HUMBOLDT-UNIVERSITÄT ZU BERLIN
MATHEMATISCH-NATURWISSENSCHAFTLICHE FAKULTÄT
INSTITUT FÜR INFORMATIK

Serious Game basierter Ansatz als Hilfe für Programmieranfänger

Bachelorarbeit

zur Erlangung des akademischen Grades
Bachelor of Science (B.Sc.)

eingereicht von: Elias John

geboren am: 21.03.1995

geboren in: Frankfurt (Oder)

Gutachter/innen: Dr. Le

Prof. Dr. Pinkwart

eingereicht am: ..... verteidigt am: .....

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

Der Arbeitsmarkt ist in Deutschland im steten Wandel. Dabei spielt vor allem die rasche Entwicklung digitaler Technik in den letzten 40 Jahren eine entscheidende Rolle. Immer mehr Berufe werden automatisiert, immer mehr digitalisiert. Zeitschriften sind mittlerweile überwiegend im Internet vertreten. Roboter übernehmen Arbeiten in Fabriken und die Nachfrage nach einer Infrastruktur vernetzter Alltagsgegenstände, dem Internet of Things, ist höher denn je. Durch die steigende Anzahl an Software, Hardware und Webseiten bedarf es auch immer mehr Fachkräfte im Informationstechnik-Sektor. Damit steigt die gesellschaftliche Bedeutung der Informatik stetig an.

Trotz einer zunehmenden Anzahl an Informatik-Studierenden, wird der Bedarf an Informatiker*innen und Softwareentwickler*innen nicht gedeckt (Gesellschaft für Informatik 2015). Stattdessen steigt die Zahl der unbesetzten Stellen seit 2014 pro Jahr stetig an (Koppel 2017). Bis 2020 werden laut der Bertelsmann Stiftung (2010) 174.000 Hochschulabsolvent*innen aus dem Bereich Mathematik und Informatik benötigt.

Dieser Mangel an Fachkräften wird zudem durch eine hohe Abbrecherquote im Studium verstärkt. So beziffert das DZHW (Deutsches Zentrum für Hochschul- und Wissenschaftsforschung) im Projektbericht 2017 (Heublein, et al. 2017) die Abbrecherquote in Informatikstudiengängen in Baden-Württemberg auf geschätzte 31 Prozent, wobei über 50 Prozent der Studierenden das Studium bereits nach zwei Semestern abbrechen. In der Wochenzeitschrift „Die Zeit“ berichtete Wiarda (2016) sogar von einer 45 prozentigen Abbrecherquote an Universitäten in Informatik-Studiengängen deutschlandweit.

Hinzu kommt, dass nur wenige Studierende die Regelstudienzeiten einhalten. Laut dem Statistischen Bundesamt (2016) haben 2014 nur 38,2 Prozent aller Informatik-Studierenden ihr Studium in der vorgesehenen Regelstudienzeit abgeschlossen. Das lässt auf eine Überforderung bei der Bewältigung des Lerninhalts in der geplanten Zeit schließen. Laut Muratet et al. (2009) empfinden viele Studierende zu Beginn des Studiums das Themengebiet als zu theoretisch und mühsam. Die gelernten Techniken scheinen zu abstrakt und nicht praxisrelevant.

Es stellt sich daher die Frage, wie Lerninhalte praxisnäher erläutert und vermittelt sowie bereits Schüler*innen für Themen der Informatik begeistert werden können. Als Mittel für die Motivation scheinen sich besonders Spiele zu eignen. Spiele ermöglichen einen leichten und

niedrigschwelligen Zugang zum jeweiligen Thema und bereiten den Lernenden Spaß. In der vorliegenden Bachelorarbeit soll daher folgende **Frage** beantwortet werden:

Können mithilfe eines Serious Games basierten Lernsystems Motivation und Lernerfolg für grundlegende Programmierkenntnisse gesteigert werden?

Um diese Frage zu beantworten, werden im folgenden Kapitel zwei zunächst bekannte Lernansätze und -theorien sowie Lern-Modelle erläutert, um eine didaktische Grundlage für das vorzustellende System zu schaffen. Dabei zeigt sich, dass Motivation und Spaß für erfolgreiches Lernen essentiell sind. Diese notwendigen Schlüsselfaktoren können unter anderem mit dem Einsatz von Spielen gefördert werden. Insbesondere wird hierbei auf die Begriffe Serious Games und Gamification eingegangen, welche den Einsatz von Spielen bzw. Spielementen in einem Nicht-Spiele-Kontext beschreiben. Durch Berücksichtigung des individuellen Lernverhaltens und Vorwissens kann der Lernerfolg ebenfalls gesteigert werden. Auch wird geklärt, welche Programmierkonstrukte grundlegend sind und daher vermittelt werden sollten.

Auf Basis dieses Wissens wird im dritten Kapitel das Konzept der Lernplattform PLAY & CODE vorgestellt. Diese vermittelt grundlegende Programmierkonstrukte mittels Serious Games und passt die Texte individuell an den Benutzenden an. Unter Berücksichtigung des ARCS-Modells von Keller (1987) sowie dem Modell von Merrill (2002) soll eine Lernplattform geschaffen werden, die die Motivation und den Lernerfolg von Schüler*innen steigert. PLAY & CODE bietet Lernkurse an, welche jeweils ein abgeschlossenes Spiel darstellen. In jeder Lektion eines Kurses wird die Geschichte weitererzählt und der Lernende wird vor ein neues Problem gestellt, welches er mit dem neu erworbenen Wissen lösen kann. Jede Lektion lässt dabei einen gewissen Grad an Kreativität und Individualität im Code zu. Beispielhaft wird der Kurs „Crocos Bruder“ vorgestellt.

Anschließend wird im vierten Kapitel ein Prototyp des zuvor konzipierten Systems präsentiert. Die Datenbank sowie Frontend- und Backendkomponenten werden erläutert und deren Beziehungen untereinander detailliert erklärt. Zudem wird ein konzipiertes Spielframework beschrieben, das einen Rahmen für zukünftige Serious-Game Kurse in PLAY & CODE darstellt.

