwicklung von Computerspielen sehr gut den oben genannten didaktischen Grundsätzen gerecht zu werden.

Zusammenfassend lässt sich sagen, dass die Entwicklung von Computerspielen das Potenzial hat, viel zur Erreichung der im Lehrplan definierten Ziele beizutragen, wenngleich sie auf den ersten Blick nicht direkt in jenem zu finden ist. Inwiefern das eigenständige Arbeiten an Spielprojekten dem tatsächlich auch gerecht wird und welche Kompetenzen tatsächlich inwieweit gefördert werden, ist Gegenstand dieser Forschungsarbeit.

## 2.2. Unity in der IT-Education

Unity<sup>7</sup> ist eine professionelle Game Development Engine zur Erstellung von 2D und 3D Spielen. Mit Unity ist es möglich, eigene kleine sowie große Spiele zu erstellen, ohne eine eigene Engine programmieren zu müssen. Hierfür stellt Unity unter anderem auch eine komplette Physik-Engine für 3D und 2D Spiele zur Verfügung. Will man ein Spiel mit Unity erstellen, so muss man sich also nicht mit den technischen Details von Rendering, Kollisionsbehandlung etc. kümmern. Unity kann außerdem in der Personal Edition kostenlos verwendet werden. Im Gegensatz zu vielen anderen kostenlos verfügbaren Gameengines, wird Unity auch für die professionelle Entwicklung von Spielen genutzt. Beispielsweise wurden populäre Spiele wie *Hearthstone*, *Ori and the Blind Forest* oder *Cities: Skylines* mit Unity erstellt (vgl. Unity Technologies, 2020). Eine weitere weit verbreitete, professionelle und für den Privatgebrauch kostenlose Game Development Engine ist die Unreal Engine.<sup>8</sup>

Verwendung in der Schulbildung findet Unity häufig im Bereich *Game Based Learning*. Das heißt, dass versucht wird, mithilfe eines Spiels bestimmte Inhalte zu vermitteln. In diesem Feld gibt es zahlreiche Forschungsliteratur. Die Verwendung von Engines wie Unity im Schulunterricht zur Erstellung von Computerspielen ist hingegen noch eher unerschlossen. Allerdings wird Unity an Universitäten zur Lehre von Computerspielentwicklung verwendet. Beispielsweise wurde es in vergangenen Semestern an der Universität Wien in der Lehrveranstaltung Gaming Technologies benützt. Mittlerweile wird hier jedoch eine speziell angefertigte, hauseigene Engine verwendet.

---

<sup>7</sup> <https://unity.com/>

<sup>8</sup> <https://www.unrealengine.com/en-US/>

Eine weiteres Beispiel ist das College of Ithaca in den USA. Hier führte Paul Dickson Game Development Kurse für Studierende ohne Vorwissen in der Computerspielentwicklung ein. Diese Kurse wurden Semester für Semester überarbeitet. Dabei stellte sich heraus, dass die Verwendung von fertigen Game Development Engines für einen solchen Kurs von Vorteil ist. Wird beispielsweise nur ein einfaches Framework zur Erstellung von IOS-Apps verwendet, lernen die Studierenden vor allem mit diesem Framework umzugehen und überhaupt erst einmal eine Gameloop zu implementieren. Das Ergebnis war, dass viele Studierenden am Ende des Kurses kein fertiges Spiel hatten. (vgl. Dickson, 2015) Wenn man diesen Aspekt von der universitären Lehre auf den Unterricht in der Schule überträgt, liegt nahe, dass er sich noch stärker auswirkt. Außerdem nennt Dickson noch weitere Gründe dafür, Unity in einem solchen Kurs zu verwenden.

Unity ...

- ... ist kostenlos
- ... ist eine fertige Game Development Engine
- ... ist einfach zu erlernen
- ... erlaubt Entwicklung auf Windows, Mac und Linux
- ... erlaubt Export auf alle gängigen Plattformen (Web, Windows, Mac, Linux, IOS, Android, Xbox, ...)
- ... ermöglicht Erstellung von sowohl 2D als auch 3D Spielen
- ... ist ein professionelles Tool, welches auch für „echte“ Projekte eingesetzt werden kann.

(vgl. Dickson, 2015)

Alle oben genannten Punkte sollten ebenso für eine Verwendung von Unity im Schulunterricht sprechen. Ein weiterer Punkt, der ebenso für Unity spricht, ist die breite Verfügbarkeit von Tutorials und Lernmaterialien im Internet. Neben einer Vielzahl an Tutorials, welche von Unity Technologies selbst zur Verfügung gestellt werden, gibt es auch zahlreiche YouTube-Kanäle, die sich der Spielentwicklung mit Unity widmen. Dies ist im schulischen Kontext sowohl für Schüler\*innen als auch für Lehrkräfte in Informatik relevant, da auch diese sich mitunter erst in die Materie einarbeiten müssen, um einen entsprechenden Unterricht zu durchzuführen.

### 3. Digitale Kompetenzen

Lernerfolge werden in den letzten Jahren vermehrt in Form von Kompetenzen erfasst und gemessen (vgl. Berges, Striewe, Shah, Goedicke, & Hubwieser, 2016, S. 1). Für die Definition des Kompetenzbegriffs stütze ich mich in dieser Arbeit auf Franz E. Weinert, der Kompetenzen wie folgt beschreibt:

*Dabei versteht man unter Kompetenzen die bei Individuen verfügbaren oder durch sie erlernbaren kognitiven Fähigkeiten und Fertigkeiten, um bestimmte Probleme zu lösen, sowie die damit verbundenen motivationalen, volitionalen [die willentliche Steuerung von Handlungen und Handlungsabsichten] und sozialen Bereitschaften und Fähigkeiten, um die Problemlösungen in variablen Situationen erfolgreich und verantwortungsvoll nutzen zu können. (Weinert, 2001, S. 27f)*

Basis einer adäquaten Messung von Kompetenzen ist ein fundiertes Kompetenzmodell (vgl. Kramer, Hubwieser, & Brinda, 2016, S. 1). Bevor man also der Frage nachgehen kann, welche Kompetenzen in einem Kontext überhaupt gefördert werden, wird ein passendes Kompetenzmodell benötigt. Im folgenden Unterkapitel wird, ausgehend von zwei bereits etablierten Kompetenzmodellen, ein Modell erarbeitet, welches die für die Entwicklung von Computerspielen nötigen Kompetenzen erfassen soll.

#### 3.1. DigiKomp12

Ausgangspunkt dafür bildet das für die Schulinformatik in der Oberstufe naheliegende Kompetenzmodell DigiKomp12 des österreichischen Bundesministeriums für Bildung, Wissenschaft und Forschung. Dieses Kompetenzmodell ist in vier Kernbereiche unterteilt, die wiederum in je vier Unterkategorien aufgespalten sind. Diese sind wie folgt definiert:

##### **1. Informationstechnologie, Mensch und Gesellschaft**

- 1.1. Bedeutung von Informatik in der Gesellschaft
- 1.2. Verantwortung, Datenschutz und Datensicherheit
- 1.3. Geschichte der Informatik
- 1.4. Berufliche Perspektiven

##### **2. Informatiksysteme**

- 2.1. Technische Grundlagen und Funktionsweisen
- 2.2. Betriebssysteme und Software
- 2.3. Netzwerke
- 2.4. Mensch-Maschine-Schnittstelle

##### **3. Angewandte Informatik**

- 3.1. Produktion digitaler Medien
- 3.2. Kalkulationsmodelle und Visualisierung
- 3.3. Suche, Auswahl und Organisation von Information
- 3.4. Kommunikation und Kooperation

##### **4. Praktische Informatik**

- 4.1. Konzepte der Informationsverarbeitung

- 4.2. Algorithmen, Datenstrukturen und Programmierung
  - 4.3. Datenmodelle und Datenbanksysteme
  - 4.4. Intelligente Systeme
- (Bundes- und Koordinationszentrum eEducation Austria, kein Datum)

DigiKomp12 deckt hiermit ein sehr breites Spektrum der Informatik ab. Für die Entwicklung von Computerspielen im Rahmen einer Projektarbeit sind hier insbesondere der gesamte Bereich 3. *Angewandte Informatik* sowie 4.2 *Algorithmen, Datenstrukturen und Programmierung* relevant. Außerdem spielt in Bezug auf die Lizenzierung der verwendeten Software und Assets von Dritten auch 1.2 *Verantwortung, Datenschutz und Datensicherheit* eine Rolle. Im Detail werden in 4.2 die folgenden Fertigkeiten beschreiben:

*Ich kann den Algorithmusbegriff erklären.*

*Ich kann einfache Algorithmen nachvollziehen und erklären.*

*Ich kann die Umsetzung von Algorithmen mit einem Computer erklären.*

*Ich kann einfache Aufgaben mit Mitteln der Informatik modellieren.*

*Ich kann einfache Algorithmen entwerfen, diese formal darstellen, implementieren und testen.*

*Ich kann an Hand von einfachen Beispielen die Korrektheit von Programmen bewerten.*

*(Bundes- und Koordinationszentrum eEducation Austria, kein Datum)*

Das Kompetenzmodell erfasst Programmierkompetenzen allerdings nur sehr oberflächlich. Dabei spielen gerade diese eine zentrale Rolle in der Entwicklung von Computerspielen mit Unity.

### 3.2. Kompetenzmodell zur Objektorientierten Programmierung

Ein Kompetenzmodell speziell für die Einführung von Objektorientierter Programmierung auf einem Oberstufenniveau wurde 2016 von Kramer et al. ausgearbeitet (Kramer, Hubwieser, & Brinda, 2016, S. 1). Dieses Modell ist wie folgt aufgebaut:

#### 1. OOP knowledge and skills

- a. data structure (graph, tree, array)
- b. class & object structure (object, attribute, association)
- c. algorithmic structure (loops, conditional statement)
- d. notional machine (data, working memory, processor, statement, program, automaton)

#### 2. Mastering representation (language, syntax, semantics)

#### 3. Cognitive Process

- a. Problem solving stage (understanding the problem, determine how to solve the problem, translating the problem into a computer language program, testing and debugging the program)
- b. Cognitive Process Type
  - i. Interpreting (Remember, Understand, Analyze, Evaluate)
  - ii. Producing (Apply, Create)

#### 4. Metacognitive processes

(Kramer, Hubwieser, & Brinda, 2016, S. 5)

Es besteht aus vier Hauptdimensionen, welche weiter in Subdimensionen unterteilt werden. Jeder mögliche Kompetenzkandidat in diesem Modell deckt dann einen Teilbereich über die Menge dieser Dimensionen ab. Beispielsweise wird von der Kompetenz „kann Arrays in Java programmieren“ ein Bereich über die Dimensionen *1.1 Arrays, 2 Java, 3.1 Translating, 3.2.1 Understand, 3.2.2. Apply* abgedeckt. (vgl. Kramer, Hubwieser, & Brinda, 2016, S. 5)

OOP (Kramer et al., 2016, S5).		Learn to proGrAME (Learn to proGrAME Ressourcen)			
1. OOP knowledge & skills	a) data structure (graph, tree, array)	1. Object Oriented Programming	a) Data Structures & Operators	i. Variables ii. Data Types iii. Operators iv. Scope and Access Control	
	b) class & object structure (object, attribute, association)		b) Control Structures	i. Conditionals ii. Loops	
	c) Algorithmic structure (loops, conditional statement)		c) Objects	i. Functions ii. Classes	
	d) Notional machine (data, working memory, processor, statement, program, automaton)		d) Automata Theory	i. Object States (e.g. Animation States)	
2. Mastering representation (language, syntax, semantics)		2. Presentation & Coding Style	a) Meaningful Identifier Labels		
			b) Coding Conventions		
			c) File Organization		
3. Cognitive Process	a) Problem solving stage (understanding the problem, determine how to solve the problem, translating the problem into a computer language program, testing and debugging the program)		3. Cognitive Processes	a) Problem Solving Orientation	i. Use of Algorithms ii. Testing iii. Debugging
	b) Cognitive process type	i. Interpreting (Remember, Understand, Analyze, Evaluate)		b) Reading (Interpreting Program Functionality)	
		ii. Producing (Apply, Create)	c) Implementing (Development of Executable Code)		
4. Metacognitive Processes		4. Metacognitive Processes	a) Team Work	i. Solving Problems Together ii. Implementing an Idea Together iii. Division of Work	
			b) Self-Direction	i. Documentation Use	

Tabelle 1 Gegenüberstellung der Kompetenzmodelle (eigene Darstellung basierend auf (Kramer et al., 2016, S. 5) & den Learn to proGrAME Ressourcen)

Von diesem, für die Objektorientierte Programmierung im Allgemeinen entworfenen, Kompetenzmodell ausgehend, wurde im Zuge des Projekts Learn to proGrAME ein Kompetenzmodell speziell für die Entwicklung von Computerspielen erarbeitet. Der grundsätzliche Aufbau und die Strukturierung des Modells von Kramer et al. blieben dabei erhalten. Einige wenige Subdimensionen, welche für den Unterricht im Rahmen des Projekts bzw. für die Spielentwicklung im Allgemeinen weniger Bedeutung haben, wurden weggelassen. Andere speziell für die Spielentwicklung wichtige Aspekte wurden eingefügt und insgesamt wurden die, im ursprünglichen Modell allgemein gehaltenen, Dimensionen (insbesondere 2 und 4) in Bezug auf die Spielentwicklung und die Arbeit im Unterricht präzisiert. In Tabelle 1 ist eine Gegenüberstellung beider Modelle zu sehen.

### 3.3. Das Kompetenzmodell zur Objektorientierten Spielentwicklung

Das *Learn to proGrAME Kompetenzmodell* (Tabelle 1) erfasst also neben Aspekten zur Objektorientierten Programmierung auch die Organisation und Repräsentation von Quellcode sowie Aspekte des Computational Thinking und Soft Skills. Ein wichtiger Aspekt, der in Bezug auf die Entwicklung von Computerspielen in diesem Modell fehlt, ist die Produktion und Verwendung digitaler Medien. Diese stellt insbesondere durch die Verbreitung von benutzerfreundlichen Game Engines einen wesentlichen Aspekt in der Erstellung von Computerspielen dar, der nicht weniger relevant als die Programmierung ist. Daher wurde das Modell noch um den Kompetenzbereich *Digitale Medien* erweitert.

Da die in den Unterdimensionen aufgelisteten Bereiche nicht wirklich unter den Aspekt der Metakognition passen, wurde die Dimension 4 *Meta Cognitive Processes* umbenannt. Um die Bereiche Team Work, Self Direction und Creativity & Originality zu einer Hauptdimension zusammenzufassen wird im fertigen Modell der Begriff *Soft Skills & Methodenkompetenzen* verwendet. Ein weiterer, in diesem Modell fehlender Aspekt, ist das eigenständige Recherchieren zu Problemlösungen. Diese Handlungsdimension wird im DikiKomp12 im Punkt 3.3 Suche, Auswahl und Organisation von Information beschrieben (vgl. Bundes- und Koordinationszentrum eEducation Austria, kein Datum). Für das eigenständige Arbeiten in Projekten spielen Kompetenzen in diesem Bereich eine wichtige Rolle, daher wurde in 4b unter Self Direction noch der Punkt Recherche eingefügt.

Daraus ergibt sich das Kompetenzmodell für die Objektorientierte Spielentwicklung, welches die Basis für diese Forschungsarbeit darstellt (Siehe Tabelle 2). Falls nicht explizit anders beschrieben, wird im Folgenden auf dieses Kompetenzmodell verwiesen, wenn von „dem Kompetenzmodell“ die Rede ist.

Kompetenzmodell zur Objektorientierten Spielentwicklung		
1. Programmierkonzepte	a) Datenstrukturen & Operatoren	i.Variablen ii.Datentypen iii.Operatoren iv.Gültigkeitsbereich & Zugriffskontrolle
	b) Kontrollstrukturen	i.Verzweigung ii.Schleifen
	c) Objekte & Klassen	i.Funktionen ii.Klassen (Erzeugung, Kapselung, Vererbung)
	d) Zustandsbasierte Automaten	i.Objektzustände ii.Zustandsübergänge
2. Darstellung, Code Stil & Dateiorganisation	a) Namensgebung	
	b) Programmierkonventionen & Code Stil	
	c) Dateiorganisation	
3. Kognitive Prozesse	a) Problemlösungsorientierung	i.Algorithmen ii.Testen iii.De-Bugging
	b) Interpretieren von Code	
	c) Implementieren von Code	
4. Teamwork & Methodenkompetenzen	a) Teamwork	i.Gemeinsam Probleme lösen ii.Gemeinsam Ideen umsetzen iii.Arbeitsteilung
	b) Selbstbestimmtes Arbeiten	i.Verwendung der Programmdokumentation ii.Recherche
	c) Kreativität & Originalität	
5. Digitale Medien	a) Erstellung & Verarbeitung	i.Sprites ii.3D-Modelle iii.Sound iv.Animationen
	b) Copyright & Lizenzierung	

Tabelle 2 Das Kompetenzmodell zur Objektorientierten Spielentwicklung

## 4. Interviewbasierte Code Reviews

### 4.1. Überblick

Im Rahmen dieser Forschungsarbeit wurde ein Design Pattern für die sogenannten Interview Based Code Reviews (IBCR) erstellt. Das Konzept der Design Patterns geht auf den Architekten Christopher Alexander zurück. Ziel ist dabei die Sammlung und Verallgemeinerung von Lösungen für wiederkehrende Probleme. Inzwischen werden Design Patterns auch in vielen anderen Disziplinen wie unter anderem Software Engineering, Cinema Studies und Interaction Design eingesetzt. (vgl. Mor, Winters, & Steven, 2010, S. 32)

Each pattern describes a problem that occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice. (Alexander, 1977, S. x)

IBCRs stellen ein Werkzeug dar, welches es ermöglicht, die individuellen Lernerfahrungen von Schüler\*innen in einem Softwareprojekt zu erfassen und in ein Kompetenzmodell einzubetten.

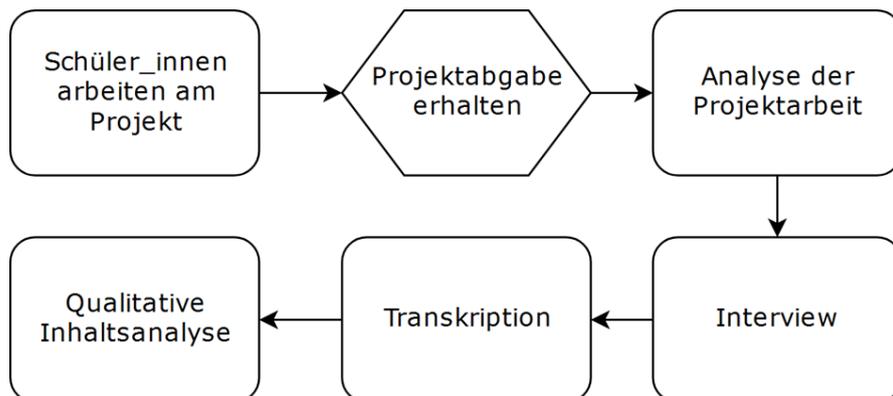


Abbildung 2 Ablauf der Interviewbasierten Code Reviews

IBCRs laufen, wie in Abbildung 2 zu sehen, in mehreren Phasen ab. Wobei die Analyse der Projektabgabe, das Interview sowie die Analyse ebendieser hier die zentralen Rollen spielt.

Für die Analyse der Projektarbeit wird das abgegebene Material nach fest vorgegebenen Kriterien analysiert. Diese Kriterien richten sich nach dem in Kapitel 3 erarbeiteten Kompetenzmodell für Objektorientierte Spielentwicklung. Für das Interview wird dann anhand festgelegter Kriterien und der Analyse der Abgabe ein individuell angefertigter Interviewleitfaden erstellt. Anschließend daran werden das Projekt und der Quellcode mit den Schüler\*innen gemeinsam am Computer besprochen. Dieses Interview ähnelt den bei Programmieraufgaben an Universitäten üblichen Abgabegesprächen. Für die anschließende Transkription wird während des Interviews – neben dem Gesprochenen – auch das Geschehen am Monitor aufgezeichnet, sodass der Kontext des Gesprochenen für die Transkription

ersichtlich ist. Die genauen Transkriptionsrichtlinien hierzu befinden sich im Anhang. Für die Analyse der Interviews werden die Transkripte einer Qualitativen Inhaltsanalyse nach Mayring (Mayring, 2014) unterzogen.

#### 4.2. Vom Kompetenzmodell zum Interviewleitfaden

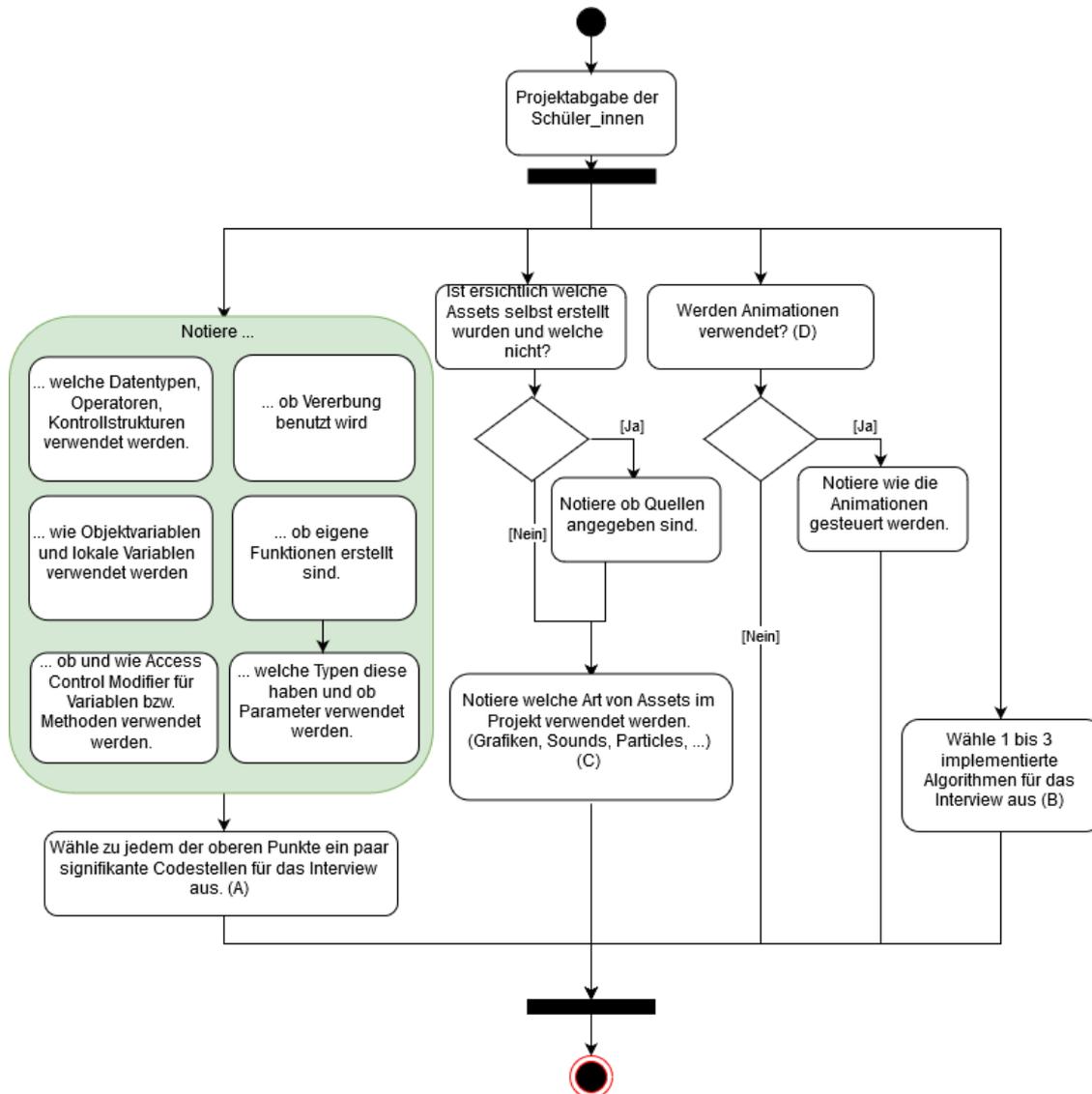


Abbildung 3 Ablaufdiagramm erste Analyse der Abgabe

Die für die IBCRs verwendeten Interviewleitfäden sollen für die einzelnen Projekte der Schüler\*innen individuell angepasst werden. Ein Grund hierfür ist, dass Spieleprojekte schnell sehr groß werden und es unmöglich ist, das gesamte Projekt bzw. den gesamten Quellcode anzusehen, daher soll dieses schon im Vorhinein betrachtet und die relevanten Teile notiert werden, sodass nur diese beim Interview besprochen werden. Ein weiterer Grund ist, dass die Art der Projekte und die tatsächlichen Tätigkeiten auch sehr variieren. Beispielsweise gibt es Projekte in denen gar keine eigenen Assets, keine Sounds etc. verwendet werden oder wenig bis gar kein eigener Code geschrieben wurde. Hier ergibt es wenig Sinn in einem Interview dazu

Fragen zu stellen. Weiters soll mit dem Interviewleitfaden das Kompetenzmodell möglichst abgedeckt werden.

Um diesen Anforderungen gerecht zu werden, wird das abgegebene Projekt zunächst einer ersten Analyse unterzogen und die Ergebnisse in speziell erstellte Vorbereitungsbögen eingetragen. In Abbildung 3 ist ein Ablaufdiagramm dieser vorbereitenden Auswertung zu sehen.

In dieser ersten Analyse wird die Abgabe auf Anzeichen für eine Anwendung der Kompetenzbereiche 1, 3 und 5 untersucht. Dabei werden Codestellen, Algorithmen und Informationen zu verwendeten Assets, sowie deren Ursprung festgehalten, welche später im Interview besprochen werden.

Im Interview werden sechs Hauptthemen behandelt. Wobei den einzelnen Themenbereichen Kompetenzbereiche zugeordnet werden.

	Zugehörige Kompetenzbereiche
1. Vorkenntnisse	-
2. Teamarbeit	4a) - Teamwork
3. Programmkonstrukte	1) Programmierkonzepte 3b) Interpretation von Code 3c) Implementieren von Code
4. Algorithmen	3a) Problemlösungsorientierung 3b) Interpretation von Code 3c) Implementieren von Code
5. Digitale Medien/Assets	1d) Zustandsbasierte Automaten 5a) Digitale Medien
6. Hürden im Projekt	3a) Problemlösungsorientierung 4) Teamwork und Methodenkompetenz

*Tabelle 3 Hauptthemen im Interview mit zugehörigen Kompetenzbereichen*

Wenn man die Abdeckung dieser Hauptthemen mit dem Kompetenzmodell vergleicht, fällt auf, dass die Dimension 2 *Darstellung, Code Stil & Organisation* fehlt. Diese Dimension wird nicht im Interview behandelt, denn sie kann im Anschluss daran direkt am abgegebenen Material analysiert werden. Zum Zeitpunkt der Durchführung der Interviews war eine solche separate Analyse geplant, da sich Kompetenzen in diesem Bereich direkt am Sourcecode bzw. den abgegebenen Dateien manifestieren. Allerdings konnte diese separate Analyse aus Zeitmangel in diesem Forschungsdurchgang nicht mehr erarbeitet und durchgeführt werden. Wie diese möglicherweise aussehen könnte wird in Kapitel 6.3 behandelt.

Außerdem ist an dieser Stelle festzuhalten, dass die Zuordnung der Kompetenzbereiche in Tabelle 3 nur einen groben Überblick über die intendierte Abdeckung des Modells durch die

Themen darstellt und keine scharfe Abgrenzung ist. Beispielsweise ist es gut möglich, dass *3b Interpretieren von Code* auch in Punkt 6. Hürden im Projekt vorkommt.