Um die Effektivität der Lernplattform messen und eine Antwort auf die Forschungsfrage geben zu können, wurden Daten erhoben und ausgewertet. Die Plattform wurde hierfür an mehreren Schulen evaluiert. Das genaue Vorgehen und die erfassten Variablen werden in Kapitel 5

detailliert erläutert. Anschließend werden die Daten im sechsten Kapitel hypothesengeleitet analysiert und ausgewertet. Es wird eine Antwort auf die Forschungsfrage gegeben. Abschließend folgen ein Fazit und ein Ausblick.

2 Stand der Forschung

2.1 Lerntheorien

Lernumgebungen stellen Lerninhalte bereit und vermitteln diese auf unterschiedliche Weise. Um solche Umgebungen beschreiben zu können, ist es von Vorteil, zunächst einige lerntheoretische Grundlagen zu erklären. Lerntheorien versuchen Lernen als psychischen und sozialen Prozess formal zu definieren. Sie haben somit keinen direkten Einfluss auf die Entwicklung einer Lernumgebung, geben aber einen wichtigen Einblick in die Funktionsweise des Lernens. Da sich der Lernprozess abhängig von der jeweiligen Person gestaltet, existieren verschiedene Lerntheorien, welche die unterschiedlichen Lernstile berücksichtigen (Reinmann 2011). Nach Hubwieser (2007) und Reinmann (2011) lassen sich dabei drei theoretische Hauptströmungen unterscheiden: Der Behaviorismus, der Kognitivismus und der Konstruktivismus.

Basierend auf dem Reiz-Reaktions-Modell beschreibt der **Behaviorismus** Lernen als Verhalten, welches sich durch Reize und deren Folgen verändern und konditionieren lässt. Der Behaviorismus verfolgt den Anspruch, die grundlegenden Erkenntnisse durch Laboruntersuchungen und Experimente hinsichtlich ihrer Evidenz zu stützen und zu begründen. Es wird dabei, laut Reinmann (2011), nach Ursache-Wirkungsbeziehungen geforscht. Der/die Lehrende nimmt dabei eine autoritäre Rolle als Reiz- und Konsequenzgeber*in ein.

Als Gegenbewegung hat sich der **Kognitivismus** etabliert. Die Ursprünge dieser Theorie liegen im mathematisch-informatischen Bereich. Lernen wird, ähnlich wie der Schreibprozess eines Computers, als Informationsspeicherung im Gehirn verstanden. Hiermit verbunden ist die Veränderung der Neuronen und somit der Struktur des Gehirns. Der Lehrende wird als Sender und der Lernende als Empfänger von Lerninhalten verstanden. Im Vergleich zum Behaviorismus nimmt der Lernende eine aktivere Rolle ein, da ihm hier statt reaktivem Verhalten auch zielgerichtetes Vorgehen und Problemlösefähigkeit zugesprochen werden. Der Lehrende bestimmt weiterhin Lerninhalt und -methoden.

Vertreter*innen des **Konstruktivismus** sind der Auffassung, dass die Wahrnehmung der Umwelt immer subjektiv vom Betrachtenden abhängt, das heißt, eine objektive Umwelt, die per se gegeben ist, existiert nicht. Demzufolge ist auch die Informationsaufnahme ein subjektiver Prozess. Wissen wird als „individuelle und soziale Konstruktionsleistung des Menschen“ be-

trachtet (Reinmann 2011, 4). Statt Instruktion durch einen Lehrenden steht daher, nach Hubwieser (2007), die individuelle Konstruktion im Vordergrund. Der Lernende wird aktiv beteiligt. Lernumgebungen sollten Platz für Erfahrungen und Reflexion bieten sowie authentische Inhalte und Problemsituationen thematisieren, sodass Wissen konstruiert bzw. „hergestellt" werden kann.

Zu jeden dieser theoretischen Hauptströmungen gibt es außerdem verschiedene Varianten und Hybridmodelle. Beispielsweise ähneln die Theorien von Vygotsky (zitiert in Kozulin 2004) dem Konstruktivismus. Wissen wird zwar auch konstruiert, allerdings wird Lernen hier verstärkt als soziokultureller Prozess verstanden (Vygotsky zitiert in Kozulin 2004). Zu der Frage, welche Lerntheorie zu bevorzugen sei, gibt es zahlreiche Diskussionen. So argumentieren beispielsweise Kirschner et al. (2006), dass der Konstruktivismus nicht mit der Beschaffenheit des Arbeit- und Langzeitgedächtnisses harmoniert. Der Lernende wäre überfordert und das Lernen daher nicht effektiv. Die Autoren plädieren für einen instruktionsgeführten Ansatz.

2.2 Didaktische Modelle als Grundlage für Lernplattformen in der Informatik

Um Konzepte von Lernplattformen zu verstehen, sind Lerntheorien aber ohnehin nicht ausreichend. Sie dienen jedoch dazu, Prinzipien für didaktische Modelle abzuleiten. Lerntheorien stellen somit die Grundlage für didaktische Modelle dar. Um ein didaktisches Modell zu entwickeln, bedarf es der Definition des Zwecks und der Ziele einer zu entwickelnden Lernplattform (Reinmann 2011). Beim Erlernen von Programmierung ist neben der Aneignung der Programmiersprache vor allem die Problemlösefähigkeit ein anerkanntes Lernziel (vgl. Hubwieser, 2007; Kazimoglu, et al., 2012). Durch ein strukturelles Verständnis der erlernten Programmierkonstrukte können eigene Problemlösestrategien für neue Aufgabenstellungen entwickelt werden. Diese Fähigkeit wird als Kompetenz bezeichnet (Reinmann 2011). Das bloße Lernen von Befehlen führt hingegen zu eher oberflächlichem und nicht weiter anwendbarem Wissen (Kazimoglu, et al. 2012). Das Wissen kann dann zwar wiedergegeben, aber in konkreten Problemsituationen nicht angewandt werden. Daher erscheint es sinnvoll, neuen Lernstoff mit einem konkret zu lösenden Problem zu verbinden. Dieses Lernen wird als problembasiertes Lernen bezeichnet und hat sich nach Dörner et al. (2016) und Savery (2006) als effektiv erwiesen: In mehreren Studien konnte nach Savery (2006) gezeigt werden, dass problembasiertes Lernen traditionellen Ansätzen in ihrer Effektivität nicht nachsteht und zudem die

Problemlösefähigkeit der Lernenden stärkt. Durch den Problembezug wird der Lerninhalt konkretisiert und wirkt so weniger abstrakt, was sich unter anderem positiv auf die Motivation der Lernenden auswirken kann.