<p>1. Vorkenntnisse</p> <p>1.1. Wie viel Erfahrung hattest du vor dem Projekt mit Unity?</p> <p>1.2. Hast du dich schon mit Programmieren beschäftigt bevor du dich mit Unity auseinandergesetzt hast?</p> <p>1.3. Beschäftigst du dich auch außerhalb der Schule mit Programmieren bzw. Spielentwicklung?</p> <p>1.4. Erkläre kurz, worum es in deinem Spiel geht und führe es kurz vor.</p>
<p>2. Teamarbeit</p> <p>Falls im Team gearbeitet wurde:</p> <p>2.1. Wie war eure Aufteilung?</p> <p>2.2. Wie war das Arbeiten im Team? Welche Vor- bzw. Nachteile ergaben sich dadurch?</p> <p>2.3. Benutztet ihr Tools zum kooperativen Arbeiten?</p> <p>Falls ja:</p> <p>2.3.1. Welche und wieso?</p>
<p>3. Programmkonstrukte</p> <p>Für alle ausgewählten Codestellen aus (A):</p> <p>3.1. Du hast an dieser Stelle ... verwendet. Erkläre kurz wieso und was dies macht.</p>
<p>4. Algorithmen</p> <p>Für alle ausgewählten Algorithmen aus (B):</p> <p>4.1. Erkläre die Aufgabe und Funktionsweise dieses Algorithmus</p>
<p>5. Digitale Medien/Assets</p> <p>Falls Fremde Assets verwendet wurden:</p> <p>5.1. Woher stammen die verwendeten Assets?</p> <p>5.2. Unter welchen Lizenzen stehen sie?</p> <p>Für alle Asset-Arten aus (C):</p> <p>5.3. Womit erstelltest du die Assets?</p> <p>5.4. Hattest du davor schon Erfahrung mit dem Tool?</p> <p>Falls Animationen verwendet werden:</p> <p>5.5. Wie hast du die Animationen erstellt?</p> <p>5.6. Erkläre wie die Steuerung der Animationen funktioniert.</p>
<p>6. Hürden im Projekt</p> <p>6.1. Welche Probleme traten im Laufe des Projektes auf?</p> <p>Für jedes dieser Probleme:</p> <p>6.2. Wie versuchtet ihr dieses Problem zu lösen?</p> <p>Falls Debugging durchgeführt wurde:</p> <p>6.2.1. Wie wurde das Debugging durchgeführt?</p> <p>Falls zu dem Problem recherchiert wurde:</p> <p>6.2.2. Wo und wie habt ihr recherchiert?</p> <p>6.3. Konntet ihr das Problem lösen?</p> <p>Falls Ja:</p> <p>6.3.1. Wie konntet ihr das Problem lösen?</p> <p>Falls Nein:</p> <p>6.3.2. Woran scheiterte es?</p>

Tabelle 4 Interviewleitfaden

Zu den oben genannten Hauptthemen wurden Fragen erarbeitet, welche den allgemeinen Interviewleitfaden bilden (Tabelle 4). Dieser allgemeine Leitfaden bildet in Verbindung mit dem ausgefüllten Vorbereitungsbogen den individuellen Interviewleitfaden.

Der Aufbau des Interviewleitfadens richtet sich nach den oben beschriebenen Hauptthemen. Wobei zu jedem Hauptthema Fragen und Unterfragen formuliert wurden. Manche Fragen werden entsprechend der Ausarbeitungen im Zuge der vorhergehenden Analyse der Projektabgabe wiederholt oder ausgelassen, wie aus Tabelle 4 zu entnehmen ist. Ziel der Fragen ist es, ein möglichst gutes Bild über vorhandene Kompetenzen der Interviewten in den bestimmten Bereichen zu erhalten. Hierfür ist es insbesondere nötig, bei zu knappen Antworten näher auf das Thema einzugehen und weiter nachzufragen. Außerdem spielen für das Interview nicht nur die gesprochenen Antworten eine Rolle, sondern auch die Handlungen der Interviewten am Computer, welche ebenfalls in Form von Video aufgezeichnet werden.

### 4.3. Ablauf der Analyse

#### 4.3.1. Strukturierende Inhaltsanalyse

Die Qualitative Inhaltsanalyse der Interviewtranskriptionen wurde als Strukturierende Inhaltsanalyse mit einfachen Kategorienlisten (nominales Skalenniveau) durchgeführt. Diese bildet eine der drei grundlegenden Techniken der Qualitativen Inhaltsanalyse (vgl. Mayring & Fenzl, 2014, S. 547). Hierbei handelt es sich um deduktive Kategorienanwendung (vgl. Mayring & Fenzl, 2014, S. 548). Das Kategoriensystem wird also im Vorhinein im Hinblick auf die Forschungsfrage erarbeitet. Da die Forschungsfrage konkret darauf abzielt, welche Kompetenzen gefördert werden, liegt eine Kategorienbildung anhand der verschiedenen Kompetenzbereiche nahe.

Der Ablauf der Strukturierenden Inhaltsanalyse ist in Abbildung 4 zu sehen. Zu Beginn wird die Forschungsfrage

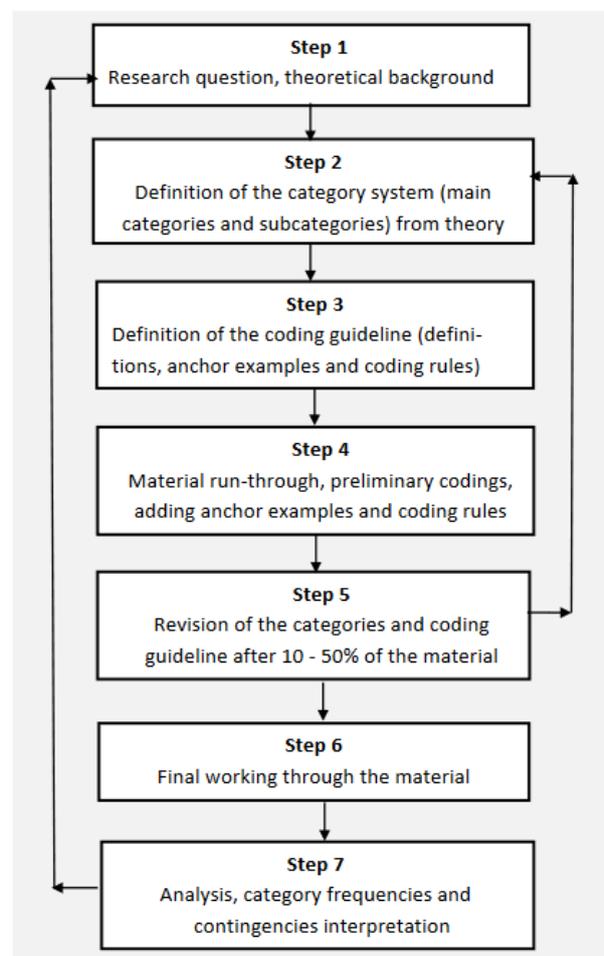


Abbildung 4 Ablauf der Strukturierenden Inhaltsanalyse (Mayring, 2014, S. 96)

festgelegt und der theoretische Hintergrund beschrieben. Anschließend werden auf die Forschungsfrage, sowie das zu untersuchende Material zugeschnittene und theoretisch fundierte Kategorien erarbeitet. In Schritt 3 wird nun der Kodierleitfaden erstellt. Das heißt, die genaue Definition der Kategorien und dazugehörige Ankerbeispiele und Kodierregeln werden festgelegt. Im Anschluss daran wird mit der Kodierung der Texte begonnen. Falls das Kategoriensystem unvollständig bzw. zu unpräzise ist oder sonstige Probleme aufweist, muss dieses wieder überarbeitet werden und man kehrt zu Schritt 2 zurück. Ist das Kategoriensystem in Ordnung wird das gesamte Material mithilfe der Kategorien kodiert und danach analysiert. (vgl. Mayring, 2014, S. 97f)

#### 4.3.2. Kategorienbildung und Durchführung

Für die erste Bildung von groben Kategorien liegen, die im Kompetenzmodell erfassten, Kompetenzbereiche nahe. Das heißt Textstellen in den Interviews, die auf ein Vorhandensein oder auch nicht Vorhandensein von Wissen oder Kompetenzen hinweisen, sollen kategorisiert werden. Für den ersten Durchlauf wurde für jede der fünf Hauptdimensionen im Kompetenzmodell eine Oberkategorie und für jede Subdimension dieses Bereichs eine Unterkategorie erstellt. In einem ersten Durchlauf wurde schnell klar, dass dieses einfache Kategoriensystem nicht ausreichend ist. Einerseits stellte sich heraus, dass in keiner der Kategorien Wissen oder Kompetenzen in Bezug auf den spezifischen Umgang mit der Game Development Engine Unity erfasst wurden. Aussagen in diesem Bezug kamen aber durchaus öfter vor, sodass hierfür die Kategorie *Unityspezifisches Wissen und Kompetenzen* erstellt wurde. Andererseits manifestierte sich, dass bestimmte Unterkategorien gar nicht, oder kaum vorkamen, sodass in diesen Bereichen Unterkategorien zusammengefasst wurden. Weiters wurde die Subdimension *3c Implementieren von Code* fallengelassen, da im Interview direkt das Implementieren nicht überprüft wird. Man könnte dafür argumentieren, dass der Code ja von den Schüler\*innen erstellt wurde und somit das Verständnis des vorhandenen Codes ein klarer Hinweis auf Kompetenzen in diesem Bereich ist. Da diese Stellen dann aber ohnehin auch in die Kategorie *3b Interpretieren von Programmcode* fallen, erschien dies als wenig sinnvoll. Die Subdimension *2a Namensgebung* wurde ebenfalls verworfen, da dieser Aspekt besser direkt anhand der Sourcecodes beleuchtet wird.

Für die Kodierung wurde die Software MAXQDA<sup>9</sup> verwendet. Zur Beurteilung der Objektivität der Kodierung (Interkoderübereinstimmung), wurde nach Fertigstellung des Kategoriensystems<sup>10</sup> und Kodierung aller Interviews, eine weitere Kodierung von Hubert Mayer

---

<sup>9</sup> <https://www.maxqda.de/>

<sup>10</sup> Das endgültige Kategoriensystem befindet sich im Anhang.

BE<sub>d</sub>.<sup>11</sup> durchgeführt. Die Überprüfung der Interkoderübereinstimmung erfolgte nach der Stufe 2 Variante (Mayring, 2014, S.114). Das heißt der zweite Kodierer erhielt die Transkriptionen, den Kodierleitfaden, sowie das Kategoriensystem, nicht jedoch die Kodierung des Erstkodierers. Nach Abschluss beider separaten Kodierungen, wurde mit MAXQDA die Interkoderübereinstimmung als Codeüberlappung der Segmente in Prozent berechnet. Bei einem Toleranzwert von zehn Prozent<sup>12</sup> ergab sich eine Überlappung von 67,2%. Die Berechnung des zufallsbereinigten Cohen Kappas (Greve & Wentura, 1997) ergab einen Wert von 0,65. Ein Wert zwischen 0,61 und 0,80 kann als *beachtlich (substantial)* eingestuft werden (vgl. Landis & G., 1977, S. 165). Jedes der nicht übereinstimmenden Segmente wurden im Anschluss gemeinsam mit dem Zweitkodierer diskutiert und in Folge dessen angepasst, wie in (Mayring, 2014, S. 114) empfohlen.

---

<sup>11</sup> Mitarbeiter am Institut CSLEARN-Educational Technologies der Fakultät für Informatik der Universität Wien

<sup>12</sup> Bei einem Toleranzwert von 10% werden zwei überlappende Segmente als Übereinstimmung gewertet, wenn der überlappende Bereich min. 90% des gesamten abgedeckten Bereichs des Segments beträgt.

## 5. Ergebnisse

### 5.1. Beschreibung der Projekte

Um die Ergebnisse der Inhaltsanalyse besser kontextualisieren zu können, werden in diesem Unterkapitel die betrachteten Projekte kurz beschrieben. Im Rahmen dieser Arbeit wurden fünf Projekte von fünf Teams bzw. Einzelpersonen analysiert. Die einzelnen Schüler\*innen konnten ihre Projekte sehr frei gestalten. Ihnen wurde bloß nahegelegt, entweder das im Unterricht erstellte Spiel Bouncyfant zu modifizieren oder ein kleines, eigenständiges Spiel zu erstellen.

Projektname	Teamgröße	Schulstufe	Beschreibung
Cookie	2	5	Ein 2D-Platformer ähnlich Bouncyfant mit eigener Spielfigur
Bouncyfant	1	5	Kleine Modifikationen des anhand des Tutorials erstellten Spiels Bouncyfant
Space Asteroids	1	7	Modifikation des Spiels Space Asteroids anhand des Tutorials
Die magische Muschel	3	5	Lokales Multiplayergame. Zwei Fische müssen versuchen zu einer magischen Muschel zu gelangen.
Lilia Grinn	2	7	Lokales Multiplayer Puzzlegame.

*Tabelle 5 Übersicht über die Projekte*

Die Vorgehensweise im Projekt wurde ihnen ebenfalls frei überlassen. Lediglich das Team des Projekts *Die magische Muschel* bekam den Auftrag, während der Projektarbeit auch ein Protokoll zu führen. Weiters ist anzumerken, dass die für die Projekte aufgewendete Zeit in der Schule mit in etwa vier Doppelstunden sehr begrenzt war. Diese Zeit ist bei weitem nicht ausreichend, um selbst ein sehr kleines Spiel fertig zu entwickeln und auch für die Modifikation eines bestehenden Spiels sehr knapp bemessen.

*„Also wir hatten irgendwie zu wenig Zeit dafür. Wir hätten ein bisschen mehr Zeit gebraucht, damit wir das wirklich halt machen können.“ (Magische Muschel 32)*

Aus diesem Grund war mit Ausnahme des Projekts *Space Asteroids* keines der Projekte fertig. *Space Asteroids* stellt allerdings kein richtig eigenständiges Projekt dar, da es anhand eines Tutorials erstellt wurde. In Bezug auf den Umfang der Projekte hebt sich *Lilia Grinn* deutlich von den anderen Projekten ab. Das liegt vermutlich zum einen daran, dass das dazugehörige Team als einziges auch in der Freizeit an dem Spiel arbeitete, zum anderen sind sie auch zwei Schulstufen höher als die drei anderen Projektteams und verfügen über entsprechend mehr Vorwissen und Erfahrung, speziell im Bereich der Programmierung.

## 5.2. Codeabdeckung

In Tabelle 6 sieht man die Anzahl der kodierten Textstellen der einzelnen Projekte. Hierfür wurden die entsprechenden Subkategorien zu den Oberkategorien zusammengefasst.<sup>13</sup> Es ist wichtig anzumerken, dass diese Häufigkeiten lediglich bedeuten, wie viele Textstellen in den Interviews zu finden sind, die auf ein Vorhandensein oder Nichtvorhandensein von Kompetenzen in den entsprechenden Bereichen hindeuten. Höhere Zahlen bedeuten keineswegs stärkere Ausprägungen vorhandener Kompetenzen. Um konkrete Aussagen über bestimmte vorhandene Kompetenzen zu tätigen, müssen die kodierten Segmente also erst noch näher betrachtet werden. Außerdem fällt auf, wenn man Tabelle 6 und Tabelle 7 vergleicht, dass die Summe der Unterkategorien nicht die Anzahl der mit den Oberkategorien kodierten Segmente wiedergibt, da eine Oberkategorie bei Mehrfachkodierung durch Unterkategorien nur einmal gezählt wird.

	Cookie	SpaceAsteroids	MagischeMuschel	Bouncyfant	LiliaGrinn	Total
1 Programmierkonzepte	2	13	12	13	34	74
2bc Code Stil & Dateiorganisation	1	0	0	0	5	6
3 Kognitive Prozesse	1	10	4	10	16	41
4 Soft Skills & Methodenkompetenzen	3	3	5	3	12	26
5 Digitale Medien	6	6	3	4	7	26
Unityspezifisches Wissen und Kompetenzen	4	14	0	5	6	29
SUMME	17	46	24	35	80	

Tabelle 6 Häufigkeiten der kodierten Segmente der Oberkategorien je Projekt

Auffallend ist in Tabelle 6, dass die Kategorie *2bc Codestil & Dateiorganisation* nur in zwei Projekten vorkommt und auch in diesen nur sehr selten. Mit dieser Kategorie (welche keine weiteren Unterkategorien hat) wurden einerseits jene Textstellen kodiert, die Auskunft darüber geben, wie die Dateien im Projekt organisiert und gespeichert wurden, ob beispielsweise Tools zur Dateiorganisation bzw. Versionskontrolle (z.B. Git) verwendet wurden. Weiters wurden auch Segmente kodiert, welche andeuten, dass gezielt versucht wurde, durch bestimmte Maßnahmen Code übersichtlicher zu gestalten. Die geringe Abdeckung resultiert daraus, dass Kompetenzen in diesem Bereich mit dem Interviewleitfaden gar nicht abgedeckt wurden. Das heißt, es bedeutet keineswegs, dass in diesem Kompetenzbereich seitens der Lernenden keine Auseinandersetzung stattfand. Es ist eher ein Hinweis darauf, dass die Sinnhaftigkeit dieser Kategorie fragwürdig ist.

In Tabelle 7 stehen die Häufigkeiten der einzelnen Subkategorien je Projekt. Man sieht hier auf einen Blick, dass die Verteilung der Häufigkeiten zwischen den einzelnen Projekten stark variiert.

<sup>13</sup> Mit den Oberkategorien selbst wurde nicht kodiert, wenn es Unterkategorien gibt.

	Cookie	SpaceAsteroids	MagischeMuschel	Bouncyfant	LiliaGrinn	Total
1a Datenstrukturen und Operatoren	0	10	6	7	22	45
1b Kontrollstrukturen	1	3	3	3	7	17
1c Objekte und Klassen	0	1	3	0	3	7
1d Zustandsbasierte Automaten	1	0	0	3	5	9
2bc Code Stil & Dateioorganisation	1	0	0	0	5	6
3a Problemlösungsorientierung	0	2	2	1	10	15
3b Interpretieren von Code	1	8	2	10	7	28
4a Team Work	1	0	4	2	6	13
4b Selbstbestimmtes Arbeiten	2	3	1	2	6	14
5a Erstellung und Verarbeitung	5	6	2	2	7	22
5b Copyright & Lizenzierung	1	0	1	2	0	4
Unityspezifisches Wissen und Kompetenzen	4	14	0	5	6	29
SUMME	17	47	24	37	84	

Tabelle 7 Häufigkeiten der kodierten Segmente der Unterkategorien je Projekt

So gibt es kein einziges Projekt, das alle Kategorien abdeckt. Alle Kategorien – mit Ausnahme von 2bc – wurden in mindestens drei Projekten vorgefunden, aber welche Kategorien in einem Projekt vorkommen, ist von Projekt zu Projekt unterschiedlich. Dies kann als Indiz für die Hypothese *H2: Der tatsächliche Lernerfolg in bestimmten Kompetenzbereichen variiert stark zwischen einzelnen Schüler\*innen* interpretiert werden. Das Fehlen von Items in bestimmten Kategorien ist in manchen Fällen leicht zu erklären. Beispielsweise wurden für Lilia Grinn sämtliche verwendeten Grafiken und Sounds eigenständig erstellt. Demnach mussten sich die Teammitglieder nicht mit dem Bereich *5b Copyright & Lizenzierung* befassen. Ähnlich ist es bei Space Asteroids. Hier wurde alleine gearbeitet und nur Assets aus dem Tutorial verwendet, somit ist es wenig verwunderlich, dass *4a Team Work* und der gesamte Bereich *5 Digitale Medien* nicht vorkommen.

Interessant hingegen ist die Tatsache, dass im Projekt *Die Magische Muschel* zur Kategorie *Unityspezifisches Wissen und Kompetenzen* keine Items vorhanden sind. Bei näherer Betrachtung des Videomaterials des zugehörigen Interviews fällt auf, dass in diesem Interview, im Gegensatz zu den anderen, ich als Interviewer den Computer bediente und nicht die Interviewten. Diese Tatsache könnte ein Grund für das Fehlen sein. Denn Wissen im Umgang mit Unity manifestierte sich in den Interviews nicht nur direkt durch das Gesagte im Gespräch, sondern auch durch die Interaktion der Interviewten während des Interviews mit dem Programm selbst. Dies wurde durch die Art der Durchführung des Interviews leider verhindert. Die suboptimale Interviewsituation kam aufgrund eines engen Zeitfensters für das Interview zu Stande, welches sich unglücklicherweise mit einer Brandschutzübung im Schulgebäude überschneidet. Durch die geringe verbleibende Zeit wurde das Interview dann mit einem Laptop im Gang der Schule durchgeführt.

### 5.3. Analyse der kodierten Segmente

Um nun ein möglichst gutes Bild über die vorhandenen und im Projekt angewandten Kompetenzen zu erhalten, werden in diesem Unterkapitel die kodierten Segmente der Interviews im Detail betrachtet. Dafür wird jede Kategorie im Kontext des jeweiligen Projekts einzeln für sich analysiert. Im Zuge dessen wurden die einzelnen Segmente meist in einem zweiten Reduktionsschritt<sup>14</sup> noch einmal in Kategorien eingeteilt, um einen besseren Überblick zu erhalten. Die Vorgangsweise dabei entspricht einer vereinfachten Form der zusammenfassenden Inhaltsanalyse bzw. induktiven Kategoriebildung (Mayring & Fenzl, 2014, S. 547). Aufgrund der kleinen Anzahl der Segmente innerhalb der einzelnen Kategorien und der geringen Länge dieser wurde auf den Schritt der Paraphrasierung verzichtet. Die jeweilig verwendeten Kategorien dieser zweiten Reduktion werden zu jeder Kategorie der ersten Reduktion angeführt (Siehe nachfolgende Tabellen). Bei kontextuellen Unklarheiten der Segmente wurde wieder das gesamte Interviewmaterial herangezogen.

#### 5.3.1. 1a Datenstrukturen und Operatoren

Kategorie	Beschreibung
(V) Variablen	Kompetenzen bzgl. Variablen (Oberkategorie von D und G)
(D) Datentypen	Kompetenzen bzgl. Datentypen
(G) Gültigkeitsbereiche	Kompetenzen bzgl. des Gültigkeitsbereiches von Variablen
(O) Operatoren	Kompetenzen bzgl. Operatoren (exkl. Zuweisungsoperator)
(X) Teilwissen/Fehlkonzepte	Hinweise auf Fehlkonzepte bzw. Teilwissen in verschiedenen Bereichen

Tabelle 8 Kategoriensystem der zweiten Reduktion - 1a Datenstrukturen und Operatoren

	V	D	G	O	X
Cookie	0	0	0	0	0
Space Asteroids	1	4	3	1	4
Magische Muschel	1	2	0	0	1
Bouncyfant	2	3	0	0	2
Lilia Grinn	1	8	4	3	2

Tabelle 9 Häufigkeiten - 1a Datenstrukturen und Operatoren<sup>15</sup>

Im Projekt *Cookie* wurde gar kein eigener Code geschrieben, daher gibt es in dieser Kategorie auch keine kodierten Segmente. In allen anderen Projekten geht klar hervor, dass ein grundlegendes Verständnis über Variablen und Datentypen vorhanden ist. Unter anderem manifestiert sich dies in Aussagen wie:

*Je nach dem Typen sind das [Variablen] entweder Buchstaben beim String oder Zahlen bei float oder int. (SpaceAsteroids 22-23)*

*Also einer Variable, der kann man einen Wert zuweisen und so einen Datentyp. – (MagischeMuschel 45)*

<sup>14</sup> Den ersten Reduktionsschritt stellt die in Kapitel 4.3 beschriebene Inhaltsanalyse dar.

<sup>15</sup> Die Häufigkeit der Oberkategorie sind hier exklusive der Unterkategorien

Bezüglich der verwendeten Datentypen wurde in den vier Projekten genannt, dass String der Datentyp für Text oder Buchstaben ist. Weiters kamen in den vier Projekten auch die Datentypen *float* und *int* vor. Allerdings konnte die Frage nach der Bedeutung von *float* in *Magische Muschel* nicht beantwortet werden. Interessant in Bezug auf *float* und *int* sind folgende Segmente aus *Space Asteroids*:

*I: Was heißt float genau? B: Ah ich glaub das halt unterschiedlich zwischen int und float abgerundet wird. – (SpaceAsteroids 24-25)*

*I: Also float sind Fließkommazahlen, also Dezimalzahlen und int sind nur ganze Zahlen. In dem Fall, eine Kommazahl für die Geschwindigkeit ist das wichtig glaubst du oder hast du das bewusst als float gemacht? B: Ja das ist ganz wichtig, damit das halt stabil langsamer und schneller werden kann. – (SpaceAsteroids 26-27)*

Aus dem ersten Segment geht hervor, dass bezüglich der beiden Datentypen ein Fehlkonzept vorliegt. Nach der Erklärung ebendieser schließt der Interviewte aber sofort auf die Bedeutung und auf die Relevanz des Datentyps in Bezug auf das Spiel.

Weiters wurden in *Lilia Grinn* auch komplexere Datentypen, wie Enum (*LiliaGrinn* 47-48), Vector3 (*LiliaGrinn* 133), Liste sowie ein zweidimensionales Array (*LiliaGrinn* 88 & 90), verwendet. Mit Ausnahme von Enum wurden diese Typen auch korrekt erklärt. Bei Enum erklärten die beiden Interviewten die Beschreibung in eigenen Worten.

*B2: Das ist so ähnlich wie ein Array, aber eben mit so vorgesetzten Datentypen und später machen wir auch ein kleine, eine kleinere Variable dieses Enums, indem wir dann speichern können, ob es jetzt eben wie hier entweder Moving oder Standing ist. Und mit dem können wir dann auch einfach agieren. Weil, das Problem bei Arrays ist eben, man kann es zwar auch mit einem Array machen, aber das kann man entweder mit Integer machen, eben mit Null und Eins. Aber dann muss man sich immer merken, was jetzt welches ist und. – (LiliaGrinn 47)*

*B1: Also ich würde es sogar eher als ein Boolean mit mehreren Wertmöglichkeiten quasi beschreiben. Ich mein ja wir haben es in unserem Fall quasi mit einem, also es hätte ein Array sein sollen, aber wir haben es mit einem Enum ersetzt, aber eigentlich finde ich ist es eigentlich ein Boolean mit mehreren Werten, mit mehreren Möglichkeiten. – (LiliaGrinn 48)*

Die Beschreibung von B2 ist hier zwar etwas schwammig und nicht korrekt, allerdings zeigt sich durch die Beschreibung dessen, dass das Enum zur Kodierung der Zustände *Moving* oder *Standing* wird, durchaus ein Verständnis für diesen Datentyp. Die Beschreibung von B1 hingegen, als Boolean mit mehreren Wertmöglichkeiten, ist sehr treffend.

Der Gültigkeitsbereich von Variablen kam in den Projekten *Space Asteroids* und *Lilia Grinn* zur Sprache. In *Space Asteroids* wird genannt, dass Objektvariablen in der gesamten Klasse verwendet werden können (*SpaceAsteroids* 34-37), dass die Schlüsselwörter *private* und *public* regeln, wie auf Variablen zugegriffen werden kann, und dass *public* Variablen im Unity-Editor sichtbar sind und *private* Variablen nicht (*SpaceAsteroids* 57-62). Ebenso wurden in *LiliaGrinn* bewusst Zugriffsmodifikatoren gewählt, um den Zugriff auf Variablen von anderen Skripten zu

ermöglichen (LiliaGrinn 64) und der Gültigkeitsbereich einer lokalen Variable wurde korrekt erkannt (LiliaGrinn 91-92).

Operatoren kamen ebenfalls nur in *Space Asteroids* und *Lilia Grinn* vor. In ersterem wurde der Vergleichsoperator im Kontext eines If-Statements beschrieben (SpaceAsteroids 55-56). In *Lilia Grinn* wird neben dem Vergleichsoperator auch das *inklusive Oder* erklärt (LiliaGrinn 64, 80 & 82). Außerdem kommt der Divisionsoperator in einem eher ungewöhnlichen Kontext vor. Und zwar wird ein *Vector3* durch einen *float* dividiert. Die Bedeutung dessen wird korrekt erklärt und angemerkt, dass dies in „In Unity ja zum Glück funktioniert“ (LiliaGrinn 137 & 139). Außerdem wird auch eine valide Alternative für den Fall angeführt, dass die Division dieser zwei verschiedenen Datentypen in einer Programmiersprache nicht erlaubt wäre. (LiliaGrinn 141)

In den Interviews wurden auch einige falsche oder nur teilweise richtigen Aussagen getätigt. Ein paar davon wurden oben schon erläutert. Beispielsweise wurde die Frage nach der Bedeutung von float mit „Äh float. Das weiß ich gerade nicht.“ (MagischeMuschel 51) beantwortet, obwohl im Programmcode Variablen vom Typ float vorkommen. Dies ist ein klares Indiz dafür, dass eine bloße Betrachtung der Projektergebnisse ein verfälschtes Bild in Bezug auf die Kompetenzmessung erzeugen kann. Weiters wurde auch behauptet, ein Objekt des Typs AudioClip seien ein Audiofile (SpaceAsteroids 31) oder Strings seien „Dateien wo Namen gespeichert werden“ (Bouncyfant 49-52). Diese Aussagen kann man als Halbwissen klassifizieren. Die Aussagen an sich sind zwar falsch, es steckt jedoch ein gewisser wahrer Kern hinter der Aussage, denn einem Objekt des Typs AudioClip liegt in Unity ein Audiofile zugrunde, bzw. kann man in einem String Namen speichern, allerdings nicht nur und weiters ist eine Variable keine Datei. Bei solchen Aussagen ist zu beachten, dass eventuell vorhandene sprachliche Barrieren eine wesentliche Rolle spielen können.

### 5.3.2. 1b Kontrollstrukturen

Kategorie	Beschreibung
(If) Verzweigungen	Kompetenzen bzgl. Verzweigungen (If, Else, switch)
(L) Schleifen	Kompetenzen bzgl. Schleifen
(X) Teilwissen/Fehlkonzepte	Hinweise auf Fehlkonzepte bzw. Teilwissen in verschiedenen Bereichen

Tabelle 10 Kategoriensystem der zweiten Reduktion - 1b Kontrollstrukturen

	If	L	X
Cookie	1	0	0
Space Asteroids	2	0	0
Magische Muschel	2	0	1
Bouncyfant	3	0	1
Lilia Grinn	6	2	1

Tabelle 11 Häufigkeiten - 1b Kontrollstrukturen

Auffallend ist sofort, dass Schleifen nur bei *Lilia Grinn* vorkommen. Hier wurde eine For-Schleife verwendet, um anhand einzelner modularer Segmente verschiedene Spiellevels zu generieren (*LiliaGrinn* 73-74). Deren Funktionsweise wurde im Zuge des Interviews anhand des Codes erklärt (*LiliaGrinn* 71-74). Ein Grund weshalb in den anderen Projekten keine Schleifen vorkommen, ist mitunter, dass Schleifen in der Spielprogrammierung mit Gamedevelopment Engines für einfache Aufgaben oft nicht nötig sind. Soll etwas öfter wiederholt werden, kann dies beispielsweise auch einfach über die *Update()* Funktion geschehen, welche von der Spielschleife periodisch aufgerufen wird. Die Spielschleife selbst, ist jedoch in der Engine fix eingebaut und somit nicht unmittelbar sichtbar. Eine andere Möglichkeit der Wiederholung ist die Verwendung einer Ko-Routine. So werden in *Bouncyfant* fortlaufend Spielobjekte über die Methode *InvokeRepeating()* erzeugt.

Programmverzweigungen sind jedoch in allen Projekten vertreten. Sogar im Projekt *Cookie*, in welchem kein Programmcode direkt geschrieben wurde, kommt in (*Cookie* 56-59) die Funktionsweise einer Programmverzweigung in Bezug auf die Animationssteuerung von Unity zur Sprache. In den anderen Projekten werden zahlreiche If-Anweisungen verwendet. Das grundlegende Verständnis, dass mit dem If-Statement etwas überprüft wird, wenn eine bestimmte Bedingung erfüllt ist, geht klar aus den kodierten Segmenten hervor (*SpaceAsteroids* 38 & 47-54; *Magische Muschel* 62-63; *Bouncyfant* 67 & 74-75; *LiliaGrinn* 76, 82). In *Lilia Grinn* wird außerdem auch noch eine Verkettung von If-Else-Statements verwendet, um den Input für vier Bewegungsrichtungen zu verarbeiten (*LiliaGrinn* 128).

In *Magische Muschel* wurde die Frage danach, was passiert, wenn die Bedingung in einem konkreten If-Statement erfüllt ist, folgendermaßen beantwortet:

*I: wenn jetzt das h ungleich null ist, was passiert dann. Passiert dann das, oder macht er da unten weiter, oder?*

*B2: Also wenn ein Else dann dabei steht, würd es zum Else rüber.*

B1: aber, wenn kein Else dabei ist?

B2: Ich glaub dann, (...) gehts wieder zurück nach oben.

(MagischeMuschel 68-70)

Hier liegt einerseits eine Unklarheit bezüglich der Bedingung vor, dies ist wahrscheinlich aber nur ein kleiner Denkfehler und kein grundlegendes Fehlkonzept. Dass das Programm bei Nichterfüllung der Bedingung im If darüber weitermacht, deutet allerdings auf ein grundlegendes Fehlkonzept in Bezug auf das If-Statement hin.

Weiters wurde in Bouncyfant in Bezug auf ein If-Statement behauptet, wenn die Bedingung nicht erfüllt ist, „ists aus der Schleife draußen“ (Bouncyfant 77). Ebenso ist auch in *Lilia Grinn* einmal die Rede von einer „If-Schleife“ (LiliaGrinn 82). Die fälschliche Bezeichnung des If-Statements als If-Schleife ist allerdings nicht ungewöhnlich und wird durchaus auch von Personen mit viel Programmiererfahrung verwendet. Man könnte den Begriff daher auch einfach als Unwort sehen.

### 5.3.3. 1c Objekte und Klassen

Kategorie	Beschreibung
(K) Kapselung	Kompetenzen bzgl. Sichtbarkeit von Objektvariablen und Methoden
(V) Vererbung	Kompetenzen bzgl. Vererbung und Klassenhierarchien
(M) Methoden	Kompetenzen bzgl. der Verwendung von Klassenfunktionen

Tabelle 12 Kategoriensystem der zweiten Reduktion - 1c Objekte und Klassen

	K	V	M
Cookie	0	0	0
Space Asteroids	(3)	0	0
Magische Muschel	0	3	0
Bouncyfant	0	0	0
Lilia Grinn	0	0	3(6)

Tabelle 13 Häufigkeiten – 1c Objekte und Klassen

Diese Kategorie ist mit nur sieben zugeordneten Segmenten eine der kleineren. Dies ist mitunter ein Resultat dessen, dass im Kategoriensystem die Punkte Zugriffsmodifikatoren und Gültigkeitsbereich von Variablen unter *1a Datenstrukturen und Operatoren* zu finden sind. Die Einteilung ist hier nicht ganz klar und wurde auch mit dem Zweitkodierer diskutiert. Schlussendlich fiel der Entschluss diese mit *1a* zu kodieren. Um vorkommende Segmente hier trotzdem sicher zu machen wurden die zugehörigen Segmente aus *1a* noch einmal betrachtet und die neuen Gesamthäufigkeiten in Tabelle 13 in runden Klammern eingetragen.<sup>16</sup>

Abgesehen von der Kapselung gibt es nur zwei Themen der Objektorientierten Programmierung, die in je einem Projekt vorkommen. Dies ist zum einen das Konzept der Vererbung, welches in *Magische Muschel* sinnvoll eingesetzt wurde, um Codewiederholung zu vermeiden und die

<sup>16</sup> Auf eine zusätzliche Diskussion der Inhalte wurde hier aber verzichtet, da hier im Großen und Ganzen dasselbe gilt wie in Kapitel 5.3.1.

Beziehung der Klassen zweier Fische zu modellieren (MagischeMuschel 74 & 76-78). Aus den Textstellen geht klar hervor, dass das Konzept verstanden und bewusst eingesetzt wurde.

*... damit du nicht für Dori genau den gleichen Code nochmal schreiben musst. Und da wird dann in der Parentklasse eine Funktion hinzugefügt damit dann Dori weiß, dass Nemo das machen muss. Also damit sie das Gleiche machen. Das ist dieses Erben. (MagischeMuschel 76)*

In *Lilia Grinn* wird in einer Klasse eine statische Variable definiert, welche benötigt wird, um ein Level zu generieren. Dies wurde ohne genaue Kenntnis über die Bedeutung dessen aus einem Tutorial im Internet übernommen. Der Schüler, der den Code schrieb, gab an:

*Ich habe mir ein Tutorial im Internet angeschaut und da haben sie static benützt und ich habe zuerst ohne static probiert und dann hats nicht funktioniert. (...) Dann habe ich aber static dazugefügt, und dann hat es auf einmal funktioniert und dann habe ich das einfach mal so hingenommen. (LiliaGrinn 66-68)*

Der andere Schüler im Projekt fügte dann allerdings folgende, richtige Überlegung hinzu:

*Ich nehme an das ist deshalb, weil wir jetzt ohne die Klasse oder ohne das Objekt zu instanzieren in anderen Klassen, dass wir einfach durch static halt direkt darauf zugreifen müssten, ohne ein neues Objekt zu instanzieren ... (LiliaGrinn 70)*

Aus der zweiten Passage geht klar hervor, dass der Schüler über ein zumindest grundlegendes Verständnis über Klassen und Instanzen verfügt.

#### 5.3.4. 1d Zustandsbasierte Automaten

Kategorie	Beschreibung
(A) Unity Animator	Anwendung der Automatentheorie im Unity Animator
(M) Modellierung	Modellierung von Zuständen und Übergängen in Spielobjekten

Tabella 14 Kategoriensystem der zweiten Reduktion - 1d Zustandsbasierte Automaten

	A	M
Cookie	1	0
Space Asteroids	0	0
Magische Muschel	0	0
Bouncyfant	3	0
Lilia Grinn		5

Tabella 15 Häufigkeiten 1d Zustandsbasierte Automaten

Wissen über Zustandsbasierte Automaten kommt explizit aus keinem der Interviews hervor. Allerdings stellt der Unity Animator<sup>17</sup> mit seinen Zuständen und bedingten Zustandsübergängen einen endlichen, deterministischen Automaten dar. Demnach manifestieren sich Kompetenzen in diesem Bereich unter anderem durch Verständnis und korrekte Anwendung des Animators.

Der Animator wurde in den Projekten *Bouncyfant* und *Cookie* verwendet. In (*Cookie* 52-53) wird die Funktionsweise der Zustände und Übergänge korrekt erklärt. Bei *Bouncyfant* geht auch hervor, dass ein grundlegendes Verständnis da ist, allerdings werden die Bedingungen zu den Zustandsübergängen nicht ganz korrekt erklärt (*Bouncyfant* 88 & 90). Der Schüler behauptet

<sup>17</sup> Das Werkzeug zum Steuern von Animationen und deren Übergänge

hier, die Zustandsübergänge werden direkt über Tastatureingaben gesteuert. Genaugenommen werden im Programmcode, falls entsprechende Tasten gedrückt werden, Variablen gesetzt. Diese Variablen werden dann in den Bedingungen für die Zustandsübergänge abgefragt. Nach kurzem Nachhaken wurde die Bedingung für den Zustandsübergang aber richtig erklärt.

In *Lilia Grinn* wird, wie in Kapitel 5.3.1 beschrieben, eine Enumeration verwendet, um den Zustand der Spielfiguren zu modellieren (*LiliaGrinn* 50, 52 & 128). In der Erklärung des Algorithmus für die Grid-basierte Bewegung ebendieser wird erwähnt, dass abhängig vom momentanen Zustand und den Tastatureingaben von einem Zustand in einen anderen übergegangen wird (*LiliaGrinn* 128). Da dieses Verhalten genau dem eines einfachen endlichen Automaten entspricht, zeigt sich hier, dass sie die zugrundeliegenden Konzepte verstanden haben und umsetzen können, auch wenn die Lernenden womöglich noch nie den Begriff des Automaten (Modellrechner) gehört haben.

### 5.3.5. 2bc Code Stil und Dateiorganisation

Wie in Kapitel 5.2 erwähnt, beschreiben die mit *2bc kodierten Segmente* die in diesen Bereich fallenden Kompetenzen nur ansatzweise und keinesfalls vollständig. Dennoch werden der Vollständigkeit halber die entsprechenden Segmente kurz diskutiert.

In *Cookie* wurde lediglich das Einfügen einer Grafik in Unity per Drag and Drop aus einem Ordner kodiert (*Cookie* 46). In *Lilia Grinn* wird von den Lernenden darüber reflektiert, dass die Dateien in ihrem Projekt sehr unstrukturiert gespeichert wurden, und dass dies bei einem größeren Projekt zu Problemen führen kann (*LiliaGrinn* 99-105). Weiters schilderten sie das Problem, mit verschiedenen Versionen von Unity gearbeitet zu haben, was dazu führte, dass sie alle Assets einzeln wieder einfügen mussten (*LiliaGrinn* 143). Als dritten Punkt beschreiben sie, dass sie, um gemeinsam parallel am Projekt arbeiten zu können, separat den Code schrieben und danach händisch zusammenführten (*LiliaGrinn* 41-42). Hier ist anzumerken, dass beide auch nach den Interviews noch weiter das Spiel weiterentwickelten und einige Wochen später für die Dateiorganisation und Versionskontrolle begeistert *Github* verwendeten. *Git* bzw. *Github* kannten die Schüler zum Zeitpunkt des Interviews noch nicht. Das Letzte verbleibende Segment beschreibt die Entscheidung, den Aufbau der Levels, die später anhand eines Skripts aus Tiles erzeugt werden, in einem zweidimensionalen Array zu speichern. Wobei für verschiedene Tiles verschiedene Symbole verwendet wurden.

*Genau und wir haben uns dafür entschieden, weil da sieht man halt das Layer- das Levellayout auf den ersten Blick. Die Einsen sollen jetzt grade den Boden darstellen, aber wenn wir das jetzt durch irgendwas austauschen, dann müssten wir das Level nicht generieren, um das sofort zu sehen wie das aussieht. (Lilia Grinn 90)*

In *Lilia Grinn* zeigen sich klare Aussagen, dass Kompetenzen im Bereich Dateioorganisation vorhanden sind. Insbesondere auch durch das Lernen von *Git* im Zuge des Spielprojektes zeigt sich hier klar, dass Kompetenzen in diesem Bereich durch Spielentwicklung gefördert werden können. Auch der Einsatz des zweidimensionalen Arrays zeigt, dass sie sich Gedanken über die Lesbarkeit und Klarheit des Codes machten.

### 5.3.6. 3a Problemlösungsorientierung

Kategorie	Beschreibung
(C) Computational Thinking	Kompetenzen in Bezug auf Computational Thinking. Insbesondere das Entwerfen und Verstehen von Algorithmen.
(T) Testen von Lösungen	Kompetenzen in Bezug auf das Testen der Korrektheit eines Programms bzw. einer Implementierung.

Tabelle 16 Kategoriensystem der zweiten Reduktion - 3a Problemlösungsorientierung

	C	T
Cookie	0	0
Space Asteroids	2	0
Magische Muschel	0	0
Bouncyfant	1	0
Lilia Grinn	7	3

Tabelle 17 Häufigkeiten 3a Problemlösungsorientierung

In drei Projekten wird je die Funktionsweise eines Algorithmus erklärt. In *Space Asteroids* ist dies ein einfacher Algorithmus zur Kollisionsbehandlung<sup>18</sup> der Asteroiden (Space Asteroids 76 & 78). In *Bouncyfant* wird eine Methode erklärt, welche wiederholt von zufälligen Stellen Müll regnen lässt (Bouncyfant 67-75). Aus den Beschreibungen geht hervor, dass die Funktionsweise des Algorithmus verstanden wurde, allerdings wurden diese nicht von den Lernenden selbst entwickelt, sondern wurden aus Tutorials übernommen. In *Lilia Grinn* wurde ein Algorithmus für eine Grid-basierte Bewegung<sup>19</sup> der Figuren, entwickelt und implementiert. Dessen Funktionsweise wird in (LiliaGrinn 128) genau erläutert. Dieser Algorithmus wurde vom Team selbstständig erarbeitet. In (LiliaGrinn 148, 156 & 169) beschreiben sie, dass dies nicht einfach war.

*... das kann ja nur einfach sein, weil die ganzen Pokémonspiele und alles, die sind ja so lange her, das heißt die Lösung muss ja eigentlich so extrem simpel sein. Ist sie aber nicht tatsächlich. (LiliaGrinn 169)*

Um den Algorithmus zu implementieren, recherchierten sie im Internet und bekamen auch Hilfestellungen von einer Lehrperson, die allerdings als „sehr kompliziert“ wahrgenommen wurden, sodass schlussendlich eine eigene Lösung gesucht und gefunden wurde (LiliaGrinn 156 & 169).

<sup>18</sup> Der Algorithmus beschreibt die auszuführenden Aktionen, wenn ein Asteroid mit einem Objekt kollidiert und keine Physikalische Kollisionsbehandlung.

<sup>19</sup> Das heißt die Figur soll sich schrittweise anhand eines Rasters bewegen. So wie dies beispielsweise bei den Pokémonspielen für den Gameboy üblich war.

Weitere Segmente, die auf Kompetenzen zu Computational Thinking hinweisen, sind (LiliaGrinn 90 & 141), wo respektive die Kodierung der Levelgenerierung und eine alternative Implementierung für ein Codesegment erklärt wird.

Bezüglich des Testens von Programmen geht aus (LiliaGrinn 66-68, 150 & 156) hervor, dass sie die Lösungsstrategie *Trial & Error* anwendeten, wenn etwas nicht funktionierte. Ansonsten lässt sich in den Interviews nichts zu diesem Thema finden. In keinem der Projekte wurde ein Debugger zur Fehlerfindung verwendet und auch ein systematisches Testen von Programmen fand nicht statt.

### 5.3.7. 3b Interpretieren von Code

	Cookie	SpaceAsteroids	MagischeMuschel	Bouncyfant	LiliaGrinn	Total
3b Interpretieren von Code	1	8	2	10	7	28
1 Programmierkonzepte	2	13	12	13	34	74

Tabelle 18 Ausschnitte aus Tabelle 6 und Tabelle 7

In Tabelle 18 sieht man, dass in allen fünf Projekten Stellen kodiert sind, aus denen die Fertigkeit, die Semantik von Programmcode und dessen Auswirkungen auf das Spielgeschehen zu interpretieren, ersichtlich ist. Wobei zu bedenken ist, dass in *Cookie* kein Programmcode geschrieben wurde. Das kodierte Segment beschreibt hier die Funktionsweise des Unity Animators.

Zwischen den beiden in Tabelle 18 dargestellten Kategorien gibt es große Überschneidungen. 27 der 28 mit *3b* kodierten Segmente sind hierbei auch mit Kategorien aus *1 Programmierkonzepte* kodiert. Aus diesem Grund wird hier auf eine ausführliche Diskussion verzichtet.

Aus den Segmenten geht hervor, dass Code, mit dem die Lernenden arbeiteten, von ebendiesen zum Großteil korrekt interpretiert werden kann. Auffallend ist, dass in *Magische Muschel* nur zwei Segmente unter *3b* fallen, während unter *1* annähernd gleich viele Segmente wie in *Space Asteroids* und *Bouncyfant* zugeordnet sind. Dies könnte an der in Kapitel 5.2 angesprochenen unterschiedlichen Interviewsituation liegen.

### 5.3.8. 4a Team Work

Kategorie	Beschreibung
(Z) Art der Zusammenarbeit	Segmente mit Auskunft darüber in welcher Form zusammengearbeitet wurde
(A) Aspekte der Teamarbeit	Aussagen darüber, wie sich die Arbeit im Team auswirkte. Welche Vor- und Nachteile ergaben sich?

Tabelle 19 Kategoriensystem der zweiten Reduktion - 4a Team Work

	Z	A
Cookie	1	1
Space Asteroids	0	0
Magische Muschel	2	1
Bouncyfant	2	0
Lilia Grinn	3	3

Tabelle 20 Häufigkeiten 4a Team Work

Drei der fünf Projekte wurden von Teams erstellt, wobei die Art der Zusammenarbeit stark differierte.

In *Cookie* arbeitete eine Schülerin grundsätzlich selbstständig, allerdings hat ihr eine Kollegin meistens bei der Arbeit zugesehen und geholfen (Cookie 17-20).

Das Projektteam von *Magische Muschel* bestand aus drei Mitgliedern, wovon immer zwei gleichzeitig am Spiel arbeiteten und die dritte Person Protokoll führte. Diese Rollen wurden dabei immer wieder durchgewechselt (Magische Muschel 29 & 32). Diese Arbeitsweise wird als Pair-Programming (Chong, et al., 2005) bezeichnet und wird neben nicht nur im Bildungskontext, sondern auch in der praktischen Softwareentwicklung verwendet. Als Vorteil wird genannt, dass Fehler leichter gefunden werden und auch das Verstehen, des Tutorials zu zweit besser funktionierte (MagischeMuschel 40).

Bei *Bouncyfant* wurde zwar allein gearbeitet (Bouncyfant 13-14), dennoch wurde bei Problemen, die nicht allein lösbar waren, Hilfe geholt (Bouncyfant 98).

Das Team von *Lilia Grinn* arbeitete zu Beginn gemeinsam, ihres Erachtens nahm das allerdings zu viel Zeit in Anspruch und sie kamen sich gegenseitig in die Quere, sodass sie sich spontan ihre Arbeitsbereiche aufteilten (LiliaGrinn 22-25). Das Erstellen und Verarbeiten der Grafiken wurde zur Gänze von einem Schüler übernommen, da der andere meinte, er kenne sich damit nicht aus (LiliaGrinn 27). Programmcode wurde von beiden separat geschrieben und am Ende der Stunde händisch zusammengelegt (LiliaGrinn 41-42). Die Arbeit im Team wirkte sich zum einen motivierend aus. „Wenn man zu zweit das macht, dann lässt man das eher nicht stehen. Dann arbeitet man gemeinsam weiter“ (LiliaGrinn 31). Zum anderen wurde auch genannt, dass man gemeinsam voneinander Ideen bekommt und sich bei Problemen gegenseitig helfen kann (LiliaGrinn 29-30).

Zusammenfassend lässt sich sagen, dass es bei den Teams *Magische Muschel* und *Lilia Grinn* zu Förderungen bezüglich Kompetenzen im Bereich Team Work kam. Erstere lernten das Konzept des *Pair Programmings* kennen und sammelten positive Erfahrungen mit diesem. Zweitere sammelten Erfahrungen durch verschiedene Arten der Zusammenarbeit und konnten einen für sie praktikablen Weg finden und darüber reflektieren.

### 5.3.9. 4b Selbstbestimmtes Arbeiten

Kategorie	Beschreibung
(E) Eigenständiges Lösungsfinden	Selbstständiges suchen von Lösungen ohne äußere Hilfsmittel
(T) Tutorials	Selbstständiges Arbeiten anhand der im Unterricht zur Verfügung gestellten Tutorials
(R) Recherche	Selbstständiges Recherchieren (z.B. in Foren, YouTube, etc.)

Tabelle 21 Kategoriensystem der zweiten Reduktion - 4b Selbstbestimmtes Arbeiten

	E	T	R
Cookie	0	1	1
Space Asteroids	0	1	2
Magische Muschel	0	1	0
Bouncyfant	1	2	1
Lilia Grinn	2	0	3

Tabelle 22 Häufigkeiten 4b Selbstbestimmtes Arbeiten

Mit Ausnahme von *Lilia Grinn* griffen alle Projektteams auf die im Unterricht zur Verfügung gestellten Tutorials zurück. Diese wurden dabei als sehr hilfreich empfunden. (Cookie 75; Space Asteroids 102-104; Magische Muschel 116-119; Bouncyfant 97-100)

*Ohne das [Tutorial] hätten wir es nicht geschafft. (MagischeMuschel 118)*

Bevor in Bouncyfant jedoch das Tutorial zur Hilfe herangezogen wurde, wurde erst versucht, auftretende Probleme selbstständig zu lösen (Bouncyfant 97-98). Extern wurde in diesem Projekt und in *Cookie* nicht recherchiert (Bouncyfant 99-100; Cookie 66-67).

In Space Asteroids trat während des Projekts ein Problem auf, zu welchem im Internet recherchiert und in einem Forum ein Post gefunden wurde, welcher zu einer Lösung verhalf (SpaceAsteroids 111-114).

In Lilia Grinn wurde zu auftretenden Problemen viel herumgetüftelt und ausgiebig recherchiert. So wurde für die Implementierung des Bewegungsalgorithmus herumprobiert, auf YouTube und in diversen Foren recherchiert und die Hilfe einer Lehrkraft hinzugezogen. Allerdings waren alle gefundenen Lösungen unzureichend bzw. wurden als sehr komplex beschrieben. Am Ende gelang jedoch eine eigenständig erstellte, funktionierende Implementierung. Es ist naheliegend, dass die intensive Auseinandersetzung mit verschiedenen Lösungsansätzen im Zuge der Recherche wesentlich dazu beitrug. (LiliaGrinn 148, 156 & 169)

Weiters fußt auch die Verwendung des Typs *Enum*, zur Speicherung des Zustands der Spielfigur, und der Gebrauch einer statischen Variable auf eigenständiger Recherche (LiliaGrinn 66 & 170). Beide Konzepte waren den Lernenden zuvor unbekannt.

### 5.3.10. 5a Erstellung und Verarbeitung Digitaler Medien

Kategorie	Beschreibung
(G) Grafiken	Kompetenzen im Bereich Erstellung und Verarbeitung digitaler Grafiken
(A) Animationen	Kompetenzen im Bereich Erstellung und Steuerung von Animationen
(P) Partikeleffekte	Kompetenzen im Bereich Erstellung und Handhabung von Partikeleffekten

Tabella 23 Kategoriensystem der zweiten Reduktion - 5a Erstellung und Verarbeitung Digitaler Medien

	G	A	P
Cookie	5	1	0
Space Asteroids	0	0	6
Magische Muschel	2	0	0
Bouncyfant	0	2	0
Lilia Grinn	6	0	0

Tabella 24 Häufigkeiten Erstellung und Verarbeitung Digitaler Medien

In den Projekten *Cookie*, *Magische Muschel* und *Lilia Grinn* wurden eigene Grafiken für das Spiel erstellt. Für *Cookie* wurde eine Spielfigur per Hand in zwei Animationsstufen gezeichnet und gescannt. Die so erhaltene Grafik wurde dann in *Gimp*<sup>20</sup> nachbearbeitet (*Cookie* 23-24, 36 & 41-44). Die Erfahrungen vor dem Projekt mit *Gimp* beschränkten sich bei der Schülerin auf die Erstellung einfacher Fotocollagen (*Cookie* 40). In *Magische Muschel* wurden Bilder von Fischen aus dem

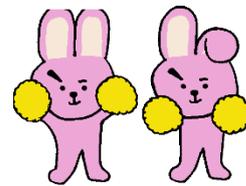


Abbildung 5 Die Spielfigur aus *Cookie*

Internet heruntergeladen und für die weitere Verwendung im Spiel mit *Gimp* freigestellt (*MagischeMuschel* 89-91). Für *Lilia Grinn* wurden alle im Spiel verwendeten Grafiken selbst erstellt. Dies umfasst die beiden Spielfiguren und fragmentierte Segmente von Bodenteilen, Fenstern und Wänden, aus welchen dann einzelne Levels beliebig zusammengesetzt werden können (*LiliaGrinn* 93-99). Die Grafiken wurden mit dem Onlinetool *Piskel*<sup>21</sup> in Pixel Art gezeichnet (*LiliaGrinn* 106-109). Dieser Stil, der sich an alten Spielen orientiert, die aufgrund technischer Limitierungen nur eine sehr geringe Auflösung hatten, wurde bewusst als Designentscheidung gewählt. Als Vorteil des Stils wurde auch genannt, dass die Zeichnung detailreicherer Bilder mehr Zeit in Anspruch nimmt (*LiliaGrinn* 110-112). Vorerfahrungen im Zeichnen von Pixel Art hatte in diesem Projekt niemand (*LiliaGrinn* 116).

<sup>20</sup> <https://www.gimp.org/>

<sup>21</sup> <https://www.piskelapp.com/>



Abbildung 6 Ein paar Grafiken aus Lilia Grinn

In *Bouncyfant* wurden simple Bild-für-Bild Animationen in Unity erstellt und in *Cookie* waren sie ebenfalls geplant. Aus (Bouncyfant 88 & 90) geht hervor, dass die Funktionsweise der Animationen verstanden wurde.

Für das Spiel *Space Asteroids* wurde ein Partikeleffekt<sup>22</sup> für das Triebwerk des Raumschiffs erzeugt. Die Funktionsweise der Partikeleffekte, die Bedeutung verschiedener Parameter, sowie die Bedeutung von Additivem Shader konnte erklärt werden. (SpaceAsteroids 64, 84, 88 & 92)

Im Bereich der Digitalen Medien zeigt sich besonders, dass das Gelernte zwischen den einzelnen Projektteams sehr stark variiert. Während beispielsweise in *Space Asteroids* fertige Grafiken und 3D-Modelle verwendet wurden und ein eigener Partikeleffekt erzeugt wurde, setzten sich die Lernenden in *Cookie* und *Lilia Grinn* mit dem Zeichnen von eigenen Bildern auseinander. Dabei unterscheiden sich auch hierbei sowohl der Stil als auch die eingesetzten Werkzeuge und Herangehensweisen wesentlich voneinander.