2.2.1 Schlüsselfaktoren für Lernerfolg

Grundsätzlich scheint Motivation eine wichtige Bedingung für erfolgreiches Lernen zu sein. So beschreibt Hubwieser die Motivierung als „das vordringlichste Ziel didaktischen Handelns“ (Hubwieser 2007, 15). Aktive Motivation ist eine wichtige Bedingung für Lernerfolg. Es wird dabei zwischen intrinsischer und extrinsischer Motivation unterschieden. Während extrinsische Motivation durch äußere Einflüsse, wie zum Beispiel gute Noten, erzielt werden kann, entsteht intrinsische Motivation aus einem innerem Antrieb, ohne externe Quelle, heraus. Nach Hubwieser (2007) sollte vorwiegend versucht werden, eine intrinsische Motivation der Lernenden zu erreichen. Das ARCS Modell von Keller (1987) stellt hierfür vier konkrete Aspekte vor, wie Motivation erzeugt werden kann (siehe auch Hubwieser, 2007):

- Die Aufmerksamkeit des Lernenden muss erzeugt werden. (Attention)
- Die Relevanz des Lerninhalts durch Kontextbeziehung sollte erkennbar gemacht werden. (Relevance)
- Zuversicht sollte erzeugt werden: Aufgaben sollten fordernd, aber machbar sein. (Confidence)
- Ein Lernerfolg muss sichtbar sein, sodass sich Zufriedenheit des Lernenden einstellen kann. (Satisfaction)

Nach Dörner et al. (2016) ist außerdem Spaß eine wichtige Komponente für intrinsische Motivation. Eine weitere wichtige Bedingung für den Lernerfolg ist Kreativität (siehe Hubwieser, 2007; Stevenson et al., 2006). Nach Hubwieser ist Kreativität im Sinne individueller Lösungsstrategien des Lernenden „Voraussetzung für die Neukonstruktion von Wissen“ (Hubwieser 2007, 17). Mit der Möglichkeit, Probleme auf unterschiedliche Weise kreativ lösen zu können, lassen sich Lerninhalte anwendungsorientiert vertiefen. Dafür ist eine offene, nicht fest vorgeplante Lernumgebung notwendig.

2.2.2 Individuelle Lernstile

Da jeder Lernende einen eigenen Lernstil und unterschiedliche Voraussetzungen mitbringt, ist es außerdem notwendig, auf diese individuell einzugehen. Diverse Studien haben nach Vandewaetere et al. (2011) gezeigt, dass dieser individuumsbasierte Ansatz uniformen Modellen, die die Bedürfnisse der Lernenden kaum berücksichtigen, überlegen ist. Lehrmethoden, -inhalte und -materialien werden bei diesem Ansatz individuell an den Lernenden angepasst (Adaptivität). Dieser Ansatz kann sowohl im schulischen Kontext als auch bei Lernplattformen Anwendung finden. Speziell bei Lernplattformen erstreckt sich Adaptivität von grafischen Oberflächenanpassungen, wie das Verändern der Schriftgröße, bis hin zu dynamischen Restrukturierungen von Kursen und der Selektion alternativer Kursmaterialien (Paramythis und Loidl-Reisinger 2003). Für Instruktionen in Lernumgebungen gibt es nach Vandewaetere et al. (2011) drei Ansätze für Adaptivität.

- Makro-adaptive Instruktion: Geht von jeweils unterschiedlichen Lerngeschwindigkeiten des Lernenden aus und erlaubt es ihm, mit einer für ihn passenden Geschwindigkeit den Lerninhalt zu verarbeiten.
- Fähigkeitsbehandlungs-Interaktion (Aptitude-treatment Interaction): Mittels eines statischen, vorgeschalteten Tests wird versucht, die Fähigkeiten des Benutzenden zu identifizieren und darauf aufbauend einen passenden Inhalt bereitzustellen.
- Mikro-adaptive Instruktion: Bedürfnisse des Lernenden werden nicht statisch, sondern kontinuierlich während der Instruktion analysiert und der Inhalt dynamisch angepasst.

Vor allem der Mikro-adaptive Ansatz scheint erfolgsversprechend zu sein, da hierbei mehr Charakteristika des Lernenden erfasst und verarbeitet werden können als bei den anderen Ansätzen. Beim Makro-adaptiven Instruktionsansatz sowie beim Aptitude-treatment Interaction Ansatz werden weniger Daten erfasst und es besteht die Gefahr, die individuellen Charakteristika des Lernenden unterkomplex zu behandeln (Vandewaetere, Desmet und Clarebout 2011).

2.2.3 Instruktionsmodelle

Auch für die Vermittlung von neuem Wissen existieren verschiedene Modelle. Ein bekanntes stellt dabei das von David Merrill vorgestellte „First Principles of Instruction“ (2002) Modell

dar. Merrill hat über mehrere Jahre verschiedene Instruktionsmodelle untersucht und dabei fünf verbreitete Prinzipien herausgearbeitet:

- Das Lösen von realitätsnahen Problemen fördert den Lerneffekt.
- Neues Wissen sollte auf bereits erworbenem Wissen aufbauen. So wird vorheriges Wissen in Beziehung gesetzt, wiederholt, angewandt und dadurch gefestigt.
- Neues Wissen muss dem Lernenden demonstriert werden. Konkrete Positiv- und Negativbeispiele, Visualisierungen und Vergleiche von Konzepten sind dabei hilfreich.
- Der Lernende sollte das neu erworbene Wissen anwenden. Diese Übung sollte konsistent mit den Lernzielen sein. Dabei sollte der Lernende Feedback erhalten, um Fehler zu erkennen.
- Durch Anwendung des Erlernten im Alltag wird Wissen langfristig bewahrt.