### 5.3.11. 5b Copyright & Lizenzierung

Diese Kategorie ist mit nur vier Codesegmenten die kleinste. In *Bouncyfant* und *Space Asteroids* wurden lediglich Assets verwendet, welche aus den zur Verfügung gestellten Tutorials stammen. Diese stehen mit CC0 bzw. CC-BY 3.0<sup>23</sup> klarerweise unter Lizenzen, die eine freie Nutzung ermöglichen, was in (Bouncyfant 84-85) bestätigt wird. Wobei hier anzumerken ist, dass lediglich mit „Ja“ auf die Frage, ob auf das Urheberrecht Rücksicht genommen wurde, geantwortet wurde. Daraus lässt sich nicht unbedingt schließen, dass sich eingehend mit den Lizenzen befasst wurde. Es könnte auch einfach sein, dass so viel Verständnis da ist, dass „Ja“ in dem Fall sicher die „richtige“ Antwort ist. In *Magische Muschel* wird dieselbe Frage mit „Nein“ beantwortet (MagischeMuschel 94-96) und in *Cookie* wird die Vermutung geäußert, dass es wohl rechtlich

---

<sup>22</sup> In einem Partikeleffekt werden einzelne Partikel (meist simple 2D-Grafiken) mit einem bestimmten Verhalten definiert. Von diesen einzelnen Partikeln werden dann viele Kopien erzeugt. Damit lassen sich zum Beispiel Effekte wie Wasserspritzer, Feuer, Rauch oder Explosionen darstellen. Mehr unter: <https://learn2progame.github.io/learn2proGrAME-Tutorial/0285-spaceasteroids/T02-particles1/>

<sup>23</sup> <https://creativecommons.org/>

gedeckt sei, da die Grafiken von der Lehrkraft empfohlen wurden (Cookie 29-30). Da für Lilia Grinn nur selbst erstellte Grafiken verwendet wurden, kam dieses Thema dort nicht zur Sprache.

In Cookie und Bouncyfant scheint zumindest ein rudimentäres Wissen darüber zu bestehen, dass die Lizenzierung von verwendeten Material Relevanz hat. Allerdings lässt sich aus den Interviews nicht schließen, dass hier tiefgehendes Verständnis vorhanden ist. Es ist auch wenig verwunderlich, dass sich die Jugendlichen in diesem Kontext wenig damit auseinandersetzen. Einerseits wird das Thema nur relevant, falls die fertigen Spiele veröffentlicht werden, andererseits trägt eine Beachtung des Urheberrechts nicht zum Spielerlebnis selbst bei. Daher liegt nahe, dass dieser Aspekt hier eher in den Hintergrund rückt.

### 5.3.12. Unityspezifisches Wissen und Kompetenzen

Kategorie	Beschreibung
(G) Grundlegendes	Kompetenzen im Umgang mit den grundlegenden Funktionen von Unity
(A) Unity Animator	Kompetenzen im Umgang mit dem Unity Animator
(PE) Partikeleffekte	Kompetenzen im Umgang mit Partikelsystemen in Unity
(PH) Physikengine	Kompetenzen im Umgang mit der Physikengine von Unity

Tabella 25 Kategoriensystem der zweiten Reduktion - Unityspezifisches Wissen und Kompetenzen

	G	A	PA	PH
Cookie	2	2	0	0
Space Asteroids	4	0	8	2
Magische Muschel	0	0	0	0
Bouncyfant	2	2	0	1
Lilia Grinn	5	0	0	1

Tabella 26 Häufigkeiten Unityspezifisches Wissen und Kompetenzen

In allen Projekten außer *Magische Muschel*<sup>24</sup> zeigen sich klare Anzeichen, dass Kompetenzen im Umgang mit den grundlegenden Funktionen von Unity vorhanden sind. Dies manifestiert sich zum einen in Erklärungen, darüber ...

- wie Assets dem Projekt hinzugefügt und Gameobjects erzeugt werden (Cookie 46 & 63).
- was zu beachten ist, wenn man Pixel Art Grafiken in Unity verwenden will (Lilia Grinn 117-118)
- dass public Variablen von C#-Skripten direkt in Unity gesetzt werden können (Space Asteroids 61-62)
- wie anhand von Vorlagen Klone von Spielobjekten erzeugt werden können (Lilia Grinn 122)

Zum anderen geht es auch direkt aus dem Videomaterial der Interviews hervor. Insbesondere bei *Bouncyfant* navigiert der Schüler im Verlauf des Interviews gekonnt durch den Unity Editor sowie Visual Studio und führt vor, was er währenddessen erklärt.

<sup>24</sup> Das Fehlen der Inhalte in *Magische Muschel* wird in Kapitel 5.2 angesprochen.

Sowohl in *Bouncyfant* als auch in *Lilia Grinn* wurde mit dem Umstand gerungen, dass sie zeitweise auf Rechnern arbeiteten, auf denen unterschiedliche Versionen von Unity installiert waren, weshalb sie Teile des Projekts in eine andere Version portieren mussten (Bouncyfant 95-96; LiliaGrinn 143). Daher mussten sich die Betroffenen zwangsweise auch mit versionsspezifischen Aspekten der Software befassen.

Weiters lernten zwei Teams mit dem Animator zur Steuerung der Animationen umzugehen und in *Space Asteroids* wurde ein Partikelsystem erstellt (Diese Aspekte wurden schon in 5.3.4 und 5.3.10 behandelt) sowie Bestehende aus der Sammlung der Unity-Standard-Assets in das Projekt eingebunden (SpaceAsteroids 12).

Ein wichtiger Aspekt für viele Spiele in Unity ist auch die Physikengine, mit der Spielobjekte relativ einfach mit physikalischen Eigenschaften und entsprechendem Verhalten ausgestattet werden können. Zentral dafür ist die *Rigidbody*-Komponente, welche in *Space Asteroids*, *Bouncyfant* und *Lilia Grinn* bewusst verwendet wurde (SpaceAsteroids 28-29; Bouncyfant 61; LiliaGrinn128)

## 6. Diskussion der Ergebnisse

### 6.1. Folgerungen

#### 6.1.1. Forschungsfrage F1

*F1: Welche Fertigkeiten und digitalen Kompetenzen werden durch die Entwicklung von Computerspielen in Unity im Informatikunterricht gefördert?*

In Hinblick auf die Forschungsfrage F1 lässt sich aus den Ergebnissen aus Kapitel 5 schließen, dass der Lernerfolg bzw. das Gebiet auf dem einer stattfand, zwischen den einzelnen Schüler\*innen variiert. Ein leicht unterschiedliches Niveau des Lernerfolgs unter den Schüler\*innen ist zwar durch unterschiedlichste Faktoren immer zu erwarten, dennoch gibt es Bereiche, die von manchen Schüler\*innen respektive deren Projekten abgedeckt sind, jedoch bei anderen keine Berührungspunkte aufweisen. In manchen Fällen ist dies trivialerweise aufgrund der Art des Projektes an sich der Fall, wie zum Beispiel im Hinblick auf Team Work. Es zeigte sich jedoch auch dadurch, dass einzelne, dennoch grundlegende Teilkonzepte, wie zum Beispiel Schleifen in *1b Kontrollstrukturen* oder Vererbung in *1c Objekte und Klassen*, nur in je einem Projekt vorkommen. Auch bei *5a Erstellung digitaler Medien* differierte die Art des Gelernten stark. Während in einem Projekt zahlreiche qualitativ hochwertige Grafiken mit einem zuvor unbekanntem Tool, in einem für den Grafiker neuen Stil erstellt wurden und hiermit eine Kompetenzförderung klar ersichtlich ist, wurden in zwei anderen Projekten gar keine Grafiken erstellt. Dieser Aspekt zeigte sich auch innerhalb der Teams, bedingt durch die Arbeitsteilung. In (LiliaGrinn 27-28) wurde angeführt, dass sich ein Schüler allein um die Erstellung dieser Grafiken kümmerte, da er dies gut könne und auch gerne tue.

*H3: Schüler\*innen setzen sich bei Arbeiten an einem Computerspielprojekt stärker mit Themen auseinander, auf die einer der folgenden Punkte zutrifft:*

*H3a: Das Thema fällt ihnen nicht schwer.*

*H3b: Sie interessieren sich stark für das Thema.*

*H3c: Das Thema ist für das Spiel sehr wichtig.*

Dies ist ein Indiz für die Hypothese H3. Da in Hinblick darauf sonst keine weiteren Hinweise vorliegen, kann diese Hypothese jedoch keineswegs bestätigt werden. Auch in anderen Bereichen sind große Unterschiede zwischen den Projekten ersichtlich, wie etwa in *4a Problemlösungsorientierung*, *4b Selbstbestimmtes Arbeiten* und den *Unity-spezifischen Kompetenzen*. Dies sind ganz klare Hinweise auf ein Zutreffen der Hypothese H2.

*H2: Der tatsächliche Lernerfolg in bestimmten Kompetenzbereichen variiert stark zwischen einzelnen Schüler\*innen.*

Dies ist im Hinblick auf den Unterricht keineswegs als negativ zu bewerten. Vielmehr kann man diesen Aspekt nutzen, um den Schüler\*innen die Möglichkeit zu geben, sich selbst zu entfalten und ihren individuellen Stärken und Interessen zu folgen und diese zu fördern. Es ist dennoch auch ein Hinweis darauf, dass Projektarbeiten dieser Art zum gezielten Vermitteln bestimmter Kompetenzen womöglich nicht geeignet sind, bzw. es einen von der Lehrkraft gesetzten Fokus bedarf.

In Bezug auf das Selbstbestimmte Arbeiten ist anzumerken, dass einzelne Schüler\*innen motiviert wurden, eigenständig zu recherchieren und sich Wissen anzueignen. Dies ist eine ganz wesentliche Schlüsselkompetenz und soll gefördert werden. Es zeigte sich jedoch, dass es hierbei zu Fehlkonzepten kommen kann, sodass eine Supervision des Gelernten durch die Lehrkraft notwendig ist. Wobei hier didaktisches Feingefühl gefordert ist, um zu verhindern, dass aus dem „eigenständigen“ Wissenserwerb eine asymmetrische Wissensvermittlung resultiert.

Die Analyse in Kapitel 5 hat gezeigt, dass in Summe alle behandelten Bereiche des Kompetenzmodells zum Großteil abgedeckt wurden. Die Dimensionen *2a Namensgebung* und *2b Programmierkonventionen & Code Stil* ist hier jedoch ausgenommen, da diese mit der Analyse, der im Zuge der Interviewbasierten Codereviews entstandenen Interviews, wie in Kapitel 4.2 und 5.3.5 beschrieben, nicht klar hervorgehen. Die Teilpunkte *4a Erstellung und Verarbeitung Digitaler Medien – 3D-Modelle & Sound*, *3a Problemlösungsorientierung – Debugging* und *4b Selbstbestimmtes Arbeiten – Verwendung der Programmdokumentation* kamen zwar nicht vor, diese stellen aber nur spezielle Teilpunkte der Kompetenzbereiche dar und könnten, beispielsweise durch Anreize der Lehrkraft, mit der Entwicklung von Computerspielen auch behandelt werden. In der Dimension *4b Copyright und Lizenzierung* zeigte sich, dass dieser Aspekt in den Projektarbeiten nur unzureichend behandelt wurde, was nahelegt, dies im Unterricht auf jeden Fall gesondert zu behandeln.

Zusammenfassend lässt sich anhand der Diskussion über die Einbettung der Spielentwicklung im Lehrplan in Kapitel 2.1 und den Ergebnissen der Interviewbasierten Codereviews schließen, dass durch die Computerspielentwicklung in Projektarbeiten ein Großteil der im Lehrplan geforderten Kompetenzen abgedeckt werden kann, wodurch sich die Hypothese H1 bestätigt.

*H1: Durch die Entwicklung von Computerspielen kann ein großer Teil der im Lehrplan geforderten Kompetenzbereiche abgedeckt werden.*

Bedingt durch den unterschiedlichen Fokus der beteiligten Schüler\*innen ist dies allerdings nicht notwendigerweise der Fall, bzw. scheint eine vollständige Abdeckung des Kompetenzmodells im Hinblick auf eine\*n einzelne\*n Schüler\*in als unwahrscheinlich.

Eine Möglichkeit zur besseren oder einheitlicheren Abdeckung wäre, die Spielentwicklung in einem weniger freien Setting durchzuführen bzw. die Projektarbeit besser zu strukturieren und zu lenken. Durch Vorgaben für die Projektarbeit, wie beispielsweise: „Es muss eine selbstgezeichnete Spielfigur in einem 2D-Spiel verwendet werden, welche animiert wird“, müssen sich die Schüler\*innen zwangsweise mit der Erstellung digitaler Grafiken und deren Animation beschäftigen. Mit solchen Maßnahmen kann ein Lernerfolg gezielt in ganz bestimmten Bereichen gefördert werden. Eine weitere Möglichkeit ist, den Schüler\*innen ein Grundgerüst eines Spielprojekts zur Verfügung zu stellen. Aufbauend auf diesem Gerüst sollen die Schüler\*innen dann bestimmte Features einbauen und so das Projekt erweitern, wie dies zum Beispiel bei dem Tutorial zu *Space Asteroids*<sup>25</sup> der Fall ist, welches im Rahmen von *Learn to proGrAME* entwickelt wurde. Der Vorteil hierbei ist, neben der Möglichkeit zur gezielten Förderung spezifischer Kompetenzbereiche, dass die Schüler\*innen schnell ein spielbares Spiel haben, ohne alles erst von Grund auf aufzubauen. Dies wirkt sich im Allgemeinen gut auf die Motivation und auch auf die Selbstwirksamkeitserfahrung aus.

Allerdings ist dem hinzuzufügen, dass die oben angesprochene unvollständige Abdeckung des Kompetenzmodells nicht zwangsweise als negativ aufgefasst werden muss. Gerade durch die Möglichkeit große Bereiche abzudecken, ohne dass dabei alles abgedeckt werden muss, bieten Projekte in der Spielentwicklung enormes Potential, die individuellen Stärken und Interessen der einzelnen Schüler\*innen zu fördern.

Es hängt hier schlussendlich davon ab, was im Zuge des Unterrichts erreicht werden soll. Ist das Ziel das Vermitteln, bzw. Erarbeiten von Kompetenzen in ganz bestimmten Bereichen, sind freie Projekte, wie sie im Zuge dieser Arbeit durchgeführt wurden, weniger geeignet, da aufgrund verschiedener Faktoren, einzelne Schüler\*innen mit diesen Bereichen mitunter gar nicht in Berührung kommen. Ist das Ziel hingegen die Förderung von individuellen Stärken, ist eine freie Projektarbeit mitunter mehr zu empfehlen.

### 6.1.2. Forschungsfrage F2

*F2: Inwieweit sind Interviewbasierte Codereviews als Mittel zur Kompetenzmessung geeignet?*

Gängiges Mittel zur Kompetenzmessung sind standardisierte Tests, wie sie zum Beispiel bei der Pisa-Testung zum Einsatz kommen. Häufige Kritik an derartigen Tests ist zum einen, dass diese nur punktuelle Momentaufnahmen sind und die Ergebnisse somit z.B. stark von der Tagesverfassung der getesteten Person abhängt, zum anderen kann auch bemängelt werden, dass durch das Anwenden einer vorgegebenen Testung für alle, jede\*r anhand derselben

---

<sup>25</sup><https://learn2progame.github.io/learn2proGrAME-Tutorial/0285-spaceasteroids/T01-spaceasteroids/>

Gesichtspunkte beurteilt wird, und somit individuelle Stärken und Schwächen eventuell verloren gehen. Brilliert beispielsweise jemand in einem bestimmten, getesteten Bereich besonders, so kann trotzdem nur erfasst werden, ob die gestellte Aufgabenstellung erfüllt werden kann. Außerdem können standardisierte Tests dazu führen, dass lediglich auf den Test hin gelernt wird. Das heißt der Lernprozess findet testzentriert statt und der eigentliche Wissens- bzw. Kompetenzerwerb rückt in den Hintergrund.

Auf Interviewbasierten Codereviews treffen die oben genannten Punkte nicht zu. Durch die Betrachtung des gesamten Projekts im Vorhinein und der Möglichkeit beim Interview nachzuhaken wird das Wissen der zu Überprüfenden nicht nur punktuell, sondern ganzheitlich betrachtet. Dadurch können auch besonders herausstechende Leistungen besser erfasst und auf die Überprüfung zugeschnittenes „Testwissen“ entlarvt werden.

Allerdings zeichnen sich standardisierte Tests durch ein hohes Maß an Objektivität aus, welches in den IBCRs definitiv weniger stark vorhanden ist. Standardisierte Tests sind auch in der Durchführung mit viel weniger Aufwand verbunden und besser für große Anzahlen skalierbar. Insbesondere die Durchführung der Kodierung durch mehrere Personen und die anschließende Diskussion sind sehr aufwändig. Lässt man diesen Schritt weg, würde die Objektivität und Reproduzierbarkeit der Messung stark leiden. Aus diesem Grund sind IBCRs für eine breite Anwendung eher ungeeignet.

Weiters ist die Wahl eines geeigneten Kompetenzmodells von zentraler Bedeutung. Ist keines vorhanden, so muss eines erarbeitet werden. Dies stellt ebenfalls einen aufwändigen Prozess dar, da die Eignung dieses Modells ebenfalls evaluiert werden sollte (vgl. Kramer, Hubwieser, & Brinda, 2016, S. 1). Auch das in dieser Arbeit verwendete Kompetenzmodell zur Objektorientierten Spielentwicklung bedarf einer solchen weiteren Evaluierung und gegebenenfalls einer anschließenden Revision.

Der Punkt 2 *Darstellung, Code Stil und Dateimanagement* des Kompetenzmodells wird in der momentanen Fassung der IBCRs auch nur unzureichend behandelt. Dies zeigt auf, dass es Kompetenzen gibt, welche durch IBCRs in der momentanen Ausführung nicht erfasst werden können. (Mehr dazu in Kapitel 6.3)

Zusammenfassend zeigen die Erfahrungen im Rahmen dieser Arbeit, dass die Interviewbasierten Codereviews einen sehr detaillierten Einblick in vorhandene Kompetenzen der getesteten Personen liefern können. Durch den hohen Aufwand sind diese allerdings für einen breiten Einsatz unpraktikabel. Eine Möglichkeit wäre eine vereinfachte Form, ähnlich der teilweise an Universitäten üblichen Abgabegespräche, zu erarbeiten.

## 6.2. Einschränkungen der Studie

Aufgrund der kleinen Stichprobengröße lassen sich die Ergebnisse aus 6.1.1 nur bedingt verallgemeinern. Es zeigte jedoch klar das Potential, große Bereiche durch Spielentwicklung abdecken zu können. Weiters ist festzuhalten, dass das verwendete Kompetenzmodell lediglich auf etablierten Modellen basiert, es selbst aber nicht validiert wurde.

Die in Kapitel 5 beschriebenen und in Kapitel 6.1.1 zusammengefassten Ergebnisse in Hinblick auf die Kompetenzförderung sind nicht zwangsweise nur auf die Arbeit in den Projekten zurückzuführen. Soweit möglich, wurde das Vorwissen berücksichtigt und eruiert, ob die erfassten Kompetenzen durch die Arbeit am Projekt gefördert oder entwickelt wurden. Dies ging jedoch in manchen Fällen nicht klar hervor.

Weiters ist anzumerken, dass die Art der Durchführung der Projektarbeiten einen wesentlichen Einfluss auf die Ergebnisse ebendieser und damit auch auf die Ergebnisse dieser Studie hat. Den Schüler\*innen wurden hier einerseits viele Freiheiten gewährt und keine genauen Vorgaben für ihre Projekte gemacht. Gleichzeitig war die Zeit im Unterricht sehr begrenzt, was auch in einem der Interviews bemängelt wurde (MagischeMuschel 32 & 107). Die Tatsache, dass ein Team auch in ihrer Freizeit maßgeblich an dem Projekt arbeitete, wirkte sich ebenso auf die Ergebnisse aus. Würde man das Projekt anders planen und bestimmte Aspekte im Projekt voraussetzen oder Aspekte des Projektmanagements miteinbringen, käme es im Zuge der Analyse vermutlich zu Abweichungen gegenüber den Ergebnissen dieser Arbeit.

## 6.3. Zukunftsausblick

Um die Interviewbasierten Codereviews hinsichtlich des Bereichs *2 Darstellung, Code Stil und Dateimanagement* des Kompetenzmodells zu vervollständigen, liegt eine detaillierte Analyse anhand der im Projekt zugrundeliegenden Quellcodes nahe. Shah et al. führten eine Studie durch, in der sie mithilfe von Qualitativer Inhaltsanalyse (Mayring, 2014) Programmierfehler von Studierenden im Hinblick auf Fehlkonzepte analysierten (Sha, Berges, & Hubwieser, 2017). Aufbauend darauf ließe sich auch eine Analyse der Quellcodes in die IBCRs integrieren, wodurch Kompetenzen im Bereich 2 des Kompetenzmodells und auch andere zusätzlich erfasst werden könnten.

Ein weiterer Schritt die IBCRs zu verbessern, wäre das zugrundeliegende Kompetenzmodell generell zu validieren und zu überarbeiten, sowie die IBCRs auch von anderen Forschenden durchführen und evaluieren zu lassen.

Hier bietet sich womöglich auch eine ganz andere Verwendung und Herangehensweise an das Werkzeug an. Im Zuge der Arbeit fiel auf, dass anhand der IBCRs immer wieder kleine Mängel

am Kompetenzmodell bemerkt wurden. Wodurch sich die Frage stellte, ob man das Ganze nicht umdrehen könnte und man das Konzept der IBCRs zur Validierung eines Kompetenzmodells einsetzen könnte.

Um weitere und bessere Ergebnisse zu erhalten, wäre es auf jeden Fall nötig eine zweite Iteration der Participatory Action Research durchzuführen. Im Zuge dieser sollten einerseits das Kompetenzmodell überarbeitet sowie der Interviewleitfaden angepasst werden, und andererseits die oben angesprochene Erweiterung der Interviewbasierten Codereviews für Kompetenzen des Bereichs *2 Darstellung, Code Stil und Dateimanagement* erfolgen. Darüber hinaus ist es nötig, die Projektarbeit im Unterricht besser zu strukturieren und dieser mehr Zeit einzuräumen. Hierzu könnten gängige Praktiken aus dem Projektmanagement verwendet werden, um die Schüler\*innen bei der Planung und Durchführung ihrer Arbeit zu unterstützen. Für eine bessere Vergleichbarkeit der Ergebnisse der einzelnen Projekte sollten vergleichbare Voraussetzungen, in Hinblick auf Teamgröße, Vorwissen, Form der Unterstützung durch Lehrkräfte und einem einheitlichen Zeitrahmen, geschaffen werden. Zusätzlich dazu wäre es vor allem nötig, die Anzahl der analysierten Projekte zu erhöhen.

## 7. Literaturverzeichnis

- Alexander, C. (1977). *A pattern language: towns, buildings, construction*. New York: Oxford university press.
- Baum, F., MacDougall, C., & Smith, D. (2006). Participatory action research. *Journal of epidemiology and community health, 60*, 854-857.
- Berges, M., Striewe, M., Shah, P., Goedicke, M., & Hubwieser, P. (2016). Towards deriving programming competencies from student errors. *2016 International Conference on Learning and Teaching in Computing and Engineering (LaTICE)* (S. 19-23). Mumbai: IEEE.
- Bundes- und Koordinationszentrum eEducation Austria. (kein Datum). *Digikomp12 - Das Kompetenzmodell*. Abgerufen am 1. Dez. 2019 von <https://digikomp.at/index.php?id=585&L=0>
- Chong, J., Plummer, R., Leifer, L., Klemmer, S., Eris, O., & Toye, G. (2005). Pair programming: When and why it works. *The 17th Workshop of the Psychology of Programming Interest Group PPIG17*, (S. 43-48).
- Comber, O. (2016). *Developing computer games in secondary schools : the influence of game development on computer science competencies*. Wien.
- Comber, O. (2019). *MC learn to proGrAME - Vorwort und Motivation*. Abgerufen am 3. Jan 2020 von <https://www4.lernplattform.schule.at/g16slsz/mod/page/view.php?id=5774>
- Dickson, P. E. (2015). Using unity to teach game development: When you've never written a game. *Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education* (S. 75-80). New York: Association for Computing Machinery.
- Greve, W., & Wentura, D. (1997). *Wissenschaftliche Beobachtung: eine Einführung*. Weinheim: Beltz.
- Kramer, M., Hubwieser, P., & Brinda, T. (2016). A competency structure model of object-oriented programming. *2016 International Conference on Learning and Teaching in Computing and Engineering (LaTICE)* (S. 1-8). Mumbai: IEEE.
- Landis, J. R., & G., K. G. (1977). The Measurement of Observer Agreement for Categorical Data. *Biometrics, 33*(1), S. 159-174.

- Mayring, P. (2014). *Qualitative content analysis: theoretical foundation, basic procedures and software solution*. Abgerufen am 8. Dez. 2019 von SSOAR: <https://nbn-resolving.org/urn:nbn:de:0168-ssoar-395173>
- Mayring, P., & Fenzl, T. (2014). Qualitative inhaltsanalyse. In N. Baur, & J. Blasius, *Handbuch Methoden der empirischen Sozialforschung* (S. 543-556). Wiesbaden: Springer.
- Molnar, M. (2020). *Video game programming – the better way to learn computer science concepts?* (Diplomarbeit) Wien: Universität Wien Fakultät für Informatik.
- Mor, Y., Winters, N., & Steven, W. (2010). *Participatory Pattern Workshops Resource Kit*. Abgerufen am 1. Dez. 2019 von <https://hal.archives-ouvertes.fr/hal-00593108>
- Sha, P., Berges, M., & Hubwieser, P. (2017). Qualitative Content Analysis of Programming Errors. *ICIET '17: Proceedings of the 5th International Conference on Information and Education Technology* (S. 161-166). New York: Association for Computing Machinery.
- Unity Technologies. (2020). *Spiele, mit Unity gemacht*. Abgerufen am 22. Jan 2020 von <https://unity3d.com/de/games-made-with-unity>
- Universität Wien. (2019). *Projektbeschreibung Learn to proGrAME*. Abgerufen am 3. Jan 2020 von <https://progame.at/projekt>
- Weinert, F. E. (2001). Vergleichende Leistungsmessung in Schulen-eine umstrittene Selbstverständlichkeit. In *Leistungsmessungen in schulen* (S. 17-32). Weinheim: Beltz.

## 8. Tabellen

Tabelle 1 Gegenüberstellung der Kompetenzmodelle (eigene Darstellung basierend auf (Kramer et al., 2016, S. 5) & den Learn to proGrAME Ressourcen) .....	22
Tabelle 2 Das Kompetenzmodell zur Objektorientierten Spielentwicklung .....	24
Tabelle 3 Hauptthemen im Interview mit zugehörigen Kompetenzbereichen.....	27
Tabelle 4 Interviewleitfaden .....	28
Tabelle 5 Übersicht über die Projekte.....	32
Tabelle 6 Häufigkeiten der kodierten Segmente der Oberkategorien je Projekt .....	33
Tabelle 7 Häufigkeiten der kodierten Segmente der Unterkategorien je Projekt .....	34
Tabelle 8 Kategoriensystem der zweiten Reduktion - 1a Datenstrukturen und Operatoren .....	35
Tabelle 9 Häufigkeiten - 1a Datenstrukturen und Operatoren.....	35
Tabelle 10 Kategoriensystem der zweiten Reduktion - 1b Kontrollstrukturen.....	38
Tabelle 11 Häufigkeiten - 1b Kontrollstrukturen.....	38
Tabelle 12 Kategoriensystem der zweiten Reduktion - 1c Objekte und Klassen .....	39
Tabelle 13 Häufigkeiten – 1c Objekte und Klassen .....	39
Tabelle 14 Kategoriensystem der zweiten Reduktion - 1d Zustandsbasierte Automaten.....	40
Tabelle 15 Häufigkeiten 1d Zustandsbasierte Automaten.....	40
Tabelle 16 Kategoriensystem der zweiten Reduktion - 3a Problemlösungsorientierung.....	42
Tabelle 17 Häufigkeiten 3a Problemlösungsorientierung.....	42
Tabelle 18 Ausschnitte aus Tabelle 6 und Tabelle 7.....	43
Tabelle 19 Kategoriensystem der zweiten Reduktion - 4a Team Work .....	44
Tabelle 20 Häufigkeiten 4a Team Work .....	44
Tabelle 21 Kategoriensystem der zweiten Reduktion - 4b Selbstbestimmtes Arbeiten.....	45
Tabelle 22 Häufigkeiten 4b Selbstbestimmtes Arbeiten.....	45
Tabelle 23 Kategoriensystem der zweiten Reduktion - 5a Erstellung und Verarbeitung Digitaler Medien .....	46
Tabelle 24 Häufigkeiten Erstellung und Verarbeitung Digitaler Medien .....	46
Tabelle 25 Kategoriensystem der zweiten Reduktion - Unityspezifisches Wissen und Kompetenzen .....	48
Tabelle 26 Häufigkeiten Unityspezifisches Wissen und Kompetenzen.....	48

## 9. Abbildungen

Abbildung 1 Ablauf Induktiver Kategorienbildung und Deduktiver Kategorienanwendung (Mayring und Brunner 2006 zitiert nach Mayring und Fenzl, 2014, S. 550) .....	13
Abbildung 2 Ablauf der Interviewbasierten Code Reviews.....	25
Abbildung 3 Ablaufdiagramm erste Analyse der Abgabe .....	26
Abbildung 4 Ablauf der Strukturierenden Inhaltsanalyse (Mayring, 2014, S. 96) .....	29
Abbildung 5 Die Spielfigur aus Cookie .....	46
Abbildung 6 Ein paar Grafiken aus Lilia Grinn.....	47

## 10. Anhang

### A. Kodierleitfaden<sup>26</sup>:

Mehrfachkodierungen sind erlaubt bzw. erwünscht. Beispielsweise tritt *3b Interpretieren von Code* häufig mit Kategorien aus *1 Programmierkonzepte* auf.

Bei Oberkategorien nur die Unterkategorien zum Kodieren verwenden, es sei denn, es findet sich etwas, das klar in die Oberkategorie, aber in keine der Unterkategorien fällt.

Kodiereinheit (eines der folgenden):

1. Ein ganzer Satz
2. Ein Hauptsatz (bei Hauptsatzreihen; und/oder/dann nicht mit kodieren)

156 B1: Ja ok, keine Ahnung, jedenfalls hat es dann einfach letztendlich funktioniert. Ich mein, ich habe halt verschiedene Ideen ausprobiert und ich hab auch im Internet ein bisschen Research gemacht, aber das Ganze was im Internet drin war, war irgendwie total kompliziert und schwer, und auch der Herr Professor hat uns eine Lösungs- einen Lösungsvorschlag gegeben, aber das war

3. Bei kurzen Antworten, bei welchen ihre Bedeutung ohne Frage unklar ist, die vorhergehende Frage einschließen.<sup>27</sup>

91 I: Ok ja. Und hier diese Variable, wo kann man die überall verwenden, wo hat die ihren Gültigkeitsbereich?  
92 B1: Nur in Public void changeGrid.

Kontexteinheit<sup>27</sup>:

Bei längeren semantisch zusammenhängenden Passagen können auch mehrere Sätze oder ein ganzer Absatz kodiert werden.

122 B1: Das sind quasi, soweit ich das richtig versteh, das sind so Vorlagen für Objekte in Unity, das heißt wir können ohne dass die jetzt direkt auf der Szene draufzuhaben ein Objekt haben das zum Beispiel ein Skript schon hat und eine Textur hat, einen Collider hat. Ohne dass es halt noch, ohne dass es auf der Szene drauf ist.

Die Auswertungseinheiten stellt die Menge der Antworten in den Interviewtranskriptionen dar, wobei, wie oben beschrieben, bei kurzen Antworten, die Frage miteinbezogen wird.

<sup>26</sup> Dies ist der Kodierleitfaden für die erste Kodierung durch den Erst- und Zweitkodierer.

<sup>27</sup> Im Zuge der gemeinsamen Diskussion fiel die Entscheidung die Kontexteinheit generell auf die Interviewfragen zu erweitern, sodass der Kontext der Aussagen aus den Antworten klarer ersichtlich ist.

## B. Kategorien zur Inhaltsanalyse

Kategorie	Definition	Ankerbeispiele
Unityspezifisches Wissen und Kompetenzen	Aussagen, die auf Vorhandensein oder nicht Vorhandensein von Unityspezifischen Kompetenzen oder Wissen über spezielle Unityfunktionen hindeuten. Insbesondere Partikeleffekte, Physikengine und Animator.	<ul style="list-style-type: none"> <li>• der Rigidbody regelt die physikalischen Eigenschaften des Spielobjekts</li> <li>• um die Figur zu bewegen setzt man die velocity des Rigidbodies</li> <li>• Partikeleffekte werden für Spezialeffekte wie zB Explosionen verwendet</li> <li>• Ein Partikelsystem erzeugt lauter kleine Bilder</li> <li>• im Animator habe ich festgelegt, dass die Figur vom Zustand gehen in den Zustand springen wechselt, wenn ...</li> <li>• Wenn man Grafiken in Unity importiert muss man aufpassen, dass ...</li> </ul>
<b>1 Programmierkonzepte</b>		
1a Datenstrukturen und Operatoren	Aussagen, die auf Vorhandensein oder nicht Vorhandensein von Wissen über oder Kompetenzen im Bereich Datenstrukturen und Operatoren hindeuten. Das heißt Variablen, Datentypen (Sowohl Primitive als auch Objekttypen), Zugriffsmodifikatoren, Gültigkeitsbereich von Variablen und Operatoren.	<ul style="list-style-type: none"> <li>• int ist ein Datentyp für Ganze Zahlen</li> <li>• AudioClip ist ein Datentyp für Sounds</li> <li>• in dieser Variable werden Namen gespeichert</li> <li>• der Unterschied zwischen int und float ist, wie die Zahlen gerundet werden</li> <li>• == schaut ob zwei Variablen gleich sind</li> <li>• private bedeutet, dass ...</li> <li>• hier wird ein Vektor durch ein float dividiert, das bedeutet, dass alle Einträge einzeln dividiert werden</li> </ul>
1b Kontrollstrukturen	Aussagen, die auf Vorhandensein oder nicht Vorhandensein von Wissen über oder Kompetenzen im Bereich Kontrollstrukturen hindeuten. Das heißt Verzweigungen (if, if-else) und Schleifen.	<ul style="list-style-type: none"> <li>• Das if überprüft ob, ... und macht dann ...</li> <li>• diese Schleife iteriert über alle ... und macht ...</li> </ul>
1c Objekte und Klassen	Aussagen, die auf Vorhandensein oder nicht Vorhandensein von Wissen über oder Kompetenzen im Bereich Objekte und Klassen hindeuten. Das heißt Typ, Vererbung, Methoden (Objektmethoden/statische Methoden), etc.	<ul style="list-style-type: none"> <li>• A ist eine Überklasse von B</li> <li>• X erbt von Y, das heißt ...</li> <li>• hier wird das Schlüsselwort static verwendet, weil</li> </ul>
Zustandsbasierte Automaten	Aussagen, die auf Vorhandensein oder nicht Vorhandensein von Wissen über oder Kompetenzen im Bereich Zustandsbasierte Automaten hindeuten. Das heißt insbesondere bei der Verwendung des Unity Animators und wenn über den	<ul style="list-style-type: none"> <li>• Sobald die Variable Geschwindigkeit größer als 0.1 ist, geht man in den Zustand gehen über</li> <li>• Zuerst wird überprüft, ob die Figur gerade gehend ist, wenn nein, dann ...</li> </ul>

	Zustand von Objekten gesprochen wird.	<ul style="list-style-type: none"> <li>• in dieser Variable wird der Zustand des Objekts gespeichert</li> <li>• diese Variable legt fest, ob die Figur gerade springt</li> </ul>
2bc Code Stil und Dateioorganisation	Aussagen, die auf Vorhandensein oder nicht Vorhandensein von Wissen über oder Kompetenzen im Bereich Coding Style Dateioorganisation und Management hindeuten. Ob zum Beispiel Tools wie Git verwendet wurden, ob Projektdateien strukturiert abgespeichert werden etc.	<ul style="list-style-type: none"> <li>• Um den Code besser lesbar zu machen, haben wir hier ...</li> <li>• Wir haben keine Tools zum gemeinsamen Arbeiten verwendet</li> <li>• Irgendwann haben wir so ein Chaos bei den Dateien gehabt, dass wir uns gar nicht mehr auskannten</li> </ul>
<b>3 Kognitive Prozesse</b>		
3a Problemlösungsorientierung	Aussagen, die auf Vorhandensein oder nicht Vorhandensein von Wissen über oder Kompetenzen im Bereich des Problemlösungsorientierten Arbeitens hindeuten. Dazu zählen das Testen von Code, DeBugging, sowie die Erarbeitung und Anwendung von Algorithmen.	<ul style="list-style-type: none"> <li>• wir wollten ein Gridbasiertes Movement programmieren und versuchten verschiedene Arten das zu machen</li> <li>• um den Fehler im Code zu finden machten wir ...</li> <li>• ... funktionierte aber irgendwie nicht, deshalb machten wir ...</li> <li>• Dieser Algorithmus macht ...</li> </ul>
3b Interpretieren von Code	Aussagen, die darauf hindeuten, dass die semantische Bedeutung von Code verstanden oder nicht verstanden wird.	<ul style="list-style-type: none"> <li>• das if macht, dass das Raumschiff nicht durch die Asteroiden fliegen kann</li> <li>• hier wird die Variable x um 1 erhöht</li> <li>• die Variable speed gibt die Geschwindigkeit, des Raumschiffs an</li> <li>• Das Ganze wird solange wiederholt, bis die maximale Anzahl erreicht ist.</li> </ul> <p>Ich habe keine Ahnung, was der Code hier bedeutet</p>
<b>Softskills und Methodenkompetenzen</b>		
4a Selbstbestimmtes Arbeiten	Aussagen die auf Kompetenzen in Bezug auf selbstbestimmtes Arbeiten und Lernen hindeuten. Das heißt die Verwendung von Dokumentation und eigenständiger Recherche.	<ul style="list-style-type: none"> <li>• Ich wollte, ein Gridbased Movement implementieren und habe mir dafür ein Tutorial im Internet angesehen.</li> <li>• Um den Fehler zu finden, versuchte ich ...</li> <li>• Wenn ich nicht weiterwusste suchte ich nicht im Internet nach Lösungen, sondern fragte einfach den Lehrer</li> <li>• ... hat irgendwie nicht funktioniert, dann stieß ich auf diesen Forumspost, wo jemand ein ähnliches Problem hatte</li> <li>• ich sah in der Unity Dokumentation nach, wie man einen Rigidbody verwendet</li> </ul>

4b Team Work	Aussagen die auf Kompetenzen in Bezug auf Teamwork hindeuten. Das heißt insbesondere ob und wie zusammengearbeitet wurde wie die Arbeit aufgeteilt wurde	<ul style="list-style-type: none"> <li>• Wir arbeiteten immer zu zweit am Computer</li> <li>• ich machte die Zeichnungen während sie sich um das Programmieren kümmerte</li> <li>• wir versuchten gemeinsam herauszufinden, wie wir ... umsetzen könnten</li> <li>• wir kamen uns dauernd in die Quere</li> <li>• Ich kann das nicht so gut, deshalb machte er das</li> </ul>
<b>Digitale Medien</b>		
5a Erstellung und Verarbeitung Digitaler Medien	Aussagen, die auf Vorhandensein oder nicht Vorhandensein von Wissen über oder Kompetenzen im Bereich der Erstellung und Aufbereitung digitaler Medien hindeuten. Insbesondere Grafiken und Animationen.	<ul style="list-style-type: none"> <li>• Die Grafiken habe ich alle selber erstellt</li> <li>• ich habe das Bild mit der Hand gezeichnet, eingescannt und dann noch nachbearbeitet</li> <li>• Bevor ich die Grafik verwenden konnte, musste ich sie zuerst bearbeiten</li> <li>• die Grafiken haben wir mit Gimp erstellt</li> <li>• den Soundeffekt, habe ich mit dem Handy aufgenommen und dann nachbearbeitet</li> <li>• Wir haben bewusst diesen Stil gewählt, weil ...</li> </ul>
5b Copyright und Lizenzierung	Aussagen, die auf Vorhandensein oder nicht Vorhandensein von Wissen über oder Kompetenzen im Bereich Copyright und Lizenzen in Bezug auf die Verwendung von Digitalen Medien hindeuten. (Grafiken, Soundeffekte, etc....)	<ul style="list-style-type: none"> <li>• Die Grafiken dürfen wir für das Spiel verwenden, weil sie unter der Creative Commons Lizenz stehen</li> <li>• um das Copyright haben wir uns keine Gedanken gemacht</li> <li>• die Assets sind frei verfügbar</li> <li>• Der Herr Professor hat uns die gegeben, daher denke ich dürfen wir sie verwenden</li> <li>• Das Bild hab ich von opengamearts.com, dort darf man alles verwenden</li> </ul>

## C. Transkriptionsrichtlinien

Transkription mit leichter Glättung:

- Umgangssprachliche Äußerungen werden an Schriftsprache angepasst
- Abgebrochene Sätze und Wörter werden notiert
- Äh öh ähm etc. werden nicht notiert
- Bestätigende Bemerkungen (Mhm, ja, gut, etc.) die während des Redeflusses der anderen Person eingeworfen werden, werden nicht explizit angegeben.
- Pausen werden notiert

Notationen:

B	Interviewte Person
I	InterviewerIn
(10)	Pause von 10 Sekunden. Pausen werden in runden Klammern und Sekundenangabe notiert.
I: Das ist also ein {Integer} B: {Integer} ja genau.	öffnende geschwungene Klammern werden an die Stelle des Textes gesetzt, an der überlappend eingesetzt wird, und schließende Klammern dort, wo das Simultansprechen endet.
<i>[Deutet auf den Bildschirm]</i>	Kursiver Text in eckigen Klammern sind Anmerkungen.
(...)	Unverständliche Äußerung
Bist du dir\	Unterbrechungen, wenn jemand ins Wort fällt, werden mit \ notiert.
Das ist eine Variable vom Typ Integ- nein float	Abgebrochene begonnene Wörter werden mit – beendet

## D. Kodierte Interviews

15.02.2020

Cookie

1/4

	1	I: Hast du vor diesem Projekt schon was mit Unity gemacht?
	2	B: Nein.
	3	I: Ok, hast du dich außerhalb von der Schule sch-, beschäftigst du dich mit Spieleprogrammierung, mit Spielentwicklung?
	4	B: Nein
	5	I: Spieldesign?
	6	B: Nicht wirklich.
	7	I: Nicht wirklich ok. Spielst du Computerspiele?
	8	B: Ja schon manchmal. Früher öfter, aber jetzt nicht mehr so.
	9	I: Mhm. Beschäftigst du dich außerhalb von der Schule mit Programmieren?
	10	B: Eigentlich nicht, nein.
	11	I: Eigentlich nicht ok. Ahm und in der Schule hast du programmieren gelernt, bevor du mit Unity angefangen hast?
	12	B: Ja wir haben mit, ich habe vergessen wie das Programm heißt.
	13	I: Visual Studio?
	14	B: Ja genau, haben wir programmiert, aber davor glaube ich auch nicht wirklich.
	15	I: Ok ja. Das heißt du hast davor schon die Grundlagen in C# gelernt. Ok, gut. Dann erklär einmal kurz worum es in deinem Spiel geht. Was, wenn das Spiel fertig ist, dabei rauskommen sollte.
	16	B: Also prinzipiell, als wir das begonnen haben zu machen, ging es ja darum, dass wir lernen wie man selbst Spielgrafiken einfügt, macht und zeichnet und ich hab mir eigentlich nicht wirklich überlegt wie das Spiel werden sollte, aber ich hätte es vermutlich so ähnlich aufgebaut wie Bouncyfant, den wir gemacht haben. Wo eben verschiedene Plattformen sind und das Ziel ist, einen Pilz der hier oben steht, einzusammeln und es aber auch Hindernisse gibt, im Sinne von zum Beispiel, dass Dinge runterfallen. Obwohl ich es versucht hätte, also ich weiß nicht ob ich es geschafft hätte, aber ich hätte vielleicht versucht Gegner einzubauen. Also auch welche die sich bewegen und so.
..4a Team Wo	17	I: Ok ja, spannend. Gut, du hast da jetzt allein gearbeitet oder?
	18	B: Ja- Nein nicht ganz. Weil <i>[eine Kollegin] hat mir geholfen. Oder bzw. mir zugeschaut.</i>
	19	I: Ok, aber ihr habt jetzt nicht irgendwie euch Arbeiten aufgeteilt, und gesagt: „ich mach den Teil, du machst den Teil“ oder so?
	20	B: Nein, ich hab schon alles gemacht. Aber wenn ich zum Beispiel Fragen hatte, habe ich immer zuerst sie gefragt und dann erst den Herrn Professor und manchmal konnte sie mir auch helfen.
	21	I: Ok, gut. Also habt euch quasi beraten und auch\
	22	B: Ja. Mhm.
..5a Erstellung und Ver.	23	I: Ok, gut. Du hast da, in deinem Projekt verwendest du verschiedene Grafiken. Ahm, hast du die alle selber gemacht oder?
	24	B: Nein, die Spielfigur Cookie habe selbst gemacht und den Hintergrund habe ich aus dem Internet.
	25	I: Ok, machen wir zuerst den Hintergrund. Weißt du noch wo du den her hast.
	26	B: Ja von hier. <i>[öffnet die Webseite gameart2d.com]</i>
	27	I: Mhm, von hier. Und ah kannst du diese Grafiken von dieser Seite einfach verwenden, oder musst du da bestimmte Dinge beachten?
	28	B: Also, man (2) muss, man kann sich die Grafiken raussuchen und dann downloaden und danach, sie sind dann in einer Zip-Datei und damit ich sie verwenden kann muss ich sie entzippen.
..5b Copyright & Lizenz	29	I: Ok, ja. Und, weißt du wie das rechtlich aussieht? Wie das mit dem Urheberrecht mit dieser Grafik ist, weil die hast ja nicht du gemacht. Es könnte ja sein, dass du da vielleicht eigentlich

..5b Copyright & Lizen		30	zahlen müsstest dafür.
		31	B: In diesem Fall glaube ich, dass es rechtlich abgesichert ist, weil ich glaube, dass der Herr Professor das- ich weiß es eigentlich nicht genau. {Dass es von der Uni ist oder so.}
		32	I: {Ja genau also} Nein es ist nicht von der Uni, geh noch einmal auf die Seite und da wenn du runterscrollst. Da bei License steht, (3) ja, wofür du die Sachen verwenden kannst. Die Free Asset License: Du kannst sie verwenden, das kostet nichts und du muss auch nicht angeben wo du es herhast.
		33	B: Mhm, ok.
		34	I: Das ist unter der Creative- unter einer Creative Commons Lizenz
		35	B: Ja.
		36	I: Ok. Gehe wir wieder da in Unity zurück. (3) Die Spielfigur, die hast du selbst gezeichnet, wie hast du das gemacht? Kannst du das erklären?
..5a Erstellung und Ver.		37	B: Zuerst habe ich sie einfach auf ein Blatt Papier gezeichnet und dann wurde es eingescannt und in einen Ordner gegeben, von dem ich sie mir dann in meinen eigenen Ordner geholt habe und dann haben wir sie auf Gimp weiterbearbeitet. Also es war damals noch nicht angemalt, sondern eben nur schwarz-weiß. Und dann habe ich zum Beispiel die Linien feiner gemacht, weil mir zum Beispiel die Form nicht ganz gefallen hat und angemalt und dann ja. Dann wars fertig.
		38	I: Ok ja, und dafür hast du Gimp verwendet?
		39	B: Ja.
..5a Erstellung und Ver.		40	I: Ok ja. Hast du davor schon Gimp verwendet, ah hast du da\
		41	B: Ja wir haben, oder ich bin mir nicht ganz sicher ob das mit Gimp war, aber ich glaub schon, haben wir in der dritten Klasse mit dem Herrn Professor {...} ein Bild bearbeitet also ich hab einen Wald gehabt und da hab ich dann Tiere hineingearbeitet.
..5a Erstellung und Ver.		42	I: Gut, dann hast du gesagt du willst die Figur auch animieren. Hast du da- Das sind zwei verschiedene Bilder oder?
		43	B: Ja, aber das sollte eigentlich nur eins sein. {...}
		44	I: {Es sollte eigentlich} nur eins jetzt da sein ok ja. Aber du hast zwei Animationsstufen sozusagen.
		45	B: Ja
		46	I: Ok und du willst sie nachher animieren? Weißt du wie das funktio- also was hast du da gemacht, wie hast du das versucht?
2bc Codestil & Dat ..5a Erstellung und Unityspezifisches V		47	B: Ähm, ich hab (2) Cookie eingef- von meinem Ordner sozusagen hier reingezogen. Und das waren aber da- das war dann noch ein Bild wo beide drauf waren, ich hab das dann zerschnitten. Und reingezogen und dann weiß ich nicht mehr.
		48	I: Und dann weißt du nicht mehr ok. Aber du hast hier diesen Animator.
		49	B: Ja
Unityspezifisches Wiss		50	[B öffnet den Animator]
		51	I: Was macht denn der?
		52	B: Er animiert.
Unityspezifisches Wi: ..1d Zustandsbasiert		53	I: Der animiert ok ja. Er kontrolliert quasi die Animation. Und man hat hier diese Felder und diese Pfeile dazwischen, weißt du was diese Felder und diese Pfeile machen.
		54	B: Also dieses Feld bedeutet, dass er steht und dieses dass er springt. Und die Pfeile machen sozusagen die Übergänge, also hier habe ich es zum Beispiel so gemacht, wenn oder wenn, also vom Stehen zum Springen, dass passiert, wenn es eine höhere Geschwindigkeit als 0.01 hat.
..3b Interpretieren von			I: Ok ja genau. Ahm also diese Felder nennt man quasi Zustände und die Übergänge hast schon genau, hast schon gesagt und hier hast du eine Bedingung eingefügt und wenn die Geschwindigkeit größer als 0.01 ist, dann soll man rüber in die andere Animationsstufe gehen. Ahm, warum hast du da größer als 0.01 und nicht größer als 0 gemacht? Weißt du das?

<p>..3b Interpretieren von </p>	<p>55</p>	B: Nein weiß ich nicht.
<p>..1b Kontrollstruktur </p>	<p>56</p>	I: Ok, gut. Wie würdest du das- Weißt du wie du das nachher einbauen könntest, dass das beim Programmieren, also diese, da steht Geschwindigkeit, was ist denn das da?
<p>Unityspezifisches Wissen </p>	<p>57</p>	B: Ein, eine Condition?
	<p>58</p>	I: Eine, ja das ganze is {eine Condition}
	<p>59</p>	B: {Eine Bedingung}
	<p>60</p>	I: Aber nur dieses Geschwindigkeit, da steht da Geschwindigkeit ist gleich 0.0
	<p>61</p>	B: Ein (3)
	<p>62</p>	I: Das ist eine Variable, die du über ein Programm dann setzen könntest. Ahm (3) Gut, welche Probleme sind im Laufe von der Arbeit so aufgetreten, was waren so Hürden, wo du irgendwie nicht weitergekommen bist, wo du länger dran arbeiten hast müssen.
	<p>63</p>	B: Ahm, also dadurch, dass ich hier eben noch nicht allzu viel gemacht habe, hier noch nicht so große Hürden. Aber zum Beispiel, als ich den Bouncyfant programmiert hab, wars zum Beispiel so, dass ich relativ große Probleme dabei hatte den Zweiten Elefanten einzufügen, weil ich den, also wir hatten ja das Tutorial. Und ich hab da zum Beispiel die Größe vom Elefanten anders gemacht, als es im Tutorial war, weil es für mich irgendwie, mir hat es besser gefallen. Und es war dann schwer sozusagen den andern Elefanten genauso zu machen und auch umzubenennen und das hat mir irgendwie Schwierigkeiten bereitet.
	<p>64</p>	I: Ok. Und wie hast du das dann lösen können, oder hast du es lösen können?
	<p>65</p>	B: Ja, also ich hab es lösen können indem. Also ich habe erstmal relativ viel rumprobiert, das hat dann eigentlich eh auch viel gebracht, aber bei Dingen wo ich wirklich nicht weiter wusste habe ich dann noch den Herrn Professor gefragt und der hat mir dann immer geholfen eigentlich.
<p>..4b Selbstbestimmtes / </p>	<p>66</p>	I: Mhm, ok. Hast du auch versucht selbst irgendwie, vielleicht im Internet zu recherchieren, wie du das Problem irgendwie\
	<p>67</p>	B: Nicht wirklich
	<p>68</p>	I: Ok, gut und du hast die Probleme nachher lösen können?
	<p>69</p>	B: Ja, also das Spiel funktioniert jetzt.
	<p>70</p>	I: Gut, dann sind wir eigentlich schon so ziemlich fertig. Was mich noch interessieren würde ist, wie findest du das grundsätzlich, wenn man im Inter- äh Informatikunterricht Spiele programmiert?
	<p>71</p>	B: Ich glaub, dass ist ein Thema, was die meisten mehr anspricht, als die meisten anderen Themen in der Informatik und deswegen glaub ich ist es eigentlich eine ziemlich gute Idee.
	<p>72</p>	I: Ok ja, und wie hast du das empfunden, also wenn man Spiele programmiert, dann muss man ja extrem viele verschiedene Sachen machen eigentlich. Du hast die Grafiken gezeichnet, später musst du das alles dann alles noch programmieren, man braucht Soundeffekte. Das sind ganz viele verschiedene Aspekte, die man alle benötigt und das ist eigentlich wahnsinnig komplex.
	<p>73</p>	B: Ja
	<p>74</p>	I: Und meinst du, dass der motivierende Effekt, dass es Computerspiele sind, da irgendwie überwiegt, oder dass es eher zu kompliziert ist, oder?
<p>..4b Selbstbestimmtes / </p>	<p>75</p>	B: Ahm, (2) naja, also nicht unbedingt, dass er überwiegt, aber er ist zumindest da. Und also zum Beispiel für mich wär es jetzt sehr schwer gewesen ohne das Tutorial das zu machen, weil es eben so viel Aspekte sind. Aber zum Beispiel mithilfe des Tutorials finde ich hat das eigentlich, das hat extrem viel geholfen. Und dann kam halt diese Motivation das ist ein Spiel, ich programmiere selbst ein Spiel. Und gleichzeitig gabs das, immer wenn ich mal nicht weiter weiß, kann ich aufs Tutorial schauen oder den Professor fragen. Deswegen hat das eigentlich nicht so ein großes Problem für mich dargestellt, dass es eben so viele Aspekte sind.
	<p>76</p>	I: Also du find- und grundsätzlich Spieleentwicklung im Informatikunterricht ist eine tolle Sache, oder?
	<p>77</p>	B: Ja schon. Ja find ich.

78 I: Gut dann vielen Dank.

- 1 I: Ok, wie viel Erfahrung hast du vor diesem Projekt mit Unity gehabt?
- 2 B: Gar keine.
- 3 I: Gar keine ok. Ahm hast du dich mit Programmieren beschäftigt bevor du mit Unity angefangen hast? {Hattest du davor schon Programmiererfahrung?}
- 4 B: {Hmm, nein noch nicht so viel,} also ich hab einen Freund, er wollte unbedingt auf die HTL Ottakring dort (..) Ähm so programmieren halt, Zweig machen und der hat sich ein bisschen beschäftigt und dann halt auch noch mit diesem Scratch halt, diese Klassiker, ansonsten nicht sehr viel.
- 5 I: Und in der Schule habt ihr davor schon programmiert, oder habt ihr direkt mit Unity angefangen?
- 6 B: In der Schule haben wir Scratch gemacht Dann haben wir jetzt mit C# angefangen, um halt die Grundlagen zu machen Matrix oder solche Sachen und dann sind wir zu Unity übergegangen.
- 7 I: Ok und beschäftigst du dich auch in deiner Freizeit mit Programmieren oder nur in der Schule
- 8 B: Hauptsächlich in der Schule und zwar aus dem Grund, weil ich zu Hause jetzt kein Unity habe aber ich kann mir gut vorstellen, dass ich in der Freizeit versuche so anfangen zu programmieren.
- 9 I: Ok, ja. Gut. Zeig mir einmal kurz ah worum es in dem Spiel geht und was du da gemacht hast.
- 10 B: An sich, ich muss das noch bewegen damit es besser funktioniert. *[B verschiebt ein Objekt in der Spielszene und startet anschließend das Spiel]* Aber an sich das Spiel ist relativ simpel. Es gibt zwei Figuren, die mit den Pfeiltasten und WASD gesteuert werden. Und man muss versuchen den Pilz zu erwischen und die Kisten und die Bälle versperren den Weg.
- 11 I: Ok ja. Du hast das Spiel anhand eines Tutorials erstellt?
- 12 B: Genau
- 13 I: Ok ja. Gut du hast es allein gemacht, also du hast nicht im Team gearbeitet oder?
- 14 B: Ja Grundsätzlich {schon}
- 15 I: {Grundsätzlich} Ok. Ahm, öffne einmal kurz bei den Scripts. Ich glaub wir beenden das Spiel einmal. Ahm, was nehmen wir? Nehmen wir das Müllscript.
- 16 B: Das Müll?
- 17 I: Ja. (5) Ahm während das jetzt öffnet machen wir mit was anderem weiter und zwar schauen wir uns an, du hast hier bei Sprites. Oder geht das grad nicht?
- 18 B: Geht grad nicht.
- 19 I: Ok, na dann warten wir einfach.
- 20 (30) *[Visual Studio lädt]*
- 21 I: So. Du hast hier zu Beginn, ah, dieser Teil vom Code. Was machst du da? Was passiert da?
- 22 *[I deutet auf am Anfang der Klasse initialisierte Objektvariablen]*
- 23 *[const int MAXANZAHLKISTEN = 30;]*
- 24 B: In dem Code leg ich fest wieviel Kugeln wieviel Kisten maximal im Spielfeld sein dürfen.
- 25 I: Und was macht diese Zeile da, die erste. Const int maxanzahlkisten = 30, was passiert da konkret im Programm?
- 26 B: Da leg ich fest, dass die Variable Maximalkisten nicht häufiger als 30 mal vorkommen soll, (.) die Variable Maximalkisten, also die Maximalkisten gleich 30 ist, dass nicht mehr als 30 generiert werden und das halt das Maximum ist.
- 27 I: Ok was heißt das int, das da davor steht?
- 28 B: Das erstellt mir eine Variable.
- 29 I: Mhm, und wofür steht das int, weißt du das?