Modelle wie das ursprünglich von Gagné 1974 vorgestellte „Nine Events of Instruction" (Gagné, Briggs und Wager 1992, 11 - 12) oder die acht Kriterien für gute Programmieraufgaben von Stevenson und Wagner (2006) setzen andere Schwerpunkte, decken aber weitestgehend die genannten Punkte von Merrill ab. So empfehlen Stevenson und Wagner (2006) für Programmieraufgaben zusätzlich das Verwenden von Programmierschnittstellen (engl. Application Programming Interfaces, APIs), um ein realistisches Szenario zu schaffen. Außerdem sollten Aufgaben Kreativität zulassen und auch fordernd, aber nicht überfordernd sein. Eine Demonstration von Wissen wird hier nicht erwähnt. Gagné (1992) führt den Demonstrationsaspekt weiter aus, erwähnt aber keine realitätsnahe Problemgrundlage.

Auf Basis der vorgestellten theoretischen Ansätze lässt sich festhalten, dass Motivation eine wichtige Bedingung für erfolgreiches Lernen darstellt. Das Lernthema sollte problemorientiert eingeführt werden, sodass die Relevanz klar erkennbar ist. Außerdem sollte der Lerninhalt individuell auf die Lernenden abgestimmt sein und Kreativität beim Lösen der Übungsaufgaben zulassen. Zudem sollten anerkannte Prinzipien für Instruktionen Berücksichtigung finden. Nachdem grundsätzliche Aspekte für erfolgreiches Lernen und deren Eigenschaften beschrieben wurden, stellt sich nun die Frage, wie diese in e-Learning Umgebungen umgesetzt werden können. Was sind geeignete Mittel um Lernende ausreichend und langfristig zu motivieren?

2.3 Spiele

Ein mögliches Mittel, um Motivation von Lernenden zu fördern sowie Kreativität und Adaptivität zuzulassen, sind Spiele. Dignan beschreibt Spiele als Konstrukte, in welchen ein konstruierter Konflikt ausgetragen wird, der durch die Anwendung vorgegebener Regeln gelöst und durch ein quantifizierbares Ergebnis, zum Beispiel dem Punktestand, belohnt wird (Dignan 2011, 35). Diese Definition deckt sowohl Brettspiele, Sport- als auch Videospiele ab. Alle Spiele haben gemein, dass sie fordernd sind und zumeist eine kreative Lösung des Problems bzw. des Konflikts verlangen. Spiele, welche ein unterhaltendes, spannendes oder lustiges Erlebnis ermöglichen, wirken motivierend und erzeugen Interesse und Neugierde. Errungenschaften bzw. Erfolge im Spielprozess tragen ebenfalls zur Motivation bei (Dörner, et al. 2016). Trotz eines festen Regelwerks sind die Spielenden frei in ihren Entscheidungen. Motivierend wirkt auch das Voranschreiten zum Ziel. In einer Studie von Amabile und Kramer (2010) wurden über mehrere Jahre Angestellte befragt, was sie am meisten in ihrer Erwerbsarbeit motiviert. Der überwiegende Teil gab „Fortschritt“ in der zu erledigenden Arbeit als Motivationsquelle an. Da bei Spielen – wie bereits erwähnt – quantifizierbare Ergebnisse von Bedeutung sind, kann hieran der Fortschritt direkt abgeschätzt und eingesehen werden (Dörner, et al. 2016).

Als Unterhaltungsmittel sind Spiele sehr erfolgreich. Laut der Entertainment Software Association (ESA) (2015) besitzen über 50 Prozent aller U.S. Haushalte eine Spielekonsole; in Deutschland sind es 26 Prozent aller Haushalte (Statistisches Bundesamt 2017), bei Haushalten mit Kindern sind es sogar 61 Prozent (Statistisches Bundesamt 2014). Zudem stehen oben genannte Eigenschaften nach Dörner et al. (2016) im Einklang mit den allermeisten Lerntheorien und Instruktionsansätzen. Es erscheint daher sinnvoll, Spiele auch für andere Zwecke als zur Unterhaltung zu verwenden.

2.3.1 Serious Games als didaktisches Mittel

Als Serious Games werden Spiele bezeichnet, die neben der Unterhaltung ein weiteres, definiertes Ziel verfolgen (siehe zum Beispiel Cheng, et al. 2015; Deterding, et al. 2011; Girard, Ecalle und Magnan 2013). Dieses Ziel kann die Vermittlung bestimmter Inhalte sein, aber auch beispielsweise die Gesundheitsförderung oder die Rekrutierung von Personal im Armeebereich. Ritterfeld et al. definieren den Begriff als „any form of interactive computer-based game software for one or multiple players to be used on any platform and that has been developed

with the intention to be more than entertainment" (Ritterfeld, Cody und Vorderer 2009, 6). Über die genaue Definition des Begriffs Serious Game herrscht bisher kein Konsens. So bezeichnen einige Autor*innen ausschließlich Videospiele als Serious Games, welche explizit für einen „sinnvollen Zweck" entwickelt wurden (Girard, Ecalle und Magnan 2013), andere bezeichnen jedes Videospiel als Serious Game, welches nicht komplett zur Unterhaltung dient (Muratet, Torguet und Jessel, et al. 2009). Nach Dörner et al. (2016) werden in einigen Definitionen auch Serious Games nicht durch die Intention des Entwickelnden bestimmt, sondern durch die des Spielenden. Damit könnte jedes Videospiel ein Serious Game sein, solange der Spielende es für einen anderen Zweck als die Unterhaltung verwendet. Einige Autor*innen schlagen vor, Serious Games von Serious Gaming zu unterscheiden, wobei Serious Games Spiele für Lehrzwecke bezeichnet und Serious Gaming jegliche Lehr-Software, welche im Spielkontext steht. So würde nach Deterding et al. (2011) eine Software zum Designen von Spielen in den Serious Gaming Bereich fallen.

Als Mittel für die Lehre werden Serious Games seit 2010 vermehrt eingesetzt (Cheng, et al. 2015), gleichwohl die Ursprünge solcher Spiele Anfang der 2000er Jahre im Militärbereich liegen (Deterding, et al. 2011). Serious Games werden sowohl unterrichtsbegleitend als Lehrwerkzeuge als auch in alleinstehenden Lernumgebungen eingesetzt, wobei letztere die Mehrheit bilden (Cheng, et al. 2015). Exemplarisch werden einige Serious Games im Folgenden vorgestellt.