..4a Team Wor 

..3b Interpretieren von   
..1a Datenstrukturen und C

..3b Interpretieren von   
..1a Datenstrukturen u

..3b Interpretieren von   
..1a Datenstrukturen u

..1a Datenstrukturen unc 

...1a Datenstrukturen unc 

...1a Datenstrukturen u 

...3b Interpretieren von 

...1a Datenstrukturen unc 

...3b Interpretieren von  
Unityspezifisches Wiss 

...1a Datenstrukturen unc 

...3b Interpretieren von C 

...3b Interpretieren von C 

...3b Interpretieren von C 

...3b Interpretieren von  
Unityspezifisches Wiss 

- 30 B: Integer
- 31 I: Mhm und was heißt also was ist das für ein Typ von einer Variable?
- 32 B: Das.
- 33 I: Was speicherst du in der Variable?
- 34 B: Ähm Namen, also (.)
- 35 I: Welcher Wert wird gespeichert in der Variable? (4) Also die Variable heißt maxanzahlkisten
- 36 B: Ja und da wird jetzt ein Zahlenwert gespeichert.
- 37 I: Genau ein Zahlenwert. Int ist für ganze Zahlen
- 38 B: Mhm.
- 39 I: Ahm, was ist da unten was sind das für Variablen das public gameobject Kiste?
- 40 B: Das sind die Objekte die ich hab, im Spiel zum Beispiel jetzt da [B wechselt von Visualstudio zu Unity] in dem Fall ist es jetzt die public gameobject Kugel ist aus dem Ordner die Kugel die hab ich da hinzugefügt. Ich glaube, ok das habe ich jetzt da nicht hinzugefügt. Und kiste ist halt diese Holzkiste in dem Fall da.
- 41 I: Genau also die, das sind quasi die
- 42 B: die Objekte
- 43 I: Objekte die nachher dann
- 44 B: hinzugefügt werden
- 45 I: hinzugefügt werden. Gut öffne noch einmal das Script Bouncyfant.
- 46 B: Bouncyfant, ich weißt gar nicht (..)
- 47 I: Ist eh offen passt.
- 48 B: Sind eh alle offen.
- 49 I: Ahm schauen wir nochmal darauf, da ist public string name. Was ist das string für ein Dateityp?
- 50 B: Das waren ähm das waren Namen der Zeiten
- 51 I: Bitte?
- 52 B: Das waren äh Dateien wo Namen gespeichert werden.
- 53 I: Aahm ok. Aah (2) Was ist das hier, das gehen? Protected void gehen, was ist das für ein Konstrukt im Programmcode?
- 54 B: Das zeigt zum Beispiel da, wenn ich das Spiel starte das ist grad so, dass ich mich überhaupt bewegen kann und dass die Animation des Gehens vorhanden ist, wie man ganz leicht sieht.
- 55 I: Ok ja, (3) Weißt du wofür dieses float h hier steht? [float h ist der Parameter der Methode]
- 56 B: Hmm, nein.
- 57 I: Ok, ahm. Was passiert in dieser Zeile?
- 58 (5)
- 59 B: Da wird festgestellt die Geschwindigkeit wie schnell der Elefant laufen darf oder laufen kann in dem Spiel. [B wechselt von Visualstudio zu Unity und führt vor] Das er zum Beispiel nicht in einer Sekunde durch den ganzen Bildschirm rennt, sondern in dem Fall jetzt nur so schnell rennt.
- 60 I: Mhm ok, geh noch einmal in das Script zurück und da steht Elefantenkörper.Velocity. Was ist denn Elefantenkörper?
- 61 B: Elefantenkörper ist der Elefant selber, also der mit dem Boxcollider ausgerichtete Elefant da.
- 62 I: Und Velocity steht wofür?
- 63 B: Geschwindigkeit

..1b Kontrollstruktur

..3b Interpretieren v  
..3a Problemlösungs

..1b Kontrollstruktur

..1b Kontrollstrukturen

..5b Copyright & Lizenz

..5b Copyright & Lizenz

Unityspezifisches Wi

..1d Zustandsbasiert

..5a Erstellung und v

..1d Zustandsbasiert

Unityspezifisches Wi

- 64 I: Geschwindigkeit genau. Also man setzt da die Geschwindigkeit vom Elefantenkörper. Geh nochmal in das Müllscript.
- 65 B: Ja
- 66 I: Und da steht void erzeugeKugel(), in dieser Methode hast du einen Algorithmus. Kannst du mir erklären was dieser Algorithmus, wie der funktioniert und was da passiert, was da gemacht wird.
- 67 B: Also, wenn die Anzahl der Kugeln, also wenn nicht die Maximalanzahl erreicht, indem Fall 45, erreicht wird, dann wird zufällig an einem Punkt, indem Fall (-13,13) von links nach rechts da oben und diese Kugel fällt dann runter durch die Schwerkraft. Und die Kugel wird dauerhaft erhöht.
- 68 I: Die Kugel wird dauerhaft\
- 69 B: Die Anzahl der Kugeln wird dauerhaft erhöht.
- 70 I: Die Anzahl der Kugeln wird erhöht, das heißt die Variable anzahlkugeln\
- 71 B: wird immer höher\
- 72 I: wird erhöht. Was hat das dann zur Folge?
- 73 B: Dass immer mehr Kugeln kommen.
- 74 I: Dass immer mehr Kugeln kommen, was ist wenn-, also kommen unendlich viele?
- 75 B: Nein das ist mit 45 festgelegt, wenn die if-Anweisung quasi überschritten wird. Das Anzahlkugeln nicht mehr kleiner als Maxanzahl ist, dann kommen keine Kugeln mehr, dann bleibts einfach stehen.
- 76 I: Aahm, das heißt da passiert einfach nix mehr.
- 77 B: Genau dann ists aus der Schleife draußen.
- 78 I: Ok, gut. Dann gehen wir einmal in Unity wieder zurück und zwar hast du bei Sprites, da hast du verschiedene Grafiken. Ahm hast du die selbst erstellt oder hast du die irgendwo her?
- 79 B: Die hab ich von der Internetseite, ich glaub Gametiles oder so hieß die glaub ich aber ich kann das schnell nachschauen im Internet. *[B minimiert Unity und öffnet Firefox]*
- 80 I: Ja es ist nicht ganz so wichtig woher du sie direkt hast. Was mich jetzt noch interessieren würde ist, darfst du die einfach so verwenden?
- 81 B: Das sind Sachen die frei zur Verfügung gestellt werden und wir haben extra nachgeschaut, dass sie kostenlos sind.
- 82 I: Ok ja.
- 83 B: Ich weiß jetzt nicht, irgendwie free\
- 84 I: Ja ich glaub freetilesets so, ja ist jetzt aber nicht so wichtig. Ahm aber du hast, also ihr habt das überprüft ob das Urheberrecht dabei nicht verletzt wird.
- 85 B: Ja genau.
- 86 I: Gut. Ahm, geh wieder in das Projekt zurück. Du hast vorher gesagt der Elefant bewegt sich, der wird animiert. Kannst du mir erklären wie das mit den Animationen funktioniert?
- 87 *[B öffnet den Animator]*
- 88 B: Ja und zwar hab ich da die Animationen festgelegt. Und zwar ist es so, dass an sich der Elefant bleibt stehen. Und dann hab ich festgelegt, *[B startet das Spiel]* wenn ich die Pfeiltaste drücke mit einer Verzögerung von 0,001 oder irgendsoeiner Verzögerung, bewegt sich der Elefant und wenn ich aufhöre die Taste zu drücken, geht er wieder zurück zum Elefant\_stehen und an sich ist der Ursprung das Elefant\_stehen, das bedeutet dass er ruht und sich nicht bewegt. Nur wenn ich Pfeiltasten drücke bewegt er sich, also kommt diese Bewegungsanimation.
- 89 I: Ok du hast, gehen wir noch in den Animator zurück. Du hast hier eben diese Zustände: Stehen und Gehen. Die anderen da werden eigentlich gar nicht verwendet. Zieh einmal diesen gehen\_0 auf die Seite a bissl. So, was bedeuten diese Pfeile dazwischen?
- 90 B: Der orange Pfeil bedeutet, dass es, wenn das Spiel startet automatisch zu Elefant\_stehen geht. Und diese kleinen Pfeile bedeuten, dass es gewechselt wird, wie gesagt, wenn ich die

..1d Zustandsbasiert  
 Unityspezifisches Wi



..5a Erstellung und Verarbe

..1d Zustandsbasierte Au



Unityspezifisches Wissen



..4a Team Wor

..4b Selbstbestimmtes



..4b Selbstbestimmtes Ai



Pfeiltasten drücke, wird der Pfeil verwendet, dass es zu gehen geht und wenn ich aufhör die Pfeiltasten zu drücken geht er wieder zu stehen.

- 91 I: Ja, ist es wirklich direkt die Pfeiltaste, die du drückst, oder wird das irgendwie über einen Umweg geregelt?
- 92 B: Es ist so, dass die Pfeiltaste aktiviert diesen Transitionpfeil und zwar hab ich das woanders eingestellt.
- 93 I: Mhm, dann würd mich aber interessieren, was ist denn da das springen und geschwindigkeit, bzw. hier diese Condition. Was, was besagt die?
- 94 B: Ähm, wenn die (10). Wenn die Geschwindigkeit des Elefanten größer als 1 ist, dann geht er in die Bewegungsformen. {Also in dem Fall 0,01}
- 95 I: {Die die die} Bedingung ist nicht direkt der Tastendruck, sondern eigentlich durch den Tastendruck wird die Geschwindigkeit erhöht und wenn die Geschwindigkeit größer ist, dann wird gewechselt. Gut. Ahm. Dann sind wir schon gleich fertig. Dann würde mich noch interessieren, was für Probleme sind aufgetreten, wie du an dem Projekt gearbeitet hast?
- 96 B: Ein Problem war, dass ich eine falsche Unityversion geö-, also dass sich eine falsche Unityversion geöffnet hat und ich einfach versucht hab das Spiel zu starten und dann ist das hängen geblieben. Dann andere Problem waren diese Plattformen, dass man die richtig hinkriegt.
- 97 I: Mhm, wenn du so ein Problem gehabt hast, wie bist du vorgegangen um das zu lösen?
- 98 B: Als erstes hab ich einmal versucht selbst zu lösen und versucht herauszufinden was das Problem war, dann hab ich zur Not, falls das nicht ging, hab ich Hilfe geholt, oder eigentlich wie gesagt versucht selbst, dann hab ich im Tutorial nachgeschaut was falsch sein könnte, dann hab ich Hilfe geholt, und dann wurde es größtenteils ähm auf die Seite(..)
- 99 I: Ok, hast du auch einmal im Internet recherchiert, wenn du ein Problem gehabt hast oder so?
- 100 B: Äh nein, also an sich mit den Tutorials oder mit den Videotutorials oder den anderen Tutorials schon, aber extra gegoogelt oder so nicht.
- 101 I: Gut dann würde mich jetzt nur noch zum Schluss interessieren, ah, was du davon haltest, dass wir im Unterricht eben Spielentwicklung gemacht haben. Findest du das gut, findest du das schlecht?
- 102 B: Ich find es gut, weil das gibt einen guten Einblick wie anstrengend teilweise es ist, oder wie schwer es ist so ein, in dem Fall jetzt einfaches Spiel zu programmieren. Für uns heutzutage wenn wir jetzt zum Beispiel andere Spiele nimmt, wo man keine Ahnung, Mobas oder andere kompliziertere Spiele, dass das viel länger dauert und dass das eigentlich viel schwieriger ist zu handhaben und das sicher auch eigene Assets verwendet werden müssen.
- 103 I: Ok ja. Und Spiel programmieren ist ja eigentlich wahnsinnig kompliziert. Man muss aus allem möglichen Bereichen eigentlich Skills haben und Dinge können. Findest du, dass das das Ganze unnötig kompliziert macht, dass es so wahnsinnig komplex ist, oder findest du, dass es eigentlich schon passt, weil\
- 104 B: Ich finde, dass manche Sachen zu komplex sind, dass man teilweise manche Sachen simpler machen könnte, aber es liegt vielleicht auch daran, dass es einfach speziell machen wollen und verschiedene Computersprachen verwenden und das halt zu Problemen führen kann.
- 105 I: Gut dann, vielen Dank!

- 1 I: Habt ihr bevor ihr mit diesem Unityprojekt gestartet habt, schon mit Unity gearbeitet, bzw. bevor in der Schule mit Unity angefangen habt?
- 2 B1: Bevor wir in der Schule angefangen haben? Weil wir haben so wie letztes Jahr auch noch ein Projekt gemacht und das war jetzt auch Vorwissen zu dem Projekt und vorher, also ich persönlich hatte auch vorher Erfahrung mit Unity. Noch vor dem ganzen Schulischen.
- 3 B2: Ich nicht, ich hatte nur das in der Schule.
- 4 I: Ok, ja und habt ihr euch auch schon vor der Spielentwicklung mit Unity mit Programmieren beschäftigt?
- 5 B1: {Ich schon}
- 6 B2: {Ja mit programmieren schon}
- 7 I: Und ja was habt ihr da so gemacht? In welchen Programmiersprachen und\
- 8 B1: Also ich habe mich meistens mit dem Webdevelopment beschäftigt. Eben mit PHP, HTML, ist jetzt keine Programmiersprache, aber halt CSS, HTML, Javascript, PHP, so Webdevelopment halt.
- 9 B2: Ja ich hab hauptsächlich Logistik und javas- ja auch mit Java und C++
- 10 I: Ok ahm und beschäftigt ihr euch auch außerhalb der Schule mit Programmieren?
- 11 B1: Ja ich schon.
- 12 B2: Ja.
- 13 I: Und mit Spielentwicklung.
- 14 B1: Auch, ich auch aber nicht so viel eigentlich.
- 15 B2: Abgesehen von dem Projekt jetzt, nein.
- 16 I: Ok, aber ihr programmiert auch in eurer Freizeit?
- 17 B1: Mhm {Ja}
- 18 B2: {Mhm}
- 19 I: In anderen Bereichen, ok gut. Ahm dann schauen wir uns einmal das Projekt an, das ihr gemacht habt und erklärt vielleicht kurz einmal worum geht es bei eurem Projekt, was wie, was für ein Spiel sollte rauskommen, wenn es fertig ist und spielt einmal vor was ihr schon habt.
- 20 B1: Ok, ah. Es soll ein Rätselspiel sein mit der Topdownperspektive. Ähm das Ganze soll auch ein grid-based Movement haben. Das heißt wir können uns nur anhand eines Rasters bewegen. Es soll auch verschiedene Blöcke und dann auch dementsprechend verschiedene Rätsel geben. Ja und was wir schon können, ist halt ein Level generieren *[startet das Spiel]*, was jetzt hier nicht ganz funktioniert, weil das wird hier einfach falsch generiert hier, komischerweise. Aber die Bewegung funktioniert schon einmal. Es ist nur ein bisschen klein. Kann man da irgendwie reinzoomen? Ja genau, man sieht das ziemlich genau. Wir haben noch keine Wände gemacht, also deshalb funktioniert das Ganze mit der Kollision noch gar nicht. Aber es ist eigentlich relativ gut mit dem Movement, da kann man sehen, egal wie sehr wir uns bewegen, man ist einfach genau in der anderen Figur drinnen. Genau. Ja. Das ist mittlerweile unser Stand.
- 21 I: Ok, sehr fein. Ihr arbeitet zu zweit im Team. Wie habt ihr euch die Arbeit aufgeteilt? (2) Oder habt ihr sie euch gar nicht aufgeteilt und alles gemeinsam gemacht?
- 22 B2: Nein wir haben uns, zuerst mal haben wir versucht so gemeinsam heranzugehen, aber dann sind wir draufgekommen, das hat recht lange dauert und wir sind uns da einander in die Quere quasi gekommen und haben alle immer eine verschiedene Vorstellung gehabt, was wir jetzt genau machen wollen und wie. Und irgendwann haben wir uns gedacht, so ok der eine macht jetzt die Bewegung und der andere macht das\
- 23 B1: Aber das war jetzt nicht wirklich geplant, das kam einfach spontan. Weil ich habe einfach angefangen an der Bewegung zu ar- also wir haben gemeinsam angefangen an der Bewegung zu arbeiten und dann bin ich einfach in die Richtung abgedriftet und du hast dann einfach das Level erstellen gemacht, das ist einfach spontan passiert. Oder?
- 24 B2: Ja

..4a Team Wor

<p>..4a Team Wor </p>	<p>25</p> <p>26</p>	<p>B1: Ich mein es gab halt diesen Moment wo du gesagt hast, du machst jetzt das Movement. Das ist einfach von selbst passiert.</p> <p>I: Also es ist quasi so selbst gelaufen, ihr habt es nicht irgendwie von vornherein strikt festgelegt, „ja ich programmiere die Bewegungen, ich kümmere mich um das und das“ und so weiter</p>
<p>..4a Team Wor </p>	<p>27</p> <p>28</p>	<p>B1: Also was wir gemacht haben, ist das ich gesagt hab, dass B2 die Grafik und Animation macht, weil ich mich damit überhaupt nicht auskenne und B2 mag das auch glaub ich.</p> <p>B2: Mhm.</p>
<p>..3a Problemlösungs C  ..4a Team Wor</p>	<p>29</p> <p>30</p>	<p>I: Ok. Wie habt ihr das Arbeiten im Team empfunden, war das, was für Vorteile hat das gehabt, was für Nachteile?</p> <p>B2: Ich meine, man hat ja voneinander Ideen bekommen. Man kann auch manchmal fragen, wie geht das nochmal oder wie genau stellen wir uns das vor.</p>
<p>..4a Team Wor </p>	<p>31</p> <p>32</p> <p>33</p> <p>34</p> <p>35</p> <p>36</p> <p>37</p> <p>38</p> <p>39</p> <p>40</p>	<p>B1: Und was ich vor allem cool finde eigentlich, ist wenn man zu zweit arbeitet, dann bleibt man auch nicht stehen, weil dann macht man ja immer weiter, weil der andere auch weiterarbeitet und er will auch weiterkommen und wenn man ganz alleine arbeitet, dann kann man, dann kanns sein, dass man einfach grade keine Lust hat. Und dann lässt man das Projekt auch so stehen, aber wenn man zu zweit das macht, dann lässt man das eher nicht stehen. Dann arbeitet man gemeinsam weiter.</p> <p>I: Also einerseits äh kommt man besser weiter weil man sich gegenseitig helfen kann?</p> <p>B1: Ja</p> <p>I: Andererseits motiviert man sich auch gegenseitig quasi?</p> <p>B1: Genau ja.</p> <p>I: Ok ahm, also das Teamarbeiten hat gut funktioniert. Und es ist positiv also, lieber im Team als alleine. Kann ich das so zusammenfassen?</p> <p>B1: Ja</p> <p>B2: Mhm</p> <p>B1: Ich würd schon sagen ja.</p> <p>I: Ok gut, habt ihr zur Zusammenarbeit irgendwelche bestimmten Tools benutzt oder habts ihr einfach, jeder schreibt sein Skript und dann fügt ihr es irgendwie zusammen oder\</p>
<p>2bc Codestil &amp; Dateio  ..4a Team Wor</p>	<p>41</p> <p>42</p> <p>43</p> <p>44</p> <p>45</p> <p>46</p>	<p>B2: Ja also wie wir es eben hauptsächlich gemacht haben, haben wir eben jeder sein Script geschrieben und dann haben wir es am Ende der Stunde irgendwie zusammengelegt und</p> <p>B1: Ja, also nicht, keine Tools wirklich.</p> <p>I: Ok, gut. So dann schauen wir uns einmal den Code an. Öffnet vielleicht einmal einfach ein Script.</p> <p>B2: Ja wollen wir zuerst den Spieler, oder?</p> <p>B1: Spieler? Also das ist erst das Gridding, aha ja.</p> <p>I: Ok, die ganzen Fehler können wir derweil ignorieren, das liegt daran, dass Visual Studio das Using Unity Engine nicht erkennt. Ihr verwendets da eine Enum, was ist denn das?</p>
<p>..1a Datenstrukturen unc </p>	<p>47</p>	<p>B2: Ahm das ist so ähnlich wie ein Array, aber eben mit so vorgesetzten Datentypen und später machen wir auch ein kleine, eine kleinere Variable dieses Enums indem wir dann speichern können ob es jetzt eben wie hier entweder Moving oder Standing ist. Und mit dem können wir dann auch einfach agieren, weil das Problem bei Arrays ist eben, man kann es zwar auch mit einem Array machen, aber das kann man entweder mit Integer machen, eben mit Null und Eins. Aber dann muss man sich immer merken, was jetzt welches ist und.</p>
<p>..1a Datenstrukturen unc </p>	<p>48</p>	<p>B1: Also ich würde es sogar eher als ein Boolean mit mehreren Wertmöglichkeiten quasi beschreiben. Ich mein ja wir haben es in unserem Fall quasi mit einem, also es hätte ein Array sein sollen, aber wir haben es mit einem Enum ersetzt, aber eigentlich finde ich ist es eigentlich ein Boolean mit mehreren Werten, mit mehreren Möglichkeiten.</p>
<p>..1d Zustandsbasierte Au </p>	<p>49</p> <p>50</p>	<p>I: Ok, wofür wirds ganz konkret verwendet in eurem Projekt? Was wird drin gespeichert?</p> <p>B1: Also um zu beschreiben, ob der Player sich eben gerade bewegt, gerade slided, gerade</p>

..1d Zustandsbasierte Au 	steht, oder halt eben nichts macht.
	51 I: Für die Zustände quasi?
..1d Zustandsbasierte Au 	52 B1: Genau, ja für die Zustände des Characters.
	53 I: Ok super. Ahm schauen wir uns vielleicht noch da kurz bei Grid an. Ihr habt da Int und Booleans, was ist das? Was ist ein Int?
..1a Datenstrukturen unc 	54 B2: Ja einfach, haben wir damit eine Variable erstellt, die einen bestimmten Nummernwert annehmen kann und hier haben wir zum Beispiel dem Level den Integer Eins gegeben.
..1a Datenstrukturen unc 	55 I: Aber was ist Int für ein Typ, was wird in Integer gespeichert?
	56 B1: Eine ganze Zahl
	57 I: Eine ganze Zahl ok. Gut, ja. Bleiben wir gleich da bei den Variablen, die am Anfang. Die Variablen die am Anfang stehen und oben deklariert werden, wo kann ich die überall verwenden?
..3b Interpretieren von ..1a Datenstrukturen u 	58 B1: eigentlich in der gesamten Klasse, wobei wir haben da auch public dazugestellt, daher nehme ich auch an, dass wir. Ist das eine public class? Ja ich nehme an wir können die, wenn wir quasi die Klasse ahm angreifen, also zugreifen auf die Klasse, dann können wir wahrscheinlich auch auf die Variablen zugreifen.
..1a Datenstrukturen unc 	59 I: Genau das stimmt ja. Aber die Variablen sind einmal grundsätzlich, die gehören\
	60 B1: der Klasse
	61 I: Zur Klasse beziehungsweise zu den einzelnen Objekten dieser Klasse. Dann hast du gesagt ja public bedeutet dass man darauf zugreifen kann von außen.
	62 B1: Mhm.
	63 I: Warum habt ihr das public gemacht? Hat das einen Grund oder?
..1a Datenstrukturen unc 	64 B2: Ahm eben auf manche dieser Variablen und auch später eine Liste müssen wir eben von verschiedenen Codes zugreifen und zum Beispiel hier deklarieren wir eben das Level und das sagt dann auch den anderen Scripts in welchem Level wir sind, was müssen wir generieren, auf was müssen wir Acht geben.
	65 I: Ok, ahm dann sehe ich, dass ihr da das Schlüsselwort Static verwendet habt. Was hat das für einen Grund?
..1c Objekt 	66 B2: Ahm das habe ich benützt damit wir eben direkt auf (5) also ganz ehrlich ich hab mir ein Tutorial im Internet angeschaut und da haben sie static benützt und ich hab zuerst ohne static probiert und dann hats {nicht funktioniert}
..4b Selbstbestimmt 	67 B1: {aber da hats nicht funktioniert}
..3a Problemlösungs 	68 B2: Dann habe ich aber static dazugefügt, und dann hat es auf einmal funktioniert und dann hab ich das einfach mal so hingenommen.
..1c Objekt 	69 I: Ok ja, da\
..1c Objekt 	70 B1: Ich glaub, ich nehme an das ist deshalb, weil wir jetzt ohne die Klasse oder ohne das Objekt zu instanzieren in anderen Klassen, dass wir einfach durch static halt direkt drauf zugreifen müssten, ohne ein neues Objekt zu instanzieren und das ist halt. Deshalb funktioniert das halt in anderen Klassen nicht.
	71 I: Genau. Dieses Static erzeugt eben statische Variablen, die sind, die gehören zur Klasse. Und das andere sind Objektvariablen die gehören zu einem bestimmten Objekt von der Klasse. Also wenn ich jetzt zum Beispiel mehrere Grids hätte, hätten alle denselben statischen Wert für Level, aber jedes einen eigenen Wert für Wall und so weiter. Gut. (3) Ihr verwendet hier in der Methode Generate dieses For. Was macht das?
..1b Kontrollstrukturen 	72 B1: Es iteriert durch die ganzen, in dem Fall Ypsilons, die von Minus Eins bis zu der Größe vom Grid gehen. Ich weiß grad nicht was für eine Zahl das ist, aber bis dahin.
..3b Interpretieren von 	73 I: Ok, und. Dieses If, was passiert da?
..1b Kontrollstrukturen 	74 B2: Ahm das haben wir eben, also wir haben quasi ein inneres Grid, wo eben unser Level aufgebaut ist und das ist im ersten Level eben ein drei mal drei Feld, das geht von null bis eben zwei. Weil da eben drei Werte, das geht von null bis zwei. Und in dieser For-Schleife tun wir nicht das was eben drinnen ist kreieren, sondern eben die Wand außen drumherum. Damit
..3b Interpretieren von 	

..1b Kontrollstrukturen ..3b Interpretieren von		eben der Player nicht aus dem Level hinausgehen kann. Und hier tun wir eben bei dem minus eins, also eben außerhalb und bei der Gridlänge, was eben beim ersten Level drei wäre, tun wir halt schauen obs eben außerhalb ist, und dann tun wir dort eine Wand hinzufügen.
75		I: Mhm, ok. Und was ist hier die Bedingung im If konkret? Ahm, da verwendets ihr verschiedene Operatoren, hier das doppelte Ist-Gleich macht was?
..1a Datenstrukturen ..1b Kontrollstruktur		76 B2: Damit tun wir eben schauen ob die Variable x jetzt den Wert eins\
..3b Interpretieren von		77 B1: {Minus eins.}
..1a Datenstrukturen		78 B2: {enthält,} minus eins enthält
..1a Datenstrukturen u ..1b Kontrollstrukturen		79 I: Ok ja, also man vergleicht damit?
..1a Datenstrukturen u ..1b Kontrollstrukturen		80 B2: Ja und diese Doppelstriche sind quasi so ein Oder.
..1a Datenstrukturen u ..3b Interpretieren von		81 I: Genau.
..1a Datenstrukturen unc		82 B1: Das heißt nur eine Bedingung in der If-Schleife muss richtig sein, damit das Ganze ausgeführt wird. Wobei eine oder mehrere, ich glaub das Funktioniert auch.
..3a Problemlösungs 2bc Codestil & Date ..1a Datenstrukturen		83 I: Ja
..1a Datenstrukturen unc		84 B2: Ja
..1a Datenstrukturen unc		85 I: Wie es konkret, also in vielen Programmiersprachen, teilweise wird es zum Beispiel auch, sobald die erste richtig ist, werden die anderen gar nicht mehr überprüft. Das hängt aber von der Sprache ab. Hier verwendets ihr Lists. Wofür, was macht ihr da, also was ist eine List?
..5a Erstellung und Verar		86 B2: Also eine Liste kann quasi auch verschiedene Datentypen, bei un- in unserem Fall jetzt Integers in sich eben speichern und hier tun wir dann halt eben auf diese Liste die wir in einem anderen, ahm Code eben kriert haben, tun wir, auf die tun wir eben hier zugreifen. Und da tun wir jetzt bestimmte Werte eben vergleichen.
2bc Codestil & Dateio		87 I: Ok, aahm. Hmm, wo ist denn das. Was habt ihr hier, was ist das für ein Datentyp?
		88 B1: Ein Array, ein ganz normales Zweidimensio- ja ganz normal, also ein zweidimensionales Array. {Nicht ganz normal}
		89 I: Genau, {Zweidimensionales Array}
		90 B1: Genau und wir haben das, wir haben uns dafür entschieden eben, weil da sieht man halt das Layer- das Levellayout auf den ersten Blick. Die Einsen sollen jetzt grade den Boden darstellen, aber wenn wir das jetzt durch irgendwas austauschen, dann müssten wir das Level nicht generieren um das sofort zu sehen wie das aussieht.
		91 I: Ok ja. Und hier diese Variable, wo kann man die überall verwenden, wo hat die ihren Gültigkeitsbereich?
		92 B1: Nur in Public void changeGrid.
		93 I: Genau. Gut, dann. (3) gehen wir weiter zu euren Assets. Ihr habt hier, ich mach das einmal größer. (10) Gut, welche Assets habt ihr für euer Spiel erstellt, bis jetzt?
		94 B1: Unsre Eigenen.
		95 I: Eure eigenen?
		96 B1: Ganz eigen, ja.
		97 I: Alles selbst gemacht?
		98 B1: Alles selbst gemacht.
		99 B2: Also wir haben zum Beispiel eben diese zwei Figuren. Die Figur Grün und die Figur Pink. Und dann haben wir uns auch schon einen Boden eben erstellt und an sich auch so ein paar mehr Sachen, die wir noch nicht benutzen, wie zum Beispiel ein Fenster oder eben eine Wandcorner, also eine Ecke. Aber die haben wir noch nicht reinprogrammiert. Und weiterhin haben wir hier auch eben unsere Szene und eben diese Skripte, die wir eben vorher angesprochen haben, haben wir hier drin gespeichert.
		100 B1: Ja was vielleicht nicht sehr vorbildlich ist von uns, ist dass wir das Ganze nicht sehr strukturiert haben und keine eigenen Ordner für die Skripte, keine eigenen Ordner für die Texturen halt haben, aber damit kommen wir mittlerweile auch noch klar, weil es ist ja noch nicht so viel.

		101	I: Ok, es geht noch aber\
		102	B1: Ja, wir sollen das demnächst machen ja. Wenn es mehr wird.
		103	I: Also es wär sinnvoll das zu ordnen, würdet ihr das sagen?
		104	B2: {Ja}
		105	B1: {Ja} auf jeden Fall
		106	I: Ok ja, ahm womit habt ihr die Sprites erstellt?
		107	B2: Ahm mit Piskel.
		108	I: Piskel ja.
2bc Codestil & Dateio		109	B2: Haben wir, eben mit der Internetseite haben wir eben online diese Sprites erstellt und haben sie sozusagen in unserem Assetsfolder gespeichert.
		110	I: Mhm. Ihr habt für die Grafiken da wie man sieht sehr niedrig auflösende Grafiken verwendet. Hat das einen Grund?
		111	B1: Das ist das Spieldesign ja. Wir haben ein Pixelartiges, Pixelart, ja genau. Ein ein {Pixelartspiel}
		112	B2: {Und auch} aus unserem Grund, ich mein so total detailreiche Figuren kosten Zeit zu machen und.
		113	B1: stimmt ja.
		114	B2: Das ist natürlich einfacher und man erkennt eben was eben gewollt ist.
		115	I: Mhm. Ahm, hast du davor schon Pixelart gezeichnet oder?
		116	B2: Ahm Pixelart speziell nicht, aber allgemein gezeichnet am Computer schon.
		117	I: Ok. Wenn man Pixelart in Unity verwendet, muss man da irgendwas, auf irgendwas aufpassen?
		118	B1: Auf alles. Dass die Rotation stimmt, dass die Position halbwegs stimmt, dass die Pixel na-also dass das Ganze gescheit skaliert ist, weil sonst passt die Größe nicht. Außerdem wenn jetzt Bilder in Unity importiert, dann werden die sofort mit einem Anti-Aliasing quasi versetzt und das zerstört natürlich das ganze Pixelartgefühl. Also man muss das auch ausschalten. Ja.
		119	I: Ok ja, also man muss sich, es reicht nicht, wenn man die Zeichnungen einfach in Unity reinzieht und verwendet. Sondern man muss sich noch näher damit beschäftigen wie das in Unity dann\
		120	B1: wenn man es dann raushat, dann gehts eigentlich relativ flott.
		121	I: ok ja. Ahm ihr habt da unten noch weitere Assets und zwar hier ein WallMetall, WandMetall, Boden. Das sind Prefabs, was sind das für Arten von Assets, was macht man damit?
		122	B1: Das sind quasi, soweit ich das richtig versteh, das sind so Vorlagen für Objekte in Unity, das heißt wir können ohne dass die jetzt direkt auf der Szene draufzuhaben ein Objekt haben das zum Beispiel ein Skript schon hat und eine Textur hat, einen Collider hat. Ohne dass es halt noch, ohne dass es auf der Szene drauf ist.
		123	I: Ok, ja. Genau. Ahm habt ihr Animationen in euer Spiel eingebaut?
		124	B2: Noch nicht.
		125	I: Noch nicht.
		126	B1: Aber die sind auf jeden Fall geplant.
		127	I: Sind geplant, ok. Ahm dann gehen wir noch einmal zu diesem Playerscript zurück. So ihr habt da beim Programmieren, einen Algorithmus entwickelt für die gridbasierte Move- also für die gridbasierte Bewegung. Wie seid ihr da vorgegangen, und was ist da, also wie funktioniert dieser Algorithmus?
		128	B1: Also eigentlich, er ist einfacher als man glaubt. Weil er schaut relativ kompliziert aus. Also zuerst schauen wir mal ob der State eben nicht moving ist. Das heißt wenn sich der Character grade nicht bewegt. Weil wir wollen ja nicht, dass wir mitten im Feld quasi uns weiterbewegen können. Das heißt wir schauen mal, wenn er mal steht. Das heißt wenn er
..5a Erstellung und Verar			
2bc Codestil & Dateio			
..5a Erstellung und Verar			
..5a Erstellung und Verar			
Unityspezifisches Wiss			
..5a Erstellung und Ver			
Unityspezifisches Wiss			
..1d Zustandsbasie			
..3b Interpretieren			
..3a Problemlösun			

...1b Kontrollstruktur	
...1d Zustandsbasie	
...1a Datenstruktur	
...1b Kontrollstrukt	
Unityspezifisches \	
...3b Interpretieren v	
...3a Problemlösungs	
...1a Datenstrukturen	
...1a Datenstruktur	
...1d Zustandsbasiert	

...1a Datenstrukturen unc	
---------------------------	--

...1a Datenstrukturen u	
...3b Interpretieren von Coc	

...1a Datenstrukturen u	
Unityspezifisches Wiss	

...1a Datenstrukturen unc	
---------------------------	--

...3a Problemlösungs Ori	
--------------------------	--

Unityspezifisches Wi	
2bc Codestil & Date	
...4b Selbstbestimmt	

steht, dann nehmen wir jetzt Inputs deshalb die Ganzen, die vier Else-Ifs, also drei Else-Ifs und ein If und die schauen halt eben, welche Taste grade gedrückt wurde. Bei Player Eins sind das eben die Pfeiltasten, bei Player Zwei die WASD-Tasten und ja. Wenn wir zum Beispiel auf den UpArrow klicken dann wird uns die Direction übergeben, also in dem Fall der Vektor nach oben, weil wir wollen ja mit der Pfeiltaste nach oben gehen. Und dann wird der Currentstate auf Moving gesetzt, das heißt wir können uns nicht weiterbewegen. Dann ahm, nachdem das quasi gesetzt wurde, das Enum auf Moving, dann geht natürlich gleich weiter in die Else-Abzweigung und hier wird dann direkt, das Objekt bewegt, also das Rigidbody wird dann direkt bewegt und zwar die Position. Es nimmt die jetzige Position, es fügt der jetzigen Position die ahm, die Richtung hinzu und die Geschwindigkeit mit der sie sich bewegt. Das heißt es dreht sich quasi, es dreht sich nicht aber, es schaut quasi in die Richtung wo sich das dann hindrehen soll und dann bewegt sich das solange weiter, bis der Counter, das ist einfach ein Integer, der null ist und dann wird er jedes Mal wenn er sich bewegt ahm einfach weiter, also nach oben gezählt. Und wenn der Counter gleich dem Speed ist, und das ist bei uns halt eben die Länge des Grids, das ergibt sich halt perfekt grade, dann bleibt das einfach stehen. Dann wird der Currentstate wieder zu Standing umgesetzt und der Counter wieder auf null, das heißt wir sind am Platz angekommen und dann bleibt er auch wirklich stehen.

129 I: Ok, also so stellts ihr quasi sicher, dass man nicht irgendwo dazwischen steckenbleibt.

130 B1: Genau, da weil das ist gar nicht gut.

131 I: Ok. (4) Ihr dividiert da Direction durch Speed, hier. Welche Datentypen haben die beiden Variablen?

132 [Anmerung: es geht hier um die Zeile: `rb.transform.position += direction/speed;`]

133 B1: Also Direction ist auf jeden Fall ein 3D Vektor und Speed ist soweit ich weiß ein Integer oder ein Float, aber ja eins von beiden.

134 I: Ok ja, können wir ja nachschauen, sollte irgendwo deklariert sein.

135 B1: Ja eh ganz oben wahrscheinlich. Ein Integer genau.

136 I: Speed ist ein Integer genau. Ahm jetzt ist es so, der Vektor3 in Unity ist ein Vektor der drei Floats speichert. Ahm normalerweise, kann man Datentypen nicht wirklich vermischen, wisst ihr wieso das da jetzt geht und was da genau passiert?

137 B1: Wir vermischen ja nicht wirklich Datentypen. Also wir dividieren ja nur jedes Float vom Vektor durch den Speed und das funktioniert in Unity zum Glück.

138 I: Genau ja, also das wird, der Vektor wird, also die\

139 B1: Genau die drei Floats einfach aufgeteilt auf drei Stück, separate Stück, die dann jeweils durch den Speed dividiert werden.

140 I: Genau ja, sie werden elementweise dividiert.

141 B1: Ansonsten würden wir das einfach so lösen, dass wir die Position von X und dann die Direction X durch Speed dividieren würden, dann die Position Y dann Direction Y durch Speed dividieren würden und einfach einzeln. Das würde dann auch funktionieren.

142 I: Gut. Ahm, so dann wollt ich mir noch was. Ahm, aber da haben wir eh auch schon darüber geredet. Gut, dann sind wir schon fast am Ende. Was mich jetzt noch interessieren würde ist, welche Probleme, Hürden sind im Laufe des Projekts aufgetaucht, also was, was waren so die Punkte wo es nicht ganz so einfach dahingegangen ist.

143 B1: Ja die Versionen von Unity (2) überhaupt nicht. Weil wir haben in der Schule sehr alte Versionen von Unity und zu Hause, wo ich das installiert hab, nehme ich natürlich die Neueste, weil die halt wahrscheinlich die Beste ist, mit den wenigsten Bugs und den meisten Features. Und das rüber quasi nehmen übernehmen von der alten Version in die Neue, das hat bei mir sehr viele Probleme gemacht, das heißt ich musste wirklich ein neues Projekt erstellen und die Assets einzeln rüberkopieren und dann hat es endlich funktioniert. Aber halt von selbst ging das gar nicht.

144 I: Ok, also es waren vor allem auch einmal strukturelle Probleme quasi, die weil halt in der Schule die Versionen veraltet sind und das hat euch quasi im Arbeitsfluss behindert.

145 B1: Ja

146 B2: Weiterhin haben wir auch am Anfang irgendwie ewig nachgedacht, was genau wir eben

		machen wollten und wir haben glaub ich auch 10.000 mal den Plan geändert, nochmal komplett neu gestartet, weil irgendwie so wie wir es uns vorgestellt haben, es eben nicht funktioniert hat und
		147 I: Ok ja.
..4b Selbstbestimmtes ..3a Problemlösungs C		148 B1: Außerdem, wir hatten noch große Probleme mit dem, naja Probleme. Wir hatten halt keine Idee wie wir das Movement jetzt machen können. Hat auch länger gedauert. Aber wir haben es dann letztendlich irgendwie geschafft. Und das war dann auch irgendwie spontan, mit einem ganz frischen Kopf, ganz von vorne begonnen, dann hats auf einmal funktioniert. Also dann hatten wir auch gleich die Idee und dann war die Umsetzung auch richtig. Und außerdem mit dem Level erstellen, mit dem Array, da haben wir auch Probleme gehabt mit dem Array, oder? Oder hat eh alles funktioniert.
		149 I: Bleiben wir noch kurz bei dem Movement. Ahm, ihr wolltet so ein gridbasiertes Movement machen und das war nicht ganz klar wie das von vornherein geht. Wie seid ihr dann vorgegangen, wie habt ihr versucht, das zu lösen? Ahm habt ihr, einfach nur gegrübelt, habt ihr versucht gemeinsam draufzukommen, habt ihr euch Hilfe geholt, habt ihr recherchiert? Oder alles?
..3a Problemlösungs Ori		150 B1: Ich glaub wir haben einfach angefangen drauf los zu programmieren, oder? Oder haben wir\
		151 B2: Ich mein, das hast jetzt hauptsächlich du das, die Bewegung jetzt gemacht hast.
		152 B1: Ja schon, aber ganz am Anfang haben wir das noch gemeinsam gemacht, oder?
		153 B2: Hast du glaub ich auch\
		154 B1: Habe ich auch alleine gemacht ok ja.
		155 B2: Bei der Bewegung hatte ich jetzt nicht so viel damit zu tun.
..3a Problemlösungs C ..4b Selbstbestimmtes		156 B1: Ja ok, keine Ahnung, jedenfalls hat es dann einfach letztendlich funktioniert. Ich mein, ich habe halt verschiedene Ideen ausprobiert und ich hab auch im Internet ein bisschen Research gemacht, aber das Ganze was im Internet drin war, war irgendwie total kompliziert und schwer, und auch der Herr Professor hat uns eine Lösu- einen Lösungsvorschlag gegeben, aber das war auch irgendwie so kompliziert und keine Ahnung, ich hab dann den einfachsten Weg mit dem wenigsten quasi Arbeitsaufwand gesucht und habe ihn einfach gefunden ja. Ist jetzt vielleicht nicht das schönste Skript und es ist vielleicht nicht fehlerfrei, aber es funktioniert.
		157 I: Aber du hast, hast du durchbeißen müssen, war das mühsam, oder?
		158 B1: Ja anfangs war das schon mühsam, aber dann gings eh.
		159 I: So das war das Erste, und das andere war mit diesem, wie ihr das da, die Tiles quasi erzeugt und in die richtige Reihenfolge bringt. Was war da das Problem? (2) Also wie? (3)
		160 B2: [fragt B1] Welche Tiles in die richtige Reihenfolge?
		161 B1: Nein nein, also das Level erstellen.
		162 I: Die Bodenplatten legen quasi.
		163 B1: Oder hats eh von Anfang an geklappt und zack fertig?
..3a Problemlösun Unityspezifisches \		164 B2: Naja, nein. Das Prinzip hat an sich geklappt. Aber zuerst waren unsere Bodenplatten eben viel zu klein. Und da mussten wir eben auch mit den Sprites so ein bisschen arbeiten, ein bisschen umändern.\
..4a Team Wor		165 B1: Das war eben ein großes Problem, weil man kann in Unity keine definitven Größen einstellen. Weiß ich nicht warum. Aber wir haben, also Felix hat dann das gemacht, dass er die Texturen einfach runterskaliert hat. Jetzt haben alle 22 mal 22 Pixel. Warum auch immer. Aber ja wir sind jetzt originell und haben jetzt eine eigene Pixelgröße. Und ja, jetzt haben die halt ein gleiche Größe.
..5a Erstellung und		166 I: Ok ja. Ahm und wie wie habt ihr das alles selber gelöst?
		167 B2: Mit Unterstützung von Ihnen und eben dem Herrn Professor
		168 I: Ihr habt vorher noch gemeint bei den, der Recherche habt ihr gegoogelt. Weißt du noch wo du da gegoogelt hast, ahm und auf welchen Seiten?
..4b Selbstbestimmtes 3a Problemlösungs C		169 B1: Überall. Ich habe auf Youtube gesucht. Weil ich find auf Youtube, da erklären das die Leute

..4b Selbstbestimmtes  
 ..3a Problemlösungs C



..4b Selbstbestimmtes A1



..1a Datenstrukturen u  
 ..1b Kontrollstrukturen



..5a Erstellung und Verar



- auch und ich finde, wenn die das wirklich zeigen mit einem Video, dann ist das viel klarer als wenn ich einfach einen langen Text lese der das erklärt. Aber ich habe eben zuerst eben nach Texten gesucht, weil ich dachte, dass kann ja nur sehr einfach sein, weil die ganzen Pokemonspiele und alles, die sind ja so lang her, das heißt die Lösung muss ja eigentlich so extrem simpel sein. Ist sie aber nicht tatsächlich. Und ja da waren die Texte wirklich sehr kompliziert und sehr lang und teils haben sie gar nicht funktioniert obwohl ich sie einfach nur überkopierte hab. Und dann die Videos waren auch sehr kompliziert, also ja eigentlich Youtube und so verschiedene Seiten.
- 170 B2: Hmm ja, und eben die Listen und das Enum, alles hab ich eben auch von Youtube Videos gesehen und.
- 171 I: Ok, ja. Dann kommen wir jetzt zum Ende. Jetzt würde mich noch interessieren, ahm findet ihr ist Spieleprogrammierung im Informatikunterricht eine gute Sache? Also es kommen ja wahnsinnig viele Bereiche aus der Informatik vor, findet ihr das sinnvoll das über Spieleprogrammierung zu machen um diese vielleicht zu verknüpfen oder wärs vielleicht eher besser jetzt Bildbearbeitung separat zu handhaben, Programmierung extra zu machen, sodass man quasi ahm nicht alles auf einmal machen muss, sondern sich den einzelnen Dingen einzeln widmen kann und sich so auf das Wesentliche konzentrieren kann oder ist es eigentlich besser das als Ganzes zu machen und ja?
- 172 B1: Also ich würde auf jeden Fall die Grundlagen des Programmierens als Einzelnes behandeln, also zum Beispiel, was sind Forschleifen, was sind Verzweigungen, was sind Datentypen? Das würde ich auf jeden Fall gar nicht erst mit Unity anfangen, weil das ist viel zu kompliziert. Und erst dann wenn man sich vielleicht auskennt, wie funktioniert der Computer jetzt eigentlich, geht er die Zeile nach der Zeile ab oder springt er jetzt hin und her oder was. Wie funktioniert das genau? Ich würde das auf jeden Fall als was eigenes quasi abhandeln und dann erst in Unity einsteigen. Und ich find das auch gut, dass wir zum Beispiel dieses Bildbearbeitungs-, dass Bildbearbeitung das große Thema auch eigentlich direkt in die Spieleentwicklung haben, weil sonst wäre das so. Sonst ist das ganze Bildbearbeitungszeugs ohne Grund wirklich, ohne Ziel. Und beim Spiel da weiß ich wirklich, ich brauch die Textur, damit das erfüllt ist, damit das funktioniert. Und dann hat man noch die Motivation weiter zu gehen und dann hat man noch die Motivation weiterzuforschen wie das funktioniert. Also ich find das zeigt schon viel Sinn.
- 173 I: Ok und findet ihr das motivierend, wenn man eigene Spiele entwickelt um diese Dinge dann weiter zu vertiefen zu erlernen, oder ist es eher mühsam, weil man eben so viel auf einmal machen muss?
- 174 B2: Ich fände es, persönlich find ich es besser so, weil man da quasi auch seine eigene Vorstellung davon hat und man hat in dem Sinn auch mehr Freiheiten wie man jetzt genau sein eigenes Projekt eben machen will und bei Tutorials ist eben das Problem, dass dieser eine Weg vorgeschlagen wird und diesen einen Weg gehst du und du weißt nicht, ob es jetzt irgendwelche anderen Möglichkeiten gibt etwas zu machen.
- 175 B1: Mhm
- 176 B2: Und ich glaub sich daraus selbst ein bisschen herumschauen was für Möglichkeiten es gibt etwas zu erreichen ist recht gut.
- 177 B1: Aber ich würde auch nicht alles durch die Spielentwicklung machen, weil ich find, man sollte das schon machen. Aber ich finde manche Themen brauchen schon einen eigenen Platz und vielleicht nicht in der Spielentwicklung. Weil das ist vielleicht auch mühsam manchmal, weil da irgendwas nicht funktioniert, weil da Unity irgendwie herumpackt und dann kann man jetzt was anderes nicht machen, was eigentlich gar nichts mit Unity zu tun hat.
- 178 I: Ok, gut, dann vielen Dank!

1	<i>[Leider ist bei diesem Interview die Aufnahmequalität sehr schlecht, daher gibt es einjiae unverständliche Stellen, welche durch (...) gekennzeichnet sind.]</i>
2	I: Gut, ahm. Als erstes würde mich einmal interessieren, was ihr, was eure Vorerfahrungen sind mit Unity und so weiter. Ahm, habt ihr schon vor diesem Projekt mit Unity gearbeitet?
3	Alle Ja.
4	I: Ja, was habt ihr davor gemacht?
5	B3: Wir haben so im Unterricht halt so ein bisschen durchgenommen, also halt so angeschaut und ja.
6	B2: Wir haben auch Spiele gemacht und eben Code angeschaut was, das bedeutet.
7	I: Ok, ahm. Habt ihr, wie lange schon davor im Unterricht, also in diesem Jahr oder schon davor?
8	B3: Ja, nein, also nur in diesem Jahr.
9	I: Nur in diesem Jahr ok. Und habt ihr auch bevor ihr mit Unity angefangen habt, schon mit Programmieren beschäftigt extra, oder nur mit Unity?
10	B2: Scratch halt.
11	B3: {Scratch ja}.
12	I: {Scratch mhm} ja.
13	B2: Aber sonst {nix}
14	B3: {sonst nix}
15	I: Ok, also C# zum Beispiel, habt ihr nicht vorher schon extra gemacht?
16	B3: Doch C# hatten wir auch
17	B1: Haben wir das nicht (...)?
18	B3: Ich weiß nicht. (...) Wir hatten irgendwas mit C# gemacht ja.
19	I: Ok und beschäftigt ihr euch auch außerhalb der Schule mit Spielentwicklung oder Programmierung?
20	Alle: Nein <i>[schütteln den Kopf]</i>
21	I: Ok, gut. Und ihr seid jetzt in der 5. Klasse?
22	B3: Ja
23	I: So, dann erklärt mir mal kurz worum es in eurem Spiel geht, und was die Spielidee ist. Wir können es jetzt leider nicht vorführen, weil das mit der Version nicht funktioniert.
24	B3: Also, wir haben halt zwei Figuren. Und die sind in einer Unterwasserwelt und also es sind zwei Fische und ähm die. Es geht halt darum, dass die so eine Schatzkiste, so eine Muschel, dass die eine Muschel finden. Und ähm von oben sind halt so Seeigel, die runterfallen und sie müssen denen halt ausweichen. (.)
25	I: Ok, cool.
26	B2: Ja.
27	I: Irgendwas?
28	B2: Nein.
29	I: Ok, passt, gut. Ihr habt in einem Team gearbeitet, zu dritt?
30	B3: Ja
31	I: Wie war das für euch? Habt ihr die Aufgaben klar aufgeteilt? Habt ihr gesagt: „Ok, ich mach die Bilder“ „ich mach die Animation“ oder?
32	B3: Also wir hatten von Anfang an so vor, dass immer zwei am Computer gearbeitet haben und eine hat dann halt Protokoll geschrieben. Dann halt abwechseln. Ja es war halt schwer, weil wir ziemlich. Also wir hatten irgendwie zu wenig Zeit dafür. Wir hätten ein bisschen mehr Zeit gebraucht, damit wir das wirklich halt machen können.
33	B2: Weil wir hätten auch (.) geplant gehabt mit den Seeigeln, aber die haben wir nicht gemacht.

..4a Team Wor 

..4a Team Wor 

		34	I: Ok ja.
		35	B3: Das ging sich nicht aus.
		36	I: Ja, ihr habt gesagt Protokoll schreiben. Was ist das für ein Protokoll?
		37	B2: Da musste man erklären um was es geht in diesem Spiel. Was wir getan haben in der Gruppe und was wir dazugelernt haben.
		38	I: Ok, dadurch dass ihr in einem Team gearbeitet habt, was für Vor- und Nachteile hat das gehabt?
		39	B3: Also man hat das zum Beispiel beim ähm bei diesem, wie hat das geheißen, mit dem (...) <i>[zeichnet mit den Händen ein Quadrat in die Luft]</i>
		40	I: Beim Programmieren? Im Visual Studio?
..4a Team Wor		41	B3: Ja genau, da ist man halt zu zweit mehr auf Fehler draufgekommen, als jetzt alleine. Oder wir hatten dieses Grundlagentutorial auf Moodle und das hat man auch zu zweit schneller verstanden.
..4a Team Wor		42	I: Ok ja, und habt ihr irgendwelche- also ihr habt immer nur gleichzeitig auf einem Computer gearbeitet. Nicht irgendwie parallel?
		43	Alle: Nein.
		44	I: Ok, gut. Dann schauen wir uns einmal ein bisschen was von dem programmierten Code an. Und zwar, es ist leider nicht Visual Studio, weil das gibts am Mac glaub ich nicht, aber es ist grundsätzlich nicht viel anders. Der Code ist der gleiche. Ahm, da haben wir hier zum Beispiel das Skript Fisch.cs ah Da passieren, am Anfang werden da Variablen deklariert. Was ist denn eine Variable? Was passiert da?
..1a Datenstrukturen unc		45	B2: Also einer Variable, der kann man eben einen Wert zuweisen und so einen Datentyp und ich glaub wir haben das ähm (..... ..)
..1a Datenstrukturen unc		46	I: Ok ja, du hast vorher gesagt Datentypen, was ist denn da, nehmen wir uns zum Beispiel diese Variable her. Was ist denn da der Datentyp? <i>[public string name;]</i>
..1a Datenstrukturen unc		47	B2: Ahm String.
..1a Datenstrukturen unc		48	I: String. Und was ist String für ein Datentyp?
		49	B2: Da wo man Text eingibt.
		50	I: Da wo man Text eingibt genau. Ahm dann haben wir da zu Beispiel diese Variable, was ist mit der? <i>[I markiert die Zeile public float MaxGeschwindigkeit = 10;]</i>
..1a Datenstrukturen unc		51	B2: Äh, float. Das weiß ich grad nicht.
..1a Datenstrukturen unc		52	I: Ok was. Wie heißt die Variable?
..1a Datenstrukturen unc		53	B2: Maxgeschwindigkeit.
..1a Datenstrukturen unc		54	I: Mhm, und was bekommt die für einen Wert?
		55	B2: Zehn
		56	I: Und was wird man dann drin speichern?
		57	B2: (...)
		58	(6)
		59	I: Ganz einfach, Zahlen. Also
		60	B2: Ok
		61	B3: Ja
		62	I: Also da ist die Zahl zehn drin gespeichert und float bedeutet noch, dass es auch Kommazahlen sein können. Hier habt ihr die Prozedur gehen und da ist so ein If. Was heißt denn das?
..1b Kontrollstrukturen		63	B3: Das heißt, dass (.) wenn, also (..) das passiert, dann passiert das (.)
		64	I: Mhm und wie ist es da, was passiert nur wenn dieses if zutrifft, also wenn die Bedingung im If wahr ist? Welche Zeilen vom Code werden dann ausgeführt? (..)