Das Spiel **„Program your Robot"** von Kazimoglu et al. (2012) ist ein Serious Game zum Üben grundlegender Programmierkonstrukte, wie konditionaler Logik und Schleifen. Die Lernenden sollen dabei vor allem lernen, wie Algorithmen aufgebaut und Programme systematisch auf Fehler untersucht werden können. Ziel des Spiels ist es, einem Roboter bei der Flucht zu helfen. Dafür muss der Spielende einen Lösungsalgorithmus bauen, sodass der Roboter über eine Reihe von Plattformen fliehen kann. Hindernissen und Gegnern muss dabei ausgewichen werden. Der Lösungsalgorithmus wird entwickelt, indem der Spielende vorgefertigte Befehle in beliebiger Reihenfolge in Slots von Funktionen zieht. Diese Slots sind begrenzt, sodass bei späteren Aufgaben komplexere Konstrukte wie Schleifen benutzt werden müssen. Die Entwicklung findet nicht textuell, sondern grafisch statt. Glaubt der Spielende eine Lösung gefunden zu haben, kann er den Play-Button klicken, um die Befehle ausführen zu lassen. Diese Befehle sind untergliedert in Aktions- und Programmierungsbefehle. Dabei steuern Aktionsbefehle

den Roboter. Programmierungsbefehle stellen Loops, konditionale Entscheidungsfindung und Funktionsaufrufe dar. Insgesamt gibt es sechs Level, wobei in jedem Level ein anderes Programmierkonstrukt vermittelt wird. Um ein Level abzuschließen, muss der Spieler einen Teleporter erreichen. Zufällig erzeugte Belohnungsitems steigern den Punktestand des Spielenden. Mit einem zusätzlichen Debug-Button kann der Lernende seine Befehlsabfolgen auf Fehler und mögliche Probleme untersuchen lassen.

„Prog&Play“ ist ein Serious Game von Muratet et al. (2009, 2011) zur Übung von Programmierung. Das Serious Game ist begleitend zum Unterricht durch einen Lehrenden gedacht und enthält keine Lehrinhalte. Prog&Play baut auf dem Open Source Projekt Kernel Panic auf, einem Multiplayer-Strategiespiel. Nach Erkenntnissen der Autor*innen ist dieser Spieltyp der am weitesten verbreitete. Das Spielgeschehen findet im Inneren eines gehackten Computersystems statt, über welches der Spielende in mehreren Leveln mit dem Steuern von Elementen wie Bits und Bytes wieder die Kontrolle erlangen muss. Gesteuert wird das Spiel über eine zur Verfügung gestellte API. Der Code der Game-Engine ist nicht einsehbar und der Lernende kann sich auf die zu schreibende Erweiterung konzentrieren. Die Integration des Codes findet dynamisch zur Laufzeit statt, ein expliziter Compile-Prozess ist nicht nötig. Verwendet werden kann prinzipiell jede Programmiersprache, welche eine C-Library ansprechen kann. Die Programmierung findet hier somit textuell statt. Nach Abschluss der Kampagne können die Spielenden eigene Codes schreiben und gegen andere Spielende antreten. Die Autor*innen erhoffen sich, dass die Lernenden selbstständig weiterspielen und so ihre Programmierfähigkeiten verbessern. Außerdem schlagen die Autor*innen einen Organisationsplan für den Kurs vor, welcher die konkrete Auflistung der Lehrziele und den dazugehörigen Zeitplan enthält.

Schäfer et al. (2013) stellen eine **kollaborative, Multitouch Lernumgebung** vor. Ziel ist es, gemeinsam an einem Multitouch-Gerät mathematische Logik zu lernen und zu üben. Nach den Autor*innen führt Kollaboration zu mehr Interaktion und einer besseren Aufnahme des Lernstoffes. Während weder Kazimoglu et al. (2012) noch Muratet et al. (2009, 2011) Lerntheorien und didaktische Modelle als Grundlage für ihre Systeme angeführt haben, orientierten sich Schäfer et al. (2013) für die Umsetzung ihres Programms am Modell von Gagné (1992) sowie am Felder-Silverman Lernmodell (1988). Letzteres gibt unter anderem Aufschluss über unterschiedliche Lernstile. Es wurde versucht, die unterschiedlichen Lernstile beim Design der Lerneinheiten zu berücksichtigen. Zudem haben Felder und Silverman (1988) festgestellt, dass

grafische Darstellungen von Informationen für die meisten Personen verständlicher sind als textuelle Varianten. Typische mathematische Logik-Algorithmen werden daher in der Lernumgebung grafisch dargestellt. Die Lernumgebung besteht aus vier Abschnitten, die kollaborativ ausgeführt werden können:

- Animierte Demonstration von grundlegenden Regeln (Animationsmodus),
- Ausführen des Algorithmus durch die Lernenden mit sofortigem Feedback (Feedbackmodus),
- Beantworten von Quiz-Fragen und Zusammenstellen von Elementen zu einer Sequenz (Quizmodus und Formularmodus),
- Kollaborativ miteinander eine Aufgabe gegen die Zeit oder gegeneinander lösen (Free play).

Während die ersten drei Abschnitte sich vor allem der Vermittlung der Inhalte widmen, kann mit dem Free Play Abschnitt die davor gelernte Theorie in einer spielerischen Aufgabe praktisch angewandt werden. In diesem Abschnitt können Punkte durch schnelle Reaktionen gesammelt werden. Somit stellt das von Schäfer et al. (2013) vorgestellte Programm ein Hybrid aus Lernumgebung und Serious Game dar. Der höchste Punktestand wird auf der Highscore-Liste für alle Spielenden veröffentlicht. Diese Konkurrenz soll intrinsische Motivation erzeugen, sodass die Lernenden angeregt werden, bessere Werte zu erreichen. Das Spiel wird dadurch länger gespielt und die Lerninhalte verinnerlicht.

Wie erwähnt, wurden nur bei der Lernumgebung von Schäfer et al. (2013) verwendete didaktische Modelle und Theorien aufgeführt. Dies scheint ein genereller Trend zu sein: In 53 untersuchten Veröffentlichungen zu Serious Games von 2002 bis 2013 stellten Cheng et al. (2015) fest, dass nur die Hälfte der Autor*innen den Einsatz solcher Spiele lerntheoretisch begründen. Zudem wurde die Effizienz von Serious Games empirisch noch nicht vollständig nachgewiesen: So stellen Boot et al. (2011) diverse methodische Mängel in Studien fest, welche einen positiven Effekt von Spielen beschreiben. Auch Girard et al. (2013) thematisieren in ihrer Meta-Analyse zu Serious Games methodische Mängel bei der Untersuchung der betrachteten Artikel. Serious Games scheinen ein hilfreiches Werkzeug für Lernzwecke zu sein, jedoch haben bisherige Evaluationsstudien noch keine hinreichende Begründung für den Erfolg des

Einsatzes von Serious Games geliefert. Studien, welche den Einsatz von Serious Games theoretisch begründen, berufen sich meist auf den Konstruktivismus und die Theorien von Vygotski (Cheng, et al. 2015). Diese Theorien scheinen den konzeptionellen Ansatz von Serious Games angemessen beschreiben zu können.

2.3.2 Gamification

Im Zusammenhang mit Serious Games steht der Begriff Gamification. Während der Begriff Serious Games vollwertige Spiele bezeichnet, welche nicht ausschließlich zur Unterhaltung gedacht sind, beschreibt Gamification nach Deterding et al. (2011) die Verwendung charakteristischer Designelemente von Spielen in einem Nicht-Spiele-Kontext. Damit soll, ähnlich zu Serious Games, die Motivation bei der Benutzung eines Programms gesteigert werden (Hamari, Koivisto und Sarsa 2014). Seaborn und Fels (2015) geben an, dass hauptsächlich die Benutzerbeteiligung gesteigert werden soll. Eine gesteigerte Motivation scheint dafür Voraussetzung zu sein. Nach Analyse mehrerer Veröffentlichungen zu Gamification haben Hamari et al. (2014) folgende Elemente als häufig vertreten erkannt:

- Punkte und Bestenlisten,
- Errungenschaften und Auszeichnungen,
- Level, Herausforderungen und Ziele,
- Feedback,
- Fortschrittsanzeigen,
- Rahmengeschichten.

Die Programmier-Lernplattform Code.org ist ein Beispiel für den Einsatz von Gamification. Mit dieser können Kinder das Programmieren erlernen, indem sie grafisch Anweisungsblöcke zusammenziehen. Nach Beendigung einer jeweiligen Lektion können sie den Fortschritt innerhalb des aktuellen Kurses einsehen; hierfür werden sie mit virtuellen Trophäen belohnt. Die Qualität ihrer Lösung entscheidet darüber, ob eine bronzene, silberne oder goldene Trophäe vergeben wird. Diese Preise können die Benutzenden auf ihrer Profilseite gesammelt betrachten. Zusätzlich wird ein farblich kodiertes Feedback zur Lösung gegeben; eine perfekte Lösung wird als grün markiert. Ist die Lösung richtig, aber nicht optimal, so wird sie hellgrün angezeigt.

Fehlerhafte Lösungen werden violett markiert (Kalelioğlu 2015). Neben der Lehre findet Gamification in vielen weiteren Bereichen wie Gesundheit, Nachhaltigkeit und sozialen Netzwerken Anwendung (Seaborn und Fels 2015).

Spiele und Anwendungen mit Spieldesign wie Code.org lassen sich keinesfalls immer klar abgrenzen. So hängt die Einordnung einer Anwendung als Spiel oder „gamifizierte“ Anwendung oftmals auch vom Blickwinkel des jeweiligen Betrachters ab (Deterding, et al. 2011). Deterding et al. (2011) schlagen fünf Level von Gamedesignelementen vor, in welche sich die oben genannten einordnen lassen und welche zur Gamification benutzt werden können:

- Häufig benutzte, konkrete Designmuster (zum Beispiel Abzeichen und Bestenlisten),
- Spielmechaniken (zum Beispiel Zeitlimits und Runden),
- Gamedesign-Prinzipien und Heuristiken (zum Beispiel klar definierte Ziele),
- Konzeptionelle Modelle von Spielkomponenten (zum Beispiel Fantasie, Neugierde, Herausforderungen),
- Gamedesign-spezifische Praktiken und Prozesse (zum Beispiel ein spielerisches Design).

Somit stellen die Elemente von Hamari et al. (2014) nur einen Ausschnitt aller möglichen Elemente dar, welche 4 Gamification benutzt werden können. Hamari et al. (2014) schlussfolgern aus den betrachteten Veröffentlichungen, dass Gamification als Mittel in den meisten Fällen positive Ergebnisse erzielt hat. Zu einer ähnlichen Schlussfolgerung kommen Seaborn et al. (2015).

2.4 Grundlegende Programmierkonstrukte

Wie bereits erwähnt, stellt neben dem Erlernen der Programmiersprache vor allem die Problemlösefähigkeit ein wichtiges Lernziel dar. Es stellt sich die Frage, welche Programmierkonstrukte und -paradigmen grundlegend sind und anfangs gelehrt werden sollten. Kazimoglu et al. (2012) führen an, dass dazu vor allem konditionale Logik, das Bauen von Algorithmen, Debugging sowie Simulation gehören. Muratet et al. (2009) nennen ebenfalls grundlegende Programmierkonstrukte, wie Schleifen und Bedingungen, sowie Algorithmen. Darüber hinaus bezeichnen sie aber auch Datenstrukturen und Rekursion als wichtige Lerninhalte. Insbesondere die grundlegenden Programmierkonstrukte seien für Anfänger*innen oftmals schwer zu verstehen, sodass hier der Fokus gesetzt werden sollte. Dabei kann es helfen, den Lernenden zu

Beginn keine vollständigen Programme schreiben zu lassen, sondern kleinere Lücken zu füllen. So kann sich der Benutzende auf die Anwendung des Lerninhalts konzentrieren (Stevenson und Wagner 2006).