..1b Kontrollstrukturen   
 ..3b Interpretieren von C   
 ..1b Kontrollstrukturen   
 ..3b Interpretieren von 

..1c Objekt 

..1c Objekt 

..1c Objekt 

..5a Erstellung und Verar 

..5a Erstellung und Verar 

..5b Copyright & Lizenz: 

- 65 (20)
- 66 B2: Ich kann das nicht lesen.
- 67 I: Du kannst das nicht lesen, vielleicht kann ich das ein bissl größer machen, mal schauen. (5) So. Hier ist If und dann kommt danach Code. Und die Frage ist jetzt, der Code, der danach kommt, so, ok also wenn jetzt das h ungleich null ist, was passiert dann. Passiert dann das, oder macht er da unten weiter, oder?
- 68 B2: Also wenn ein else dann dabei steht, würd es zum else rüber
- 69 I: Mhm, aber wenn kein Else dabei ist?
- 70 B2: Ich glaub dann, (...) gehts wieder zurück nach oben.
- 71 I: Ok in dem Fall ist es so, wenn diese Bedingung, wenn das h ungleich null ist, dann wird einfach diese Zeile ausgeführt und sonst nicht.
- 72 B2: Ok
- 73 I: Ok, ahm. Jetzt schauen wir uns noch. (25) Ihr habt hier eine Datei die heißt Fisch.cs und eine Datei die heißt Nemo.cs und dann habt ihr auch noch eine Datei, die heißt Dori.cs. Wie stehen die in Verbindung zueinander?
- 74 B2: Ja das ist das mit der Parent Klasse und mit der Child Klasse.
- 75 I: Mhm, und was macht das, warum macht man sowas?
- 76 B2: Also in der, naja damit du nicht für Dori genau den gleichen Code nochmal schreiben musst. Und da wirts dann in der Parentklasse eine Funktion hinzugefügt damit dann Dori weiß, dass wenn Nemo das machen muss, dass Dori auch das machen muss. Also damit sie das Gleiche machen. Das ist sozusagen dieses Erben.
- 77 I: Genau, also ihr habt drei Dateien: Fisch, Nemo und Dori. Und was ist die Parentklasse?
- 78 B2: Fisch.
- 79 I: Fische genau. Und alles was ein Fisch kann, kann Nemo und Dori auch.
- 80 B2: Ja
- 81 B3: Genau.
- 82 I: Super, ahm, dann schauen wir uns kurz an. Hier in Spielfeld.cs was passiert mit diesem Skript. Was habt ihr da gemacht?
- 83 B2: Das mit dem Seetang haben wir (..)
- 84 B3: Das sollte dieses. Das sollte halt so, im Weg stehen. So für die Fische
- 85 B2: Na da sollte man drauf sitzen,
- 86 (...)
- 87 B2: Deswegen haben wir eben auch Seetang, wir wollen, aber dann haben wir uns gedacht, dass wir das nicht brauchen (...). Aber das mit dem Spielfeld haben wir schon eigentlich. Also das außen, rechts, links das es einfach die Grenze von dem Spiel was man (..)
- 88 I: Ok die habt ihr dann in Unity eingefügt nehme ich an. Ahm gut, dann gehen wir da jetzt wieder in Unity rein und ihr habt hier verschiedene Assets. Ahm und bei den Sprites, habt ihr Bilder. Dori, Muschel, Nemo und so weiter. Habt ihr die selbst erstellt, oder habt ihr die irgendwo hergenommen?
- 89 B3: Also, wir haben versucht im Internet Fische ohne Hintergrund zu finden, gabs aber meistens nicht. Und deswegen haben wir die dann ausgeschnitten auf ähm Gimp.
- 90 I: Ok, also ihr habt die Bilder ausm Internet geholt und dann noch ein bisschen bearbeitet und dafür Gimp verwendet.
- 91 B3: Genau.
- 92 I: Ok, habt ihr Gimp schon davor verwendet oder habt ihr das\?
- 93 B2&B3: Ja
- 94 I: Ok, wenn ihr die Bilder gesucht habt, habt ihr euch da, habt ihr geschaut, dass ihr da das Urheberrecht berücksichtigt oder nicht?

..5b Copyright &amp; Lizenz



- 95 B3: Eigentlich nicht oder?
- 96 B2: Nein
- 97 I: Ok, gut. Habt ihr auch Soundeffekte oder irgendetwas?
- 98 B2: Nein, noch nicht,
- 99 I: Ok noch nicht. Gut. Da habt ihr einen Ordner mit Vorlagen und da ist ein Seeigel drin. Was passiert da, wofür ist dieser Seeigel?
- 100 B2: Der wäre eben für die (...)
- 101 I: Ok und was macht die Vorlage, wofür haben wir eine Vorlage bei dem Seeigel? Könnt ihr mir das erklären? (5) Weil grundsätzlich, ja alle Objekte die im Spiel sind, sind ja da oben. Da haben wir den Hintergrund, da haben wir Nemo, Dori, Müllgenerator, Muschel. Und noch ein paar andere Sachen und der Seeigel ist nicht da, aber ich hab da unten eine Vorlage, wofür?
- 102 B3: Ich weiß es nicht glaub ich.
- 103 B2: (..)
- 104 I: Na, ist kein Problem. Habt ihr eure Figuren animiert im Spiel?
- 105 (..)
- 106 I: Habt ihr nicht gemacht, ok.
- 107 B3: Also das hätten wir auch machen können, aber haben wir keine Zeit mehr gehabt.
- 108 I: Ok, na es ist ganz klar, so ein Spiel zu erstellen braucht viel Zeit. So jetzt kommen wir schon langsam zum Ende. Jetzt würde mich interessieren, welche Probleme habt ihr beim Arbeiten am Projekt gehabt. Was waren so Hürden, wo ihr angestanden seid und ihr kamt nicht weiter. Was (..)
- 109 B3: Entweder eben im VisualStudio diese Fehler nicht gefunden haben, also die Fehler. (...)
- 110 B2: Also eigentlich hatten wir die Spiele schon relativ fertig und es hat einfach nicht funktioniert (....) Und da hat uns der Herr [Lehrer] dann geholfen, da haben wir beim Fishskript eine Funktion vergessen und deswegen hat das auch nicht funktioniert und dann ist das dann, also wir übersehen nachher so Funktionen, die dann doch wichtig sind.
- 111 I: Ok und wie habt ihr das Problem dann versucht zu lösen, wie seid ihr da vorgegangen?
- 112 B1: Wir haben dann den Herrn [Lehrer] gefragt.
- 113 I: Ok, ihr habt euch quasi Hilfe geholt und
- 114 B2: Ja.
- 115 I: Hat es sonst noch irgendwelche Dinge gegeben, wo ihr (.) das war besonders schwer oder irgendwie?
- 116 B3: Also ich persönlich fand immer VisualStudio am schwersten eigentlich. Also ich, das war für mich, so mit diesen ganzen Codes und ja. Und sonst aber mit dem Tutorial gings eigentlich recht gut.
- 117 I: Ok, also das Tutorial war\
- 118 B2: Ja ohne das hätten wir es nicht geschafft.
- 119 B3: Nein das glaub ich auch nicht.
- 120 I: Ok, so. Und dann ganz zum Schluss würde mich jetzt noch interessieren, wie ihr das seht, das Spiele programmieren im Informatikunterricht. Macht das Spaß, ist das eher blöd, ist das kompliziert?
- 121 B2: Es ist nicht einfach.
- 122 I: Mhm, es ist nicht einfach.
- 123 B2: Es ist aber, wenn man selbstständig arbeitet, dann lernt man auch mehr, als wenn man\
- 124 B3: nur im Unterricht sieht\
- 125 B2: ja der Lehrer zeigt das vor und du musst das nachmachen und du weißt nicht was das ist und warum aber du machst es einfach.

..4b Selbstbestimmtes A



- 126 B3: Das Problem ist einfach (.....) irgendwann, weil wir keine Spielprogrammierer und das ja.
- 127 I: Ok, ahm und eben ihr habt schon gemeint, das ist sehr kompliziert.
- 128 B3: Ja aber ich find es besser, wenn wir sowas machen, als wenn wir irgendwie nur so im Unterricht sitzen und halt irgendwas auswendig lernen oder wie diese Codes und das halt gar nicht anwenden können.
- 129 I: Also ja, Spieleprogrammierung ist ja vor allem. Also ein Aspekt, warum es so kompliziert ist, ist weil man so viele verschiedene Dinge können muss. Man muss programmieren, man muss Bilder bearbeiten, man muss sich um Soundeffekte Gedanken machen. Man muss also ganz breite. Man braucht ganz viel äh, ein breites Spektrum an Fähigkeiten und Fertigkeiten und würdet ihr das besser finden, wenn man das alles separat lernt, sodass es ein bisschen einfacher ist, So jetzt mach ich nur Code, jetzt mach ich nur Bildbearbeitung? Oder findet ihr das schon auch gut, das mit Spielen zu machen, weil Spiele sind einfach lustig und machen wir gern, oder wie seht ihr das?
- 130 B3: Ich find es gut, wenn man das davor durchgeht, und ahm für die wichtigsten Sachen mal so kennenlernt und das dann eben bei einem Spiel eben noch mehr ausprobiert und ja.
- 131 I: Ok. Habt ihr noch irgendwas zu sagen?
- 132 B2: Ja also es ist wichtig, dass man zuerst die einzelnen Sachen lernt, weil wenn man alles zusammenlernt.
- 133 *[Aufnahme endet hier]*

..5a Erstellung und Ver  
 Unityspezifisches Wiss  
 Unityspezifisches Wiss

Unityspezifisches Wiss

..1a Datenstrukturen unc

..1a Datenstrukturen unc

- 1 I: Wie viel Vorerfahrung hast du mit Unity, hast du schon bevor wir in der Schule angefangen damit zu arbeiten irgendwelche Erfahrungen gemacht?
- 2 B: Nein, absolut nicht.
- 3 I: Ok, und hast du mit Unity zu programmieren begonnen oder hast du davor schon mit C# oder anderen Sprachen programmiert?
- 4 B: Wir haben in der Schule mit C# angefangen und dann erst sobald wir mit C# wirklich gut waren mit Unity begonnen.
- 5 I: Ok und außer mit C# hast du noch keine Erfahrung mit Programmiersprachen?
- 6 B: Nicht wirklich.
- 7 I: Ok und beschäftigst du dich nur in der Schule mit programmieren und Spielentwicklung oder ist das was, was du auch zu Hause teilweise machst oder\
- 8 B: Früher dachte ich immer, dass ich später mal beruflich in die Richtung gehen werde, aber mittlerweile interessiert mich das nicht mehr so sehr. Deswegen habe ich mich früher auch ein wenig damit auseinandergesetzt, aber mittlerweile nur mehr schulisch.
- 9 I: Mhm, ok dann wars das schon mit dem ersten Teil. Dann gehen wir zu Unity und erklär mir einmal kurz worum es in dem Spiel geht. Also das Spiel ist ja, da hast ein vorgefertigtes Projekt genommen und dann quasi erweitert nach einem Tutorial. Ahm ja, erklär mir einmal kurz worum es geht und führe es mir vielleicht auch einfach vor.
- 10 B: *[Spielt das Spiel] Ja es ist relativ simpel, man hat einfach ein Raumschiff, das man herumbewegen kann, nach links und rechts und auch sogar nach vorne und muss dann den Asteroiden ausweichen und sie zerschießen. Wenn man die Asteroiden berührt dann ist es vorbei und wenn man einen zerschießt kriegt man mehr Punkte.*
- 11 I: (..) und was von dem Programm hast du da selbst gemacht?
- 12 B: Ah ich glaub am Anfang war die Bewegung der Asteroiden noch gar nicht vorhanden. Heißt die sind nur einfach gespawnt und an einer Stelle geblieben. Dann hat man halt die Bewegungen geführt und später auch die Rotation, weil die sonst nur leblos nach unten gegangen sind. Ah die Partikeleffekte haben wir zuerst selber welche prog- herumprobiert und dann später welche vorgefertigt benutzt. Das sollt halt wahrscheinlich zum Lernen da sein, dass man es auch selber erfahren kann. Ah was alles noch. Soundeffekte hat man halt verknüpft. Und ja das wars eigentlich.
- 13 I: Ok also du hast ah vor allem die die Asteroiden programmiert und diese Partikeleffekte {für die Explosionen}
- 14 B: {Ja Kollisionen eigentlich auch}
- 15 I: und die Kollisionen genau von den Asteroiden. Ok gut ahm, du hast allein gearbeitet an dem\?
- 16 B: Mhm.
- 17 I: Ok, dann schauen wir uns einmal an und zwar geh in das Script für die Asteroiden.
- 18 B: In Ordnung
- 19 (70) *Visual Studio lädt*
- 20 I: So, zunächst werd ich einmal über die Daten und Referenztypen, Kontrollstrukturen, Operatoren, Variablen kurz also, werden wir das besprechen. Und da würde mich einmal interessieren, du hast hier am Anfang, es werden sehr viel Variablen deklariert,
- 21 (50) *Technische Unterbrechung*
- 22 I: So, fangen wir vielleicht ganz grundlegend an, was ist eigentlich eine Variable beim Programmieren, was wozu verwendet man die?
- 23 B: Ja, also je nach dem Typen sind das entweder Buchstaben, beim String, oder Zahlen bei float oder int.
- 24 I: Mhm, ok. Was, da steht public float geschwindigkeit. Was heißt das float jetzt genau?
- 25 B: Ah, ich glaub das halt unterschiedlich zwischen int und float wie das abgerundet wird.
- 26 I: Mhm, ok ja. Also float sind Fließkommazahlen, also Dezimalzahlen und int sind nur ganze

..1a Datenstrukturen unc		Zahlen. In dem Fall, eine Kommazahl für die Geschwindigkeit ist das wichtig glaubst du oder hast du das bewusst als float gemacht, oder?
Unityspezifisches Wiss ..1a Datenstrukturen u		27 B: Ja das ist ganz wichtig, damit das halt stabil langsamer und schneller werden kann.
..1a Datenstrukturen unc		28 I: Mhm, ok ja damit du auch zwischen den, also nicht nur ganzzahlige Geschwindigkeiten haben kannst ja. Ja das ist gut. Ahm dann hast du hier noch andere Variablentypen, zum Beispiel Rigidbody. Was ist das?
Unityspezifisches Wiss ..1a Datenstrukturen u		29 B: Hmm (2) da muss ich zugeben, dass ich mich nicht so gut auskenne. Aber ich denk halt einfach, dass das halt die Physik von dem Objekt halt kontrolliert.
..1a Datenstrukturen unc		30 I: Genau, also jedes Objekt in Unity, dass ah hmm teil der Physikengine sein muss ah hat einen Rigidbody oder braucht einen Rigidbody und ja über den regeln wir die Physiksachen. Dann haben wir da noch weitere, ähm, Audioclip wird?
Unityspezifisches Wiss ..1a Datenstrukturen u		31 B: Ja ist einfach einfach nur ein Audiofile.
..3b Interpretieren von ..1a Datenstrukturen u		32 I: Genau ja und Gameobject?
..1b Kontrollstrukturen ..3b Interpretieren von		33 B: Das ist eigentlich ziemlich das selbe, halt nur ein Effekt glaub ich.
..1b Kontrollstrukturen		34 I: Ja Gameobject is ganz allgemein für alle Objekte, das ist ein Übertyp für alle Objekte die in Unity im Spiel sein können. Ahm diese Objekte, diese Variablen die da oben sind. [I deutet auf Objektvariablen] Kann man die überall in dem ganzen Skript verwenden?
..3b Interpretieren von ..1a Datenstrukturen u		35 B: Ah nein.
..1b Kontrollstrukturen		36 I: Nicht?
..3b Interpretieren von		37 B: Also, obwohl schon.
..1b Kontrollstrukturen		38 I: Schon ja, ahm, das sind nämlich eben Objektvariablen, die kann man da im Ganzen Skript immer verwenden und das machst du auch also. Ahm, also genau. (2) Kommen wir zu Kontrollstrukturen. Du verwendest hier ein If, wieso? Was macht das If?
..3b Interpretieren von ..1a Datenstrukturen u		39 B: Ja, es schaut halt ob das Objekt mit dem, das Ship collidet, obwohl der Asteroid collidet. Äh ein Geschoß ist, oder jetzt in dem Fall wärs halt das Raumschiff jedenfalls stattdessen. Und man will ja nicht, dass der Asteroid zerspringt, wenn das Raumschiff den Asteroiden berührt. Sondern umgekehrt
..1b Kontrollstrukturen		40 I: Ok ahm. Aber du überprüfst hier quasi other.tag. Weißt du was das ist? [other ist der Collider des die Kollision triggernden Spielobjekts]
..3b Interpretieren von ..1a Datenstrukturen u		41 [if(other.tag == „Geschoß“)]
		42 B: Ich weiß nicht mehr genau.
		43 I: Also other ist ahm haben wir gesagt ein Collider und\
		44 B: Wahrscheinlich irgendein anderes Objekt.
		45 I: Ja und welches Objekt?
		46 B: Ja in dem Fall wärs halt entweder das Geschoß, oder das Raumschiff glaub ich.
		47 I: Mhm, ok, aber schauen wir uns einmal an. Wir sind jetzt gerade im Asteroidenscript und da passiert eine ganze Reihe von Dingen und davor steht ein if. Was macht das if ganz generell?
		48 B: Ja überprüfen ob das stattfindet.
		49 I: Ja also, wie nennt man das, was da in den runden Klammern dazwischen drinnen ist?
		50 B: Ah (2) ich weiß es nicht genau. Also ich weiß was es macht, also
		51 I: Ok, was machts denn?
		52 B: Ja eine Bedingung.
		53 I: Genau eine Bedingung, das ist die Bedingung und wenn die Bedingung erfüllt ist, dann\
		54 B: Ja dann passiert das Ganze.
		55 I: Dann passiert das Ganze genau. Und in dem Fall wird überprüft ob der Tag von dem anderen Objekt, mit dem die Kollision stattfindet, ob das den Tag Geschoß hat. Was macht dieses doppelte = da?

...3b Interpretieren von ...1a Datenstrukturen u		56	B: Ah Ist Genau das.
		57	I: Genau das überprüft auf Gleichheit. Ahm gut. (10) Weißt du was dieses private und public bedeutet bei den Objektvariablen oben.
...1c Objekt ...1a Datenstrukturen u		58	B: Wahrscheinlich wie man drauf zugreifen kann.
		59	I: Ja genau, wie man die generell, wie der Begriff für diese Dinge lautet?
		60	B: mh mh {schüttelt den Kopf}
...1a Datenstrukturen u Unityspezifisches Wiss		61	I: Ok ja das sind die Zugriffsmodifikatoren, mit denen kann man eben, ja von wo man aus drauf zugreifen kann. Ahm wenn du in Unity reingehst, siehst du, wenn du auf den einen Asteroiden, wenn du auf die Vorlage gehst. Siehst du hier die Geschwindigkeit, Rotationsgeschwindigkeit, Clipaudio und Explosionseffekt.\
		62	B: Aber die private Sachen halt nicht.
		63	I: Genau, die public Sachen sieht man in Unity und die private Sachen nicht.
		64	B: (.) wird den Rigidbody wahrscheinlich nicht verändert- selber verändern müssen.
Unityspezifisches Wiss		65	I: Bitte?
		66	B: Weil wir den Rigidbody und sowas wahrscheinlich selber nicht verändern müssen wie die Sachen.
		67	I: Ahm, genau nicht von außen. Der Rigidbody wird verändert (3) hier drin, aber eben nicht von außen. Und der Vorteil ist eben, die Geschwindigkeit kann man hier einfach setzen. Gut, so. (10) Passt dann gehen wir weiter und schauen uns an. (5) Hier eben ja genau diesen Algorithmus. Was ist ein Algorithmus eigentlich?
...3a Problemlösungs Ori		68	[I markiert den Algorithmus zur Kollisionsbehandlung in Asteroid.cs]
		69	B: Ah ja, ich würd sagen. Ich bin nicht ganz sicher, ich würde sagen eine bestimmte Vorgehensweise.
...3b Interpretieren von C		70	I: Mhm, ja. Um ein bestimmtes Problem zu lösen. Und hier in diesem in dieser Funktion wird ein bestimmtes Problem behandelt.
...3b Interpretieren von C		71	B: Was passiert wenn man auf ein Geschöß trifft mit dem Asteroiden.
...3b Interpretieren von C		72	I: Genau. Und wir haben schon vorher über das If geredet. Ahm passiert da eigentlich irgendwas, wenn wir nicht mit einem Geschöß zusammentreffen, sondern mit etwas anderem.
		73	B: Nein, also von dem aus nicht.
		74	I: Genau, da passiert gar nix. Ahm und wenn was passiert, was passiert dann?
		75	[B geht den Code Zeile für Zeile durch]
...3b Interpretieren von ...3a Problemlösungs C		76	B: Ja das andere Objekt, was in dem Fall das Geschöß ist, wird zerstört. Weil man will ja nicht dass es weiter herumfliegt.
		77	I: Das würd sonst durchfliegen.
...3b Interpretieren von ...3a Problemlösungs C		78	B: Ja, äh der Asteroid selber wird zerstört. Wir kriegen zehn Punkte, damit wir den (.) den Punktestand testen können. Der Soundeffekt für die Explosion wird abgespielt. Und der Explosionseffekt wird halt dargestellt.
Unityspezifisches Wiss ...5a Erstellung und Ver		79	I: Ok ja. Der Explosionseffekt hier. Was ist das da?
		80	B: Ein vorgefertigter Partikeleffekt, denke ich.
		81	I: Genau also ein vorgefertigter Partikeleffekt. Der wird hier ins Spiel eingefügt. Ahm (3) genau. Dann können wir gleich weitergehen und zwar schauen wir uns einmal an, was für Assets du im Spiel selbst erstellt hast. Und zwar hast du Partikelsysteme erstellt oder eins erstellt vor allem. Ein Material erstellt für das Partikelsystem und du verwendest Vorlagen aus den Standardassets von Unity. Ahm bei dem Partikelsystem, kannst du das einmal aufmachen und anschauen, wo du das findest.
		82	B: Achso ich dachte darstellen. Dargestellt wirds auf jeden Fall da. [B startet das Spiel und zeigt den Partikeleffekt]
Unityspezifisches Wiss		83	I: Mhm, kannst du mir erklären wie so ein Partikelsystem grundsätzlich funktioniert? Was das

Unityspezifisches Wiss ..5a Erstellung und Ver		eigentlich 84 B: Wahrscheinlich einfach ein bestimmtes Bild halt, öfters pro Sekunde, also definiert aussenden. Je nachdem auch mit den bestimmten, mit einer bestimmten Richtung. Weil die fliegen jetzt zum Beispiel nur nach hinten.
Unityspezifisches Wiss ..5a Erstellung und Ver Unityspezifisches Wiss		85 I: Genau, so und wenn du da auf Player, da ist das Partikelsystem. Weißt du noch, also kannst du mir nur erklären wie du das erstellt hast? Nur ganz grob du musst nicht jedes Detail. Ahm (5) 86 B: Naja, ich denke (4) ich weiß nicht mehr genau. Ja hier ist halt das Bild, was da wirklich ist. <i>[B öffnet die Einstellungen des Partikeleffekts]</i> 87 I: Mhm, das Bild da ahm. Das ist das verwendet was? Du hast hier, ah\ <i>[I deutet auf die Einstellung „Shader Particles/Additive“]</i> 88 B: Ja Partikel die additiv sind, heißt wenn mehrere übereinander sind, wird es heller. 89 I: Genau das sieht man da auch ganz schön. In der Mitte ist es ganz hell und auf der Seite weniger hell. Was heißt zum Beispiel die Duration da oben.
Unityspezifisches Wiss		90 B: Äh wie lang die bleiben bevor sie verschwinden.
Unityspezifisches Wiss ..5a Erstellung und Ver		91 I: Genau, also jedes Partikel, jedes also. Eben das Partikelsysteme erzeugt kleine Bilder, ganz viele davon, die Ganze Zeit. Und die haben alle eine Duration, eine Lebensdauer sozusagen. 92 B: Das war noch ganz wichtig. <i>[Zeigt auf Size over Lifetime] Damit die kleiner werden. Also mit der Zeit kleiner werden. Ich glaube die Shape habe ich nicht verändert. Ah das ist glaub ich einfach wie viele rauskommen. Aber ich bin mir nicht sicher ob ich das verändert habe. Und sonst war eigentlich eh alles schon da glaub ich.</i> 93 I: Ok ja, ok. Genau über das Material haben wir auch schon kurz gesprochen. Geh einmal in den Ordner Materials. Und da ist das Material Raketenantrieb. Und das wird eben im Partikelsystem verwendet und du hast schon richtig gesagt. Der Shader, das Particles-Additive gibt an, dass wenn mehrere übereinanderliegen, dass die, dass es heller wird. Ahm was passiert, wenn du jetzt da zum Beispiel ein anderes Bild auswählst. 94 B: Es wird sich wahrscheinlich, also. 95 I: Nimm einmal irgendwas anderes. 96 B: Irgendwas anderes? 97 I: Irgendwas. 98 B: Das ist halt größer und dadurch tut es halt auch ein bisschen in das Raumschiff selbst reinglitchen. Und es wird halt kleiner, aber ist immer noch größer als zuvor.
Unityspezifisches Wiss ..5a Erstellung und Ver		99 I: Genau, also man kann da einfach verschiedene Bilder auswählen. Und was passiert mit dem Schwarz, mit dem Hintergrund vom Bild? 100 B: Das ist wahrscheinlich gar kein Schwarz sondern einfach durchsichtig. 101 I: Genau, das ist dann transparent im Partikeleffekt. Ok gut. Welche Probleme hast du beim Erstellen von diesem Projekt gehabt. Was, was, was waren so Hürden wo du kurz angestanden bist oder wo du dir gedacht hast\
..4b Selbstbestimmtes At		102 B: Im Großen und Ganzen, wars ganz in Ordnung, aber ich glaub, es gab so ein, zwei Stellen in dem Tutorial, wo irgendwas nicht ganz klar war. 103 I: Mhm, wie hast du das versucht dann zu lösen, das Problem? Also wie bist du dann vorgegangen. 104 B: Einfach wieder zurückgegangen im Tutorial und noch einmal durchlesen. 105 I: Ok ja. Also du hast einfach\ 106 B: Und schauen ob ich irgendwas fehlverstanden habe. Am Ende hats eh geklappt dann.
..4b Selbstbestimmtes At		107 I: Ok. Aber du hast nur mit dem Tutorial gearbeitet und sonst\ 108 B: Ich glaube einmal habe ich was gegoogelt, aber ich bin mir nicht sicher ob das wirklich geholfen hat. 109 I: Ok ja du hast was gegoogelt. Weißt du noch äh was das da für ein Problem war? Was du da gegoogelt hast?

..4b Selbstbestimmtes Ai

- 110 B: Es kann sein, dass das da bei dem war. *[Zeigt im Script auf den Funktionsaufruf FindeGameController()] obwohl, hmm (5). Ich glaube irgendwas war da nicht definiert, weil ich es falsch gemacht hab.*
- 111 I: Ok ja. Und du weißt nicht mehr ob die Google-Recherche dann schlussendlich das Problem gelöst hat?
- 112 B: Ich glaub sogar schon, zumindest in dem einen Fall.
- 113 I: Und weißt du, hast du da ein Video gefunden, ein Youtubevideo oder\
- 114 B: Nein nur so einen kurzen Post, wo jemand eine Frage gestellt hat und dann halt in den Kommentaren stand.
- 115 I: Ok ja, gut. Und sonst, sonst noch irgendwelche interessanten Dinge die du dazugelernt hast, die dir aufgefallen sind beim arbeiten?
- 116 B: Nicht wirklich, aber ich fand das ganze schon ganz cool.
- 117 I: Und würdest du, würd es dich interessieren sowas weiterzumachen? Also weiterhin zum Beispiel an genau diesem Projekt noch weiterarbeiten, mehrere Features einzubauen.
- 118 B: Hmm naja, ich glaub da ist das Potenzial schon ziemlich ausgeschöpft.
- 119 I: Mhm, ok. Und dann würd mich noch interessieren, wie du das Programmieren in Verbindung mit Spielentwicklung eigentlich siehst. Findest du, dass das Programmieren da eher in den Hintergrund rückt, oder ist es eigentlich toll da zu Programmieren und gleichzeitig da ein Spiel zu machen?
- 120 B: Ich find das viel interessanter, weil man halt auch direkt sehen kann, was deine Änderungen halt machen. Und normalerweise hat man entweder einfach nur Bilder oder irgendwie läuft irgendwas nicht so wie man es will. Aber hier hat man halt auch so Sachen, wo man sich denkt, dass es halt einfach vom Spielgefühl her einfach besser sein könnte. Wenn jetzt irgendwie die Asteroiden nicht schnell genug sind oder sowas.
- 121 I: Also das Ergebnis ist viel interessanter und angreifbarer und findest du das motivierend?
- 122 B: Ja schon.
- 123 I: Also du würdest lieber Programmieren lernen mit Spielen als irgendwie Konsolenprogramme schreiben. {Kann man das so sagen?}
- 124 B: {Mhm}
- 125 I: Gut dann vielen Dank für das Interview.