2.5 Syntaktische und semantische Überprüfung

Bisher noch nicht im Detail betrachtet wurde die Ergebnisauswertung: Sowohl bei Serious Games als auch bei Lernumgebungen mit Gamedesignelementen muss syntaktisch und semantisch verifiziert werden, ob das gewünschte Ergebnis erzielt wird und welche Fehler es gegebenenfalls gibt. Diese Überprüfungen fallen beispielsweise bei Code.org relativ einfach aus. Syntaktische Fehler können nicht entstehen, da sich nur passende Blöcke verbinden lassen. Semantisch wird eine Aufgabe scheinbar bereits als korrekt angesehen, wenn die Spielfigur einen bestimmten Checkpoint erreicht oder einen bestimmten Zustand einnimmt.[1] Diese Überprüfungen scheinen für die Aufgaben von Code.org ausreichend, sind aber für komplexere Probleme ungeeignet. Eine komplexere Semantiküberprüfung wurde beispielsweise von Jeuring et al. (2014) für Ask-Elle vorgeschlagen. Ask-Elle ist ein interaktiver Tutor, mit welchem schrittweise Haskell-Programme geschrieben werden können. Der Tutor gibt dabei Feedback. Um zu überprüfen, ob ein Programm die Aufgabenstellung erfüllt, wird zunächst versucht, die Lösung auf hinterlegte Modelllösungen zu reduzieren. Sollte das nicht funktionieren, wird die Lösung des Benutzenden mit dem Programm QuickCheck von Claessen und Hughes (2000) gegen mehrere definierte Eigenschaften getestet und es werden Gegenbeispiele gesucht. So sind die Aufgaben semantisch vollständig gelöst und dem Benutzenden werden Probleme durch Gegenbeispiele anschaulich erklärt.

Zusammenfassend lässt sich festhalten: Die vorgestellten Lerntheorien bilden die Grundlage für didaktische Modelle, welche in Lernumgebungen Anwendung finden. Kernaspekte für den Lernerfolg sind dabei die Motivierung des Lernenden sowie die Anpassung des Lernprozesses an die unterschiedlichen Bedürfnisse. Außerdem spielt die Kreativität eine wichtige Rolle im Lernprozess. Für die Einführung neuer Inhalte sind außerdem Instruktionen von Bedeutung. Für diese Aspekte wurden unterschiedliche, didaktische Modelle beschrieben. Ein mögliches

[1] Am Beispielkurs 2 der Code.org-Plattform (https://code.org/) zuletzt beobachtet am 16.04.2018 bei Level 19 von Stunde 13: Mittels verschachtelten Aufrufen soll ein Bild gezeichnet werden. Tatsächlich ist es aber unerheblich, ob Schleifen eingesetzt werden oder nicht. Das Ergebnis wird in beiden Fällen als korrekt akzeptiert.

Lehrmittel stellen dabei Serious Games und Lernumgebungen mit Gamedesignelementen dar, welche Lerninhalte unterschiedlicher Fachrichtungen spielerisch vermitteln. Speziell für die Informatik und angelehnte Fächer werden mit solchen Programmen oftmals Programmierparadigmen und -konstrukte eingeführt. Nachdem diese notwendigen Grundlagen dargestellt wurden, soll im Folgenden eine Serious Games basierte Lernumgebung für Programmierung vorgestellt werden.

3 Konzept der Lernplattform PLAY & CODE

Nachdem lerntheoretische und didaktische Grundlagen sowie Spiele als didaktisches Mittel erläutert wurden, wird im Folgenden die Serious Games basierte Online-Lernplattform PLAY & CODE vorgestellt, mit welcher Programmierparadigmen spielerisch gelehrt werden können. Um das Programm vorzustellen, werden zunächst Prämissen bzw. Anforderungen an die Lernplattform benannt und begründet. Im Anschluss wird die grundsätzliche Struktur, d.h. der Aufbau des Systems erläutert.

3.1 Anforderungen an die Lernplattform

Didaktisch gründe das Lernsystem[2] auf dem im Kapitel 2.2.1 vorgestellten ARCS-Modell von Keller (1987) sowie den „First Principles of Instruction“ von Merrill (2002). Das Modell von Merrill basiert auf einer umfangreichen Untersuchung verschiedener vormals vorgestellter Instruktionsmodelle und versucht die Essenz in einem Modell zu vereinen. Die vielzitierte und renommierte Arbeit Merrills fußt auf dem problembasierten Lernansatz, welcher sich als sehr effektiv erwiesen hat. Da – wie bereits in Kapitel 2 beschrieben – die Motivation ein wichtiger Faktor für erfolgreiches Lernen ist, kommt zusätzlich das ARCS-Modell von Keller zur Verwendung. Der Einsatz eines Serious Game als Lehrmittel trägt zum Spaß beim Lernen bei und ruft intrinsische Motivation hervor. Da Programmieraufgaben auf verschiedene Wege gelöst werden können, wird somit Kreativität beim Lernen befördert.

Im Gegensatz zu anderen spielebasierten Lernplattformen, wie „Program your Robot“ von Kazimoglu et al. (2012) oder der Lernplattform Code.org, wird hier keine grafische, sondern die textuelle Programmiersprache JavaScript eingesetzt. Nach Merrill (2002) sollen in den Übungen realitätsnahe Probleme gelöst werden und das erlernte Wissen in den Alltag integrierbar sein. Keller (1987) gibt die Vermittlung der Relevanz des Lerninhalts als eine der vier Prämissen für Motivation an. Relevanz und Alltagstauglichkeit können womöglich für die in den genannten Programmen gelehrten Programmierkonstrukte angenommen werden, für die grafische Sprache gilt dies jedoch offensichtlich nicht. So listet der monatlich aktualisierte TIOBE Programming Community Index (2018) beispielsweise keine einzige grafische Sprache unter den Top zehn Programmiersprachen. Ähnliche Ergebnisse liefern die Seiten RedMonk Programing

[2] Die Begriffe Lernsystem und Lernplattform werden in diesem Kapitel synonym verwendet.

Language Rankings (O'Grady 2017), sowie der PYPL Index (Carbonnelle 2018), welche ebenfalls die Popularität von Sprachen vergleichen. Ihre Relevanz für den professionellen Einsatz scheint daher nachgeordnet zu sein. Wünschenswert ist jedoch, dass Lernende ihr neu erworbenes Wissen unmittelbar nach dem Lernkurs anwenden können.

Im Gegensatz dazu erfreut sich JavaScript quellenübergreifend steigender Popularität und ist durchweg unter den ersten zehn Sprachen gelistet. Das Ranking von RedMonk listet JavaScript sogar an erster Stelle. Durch Laufzeitumgebungen wie Node.js eignet sich die Sprache zudem nicht nur für Web Frontend-Auftritte, sondern auch für Backend-Programmierung. Auch Desktop-Anwendungen können mithilfe von Frameworks wie Electron oder NW.js mit JavaScript programmiert werden. Die Sprache ist also für eine Vielzahl von Szenarien einsetzbar und ihre Relevanz durchaus gegeben. JavaScript eignet sich daher sehr gut als zu verwendende Programmiersprache. Zudem lässt sich JavaScript im Gegensatz zu vielen anderen populären Sprachen, wie Java oder Python, ohne zusätzlichen Aufwand direkt in Internetbrowsern ausführen. Die Darstellung von grafischen Inhalten und das Ausführen dynamischer Codes sind somit technisch unkompliziert.

3.2 Struktur des Systems

Die Lernplattform ist als ein Online-Mehrbenutzersystem konzipiert. Über eine Registrierungsmaske können sich Interessierte selbstständig registrieren und anschließend einloggen. Jedem Anwendenden wird eine Rolle zugewiesen, mit der bestimmte Berechtigungen verbunden sind. „User“ können alle Standardaktionen des Systems, wie zum Beispiel die Verwaltung des eigenen Profils oder die Durchführung eines Kurses, ausführen. Administratoren*innen sind zusätzlich berechtigt, neue Kurse zu erstellen und alle angelegten Benutzerprofile zu verwalten. Es können beliebig viele Kurse angelegt werden. Nach dem erstmaligen Login muss der Benutzende zunächst einen kurzen Fragebogen zu seinen Programmier-Vorerfahrungen beantworten. Dabei wird mit Likert-Skalen von 1 (keine Erfahrung) bis 5 (sehr erfahren) nach Programmiererfahrungen im Allgemeinen und Erfahrungen mit JavaScript im Speziellen gefragt. Diese Informationen werden zur Errechnung eines Erfahrungswertes verwendet, welcher die Länge der Lehrtexte beeinflusst. Hierauf wird später detaillierter eingegangen.

Der Benutzende kann aus einer Kursübersicht seine Kurse frei auswählen. Die Kurse werden mit einem kurzen Text und einem Thumbnail beschrieben. Jeder Kurs ist in mehrere Lektionen

gegliedert, welche der Reihe nach abgeschlossen werden müssen. Der Gesamtfortschritt des Kurses ist jederzeit in der Kursübersicht einsehbar. Hat sich der Benutzende für einen Kurs entschieden, kann er direkt mit der ersten bzw. der von ihm zuletzt bearbeiteten Lektion starten oder eine Übersicht aller Lektionen des Kurses öffnen. Lektionen werden erst freigeschalten, wenn die vorherige Lektion beendet wurde. Fortschritte werden persistent in einer Datenbank und auf dem Server gespeichert, sodass Kursfortschritte auch nach einem erneuten Login weiter vorhanden und einsehbar sind.

Jeder Kurs stellt ein Videospiel dar, mit welchem eine in sich abgeschlossene Geschichte erzählt wird. Die Lektionen des Kurses sind die einzelnen, aufeinanderfolgenden Abschnitte dieses Spiels.[3] In jeder Lektion wird in einer animierten Intro-Sequenz die Geschichte weitererzählt und der Protagonist vor ein neues Problem gestellt. Hiermit soll sich der Benutzende in das Spielgeschehen integriert fühlen. Eine interessante Rahmengeschichte ist nach Dignan (2011) ein Hauptaspekt von Spielen.

Auf die Intro-Sequenz folgt die Lern- und Programmierphase, in welcher der Benutzende neue Inhalte erlernt. Dafür muss zunächst die Coding-Ansicht geöffnet werden, in welcher ihr oder ihm die Lehrtexte, die Aufgabe und die Programmiereingabe angezeigt werden. Die Texte können auch Bilder und formatierten Text enthalten (vgl. Abbildung 1).

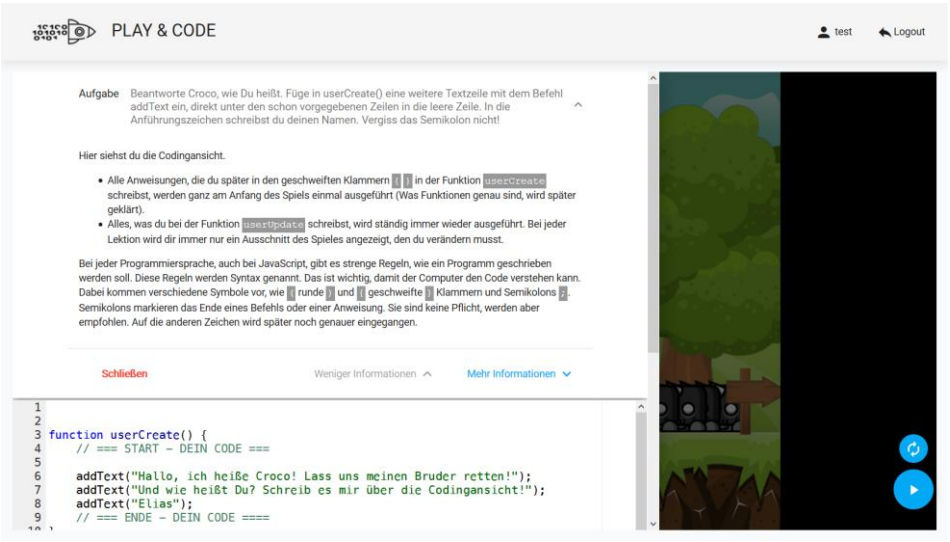

Abbildung 1: Screenshot aus dem Kurs "Crocos Bruder" mit geöffneter Coding-Ansicht.

[3] Tatsächlich sind die definierten, inhaltlichen Eigenschaften von Kursen und Lektionen nur Empfehlungen, welche für den hier implementierten Beispielkurs umgesetzt wurden. Spätere, von Administratoren hochgeladene Kurse können von diesen Eigenschaften abweichen.

Der Text unterteilt sich in folgende Absätze:

1. Grundlegende Informationen zum jeweiligen Lernthema werden bereitgestellt.
2. Informationen werden weiter ausgeführt und Konstrukte detaillierter erklärt.
3. Konkrete Positiv- und Negativbeispiele werden aufgelistet und weiterführende Ressourcen verlinkt.

3.3 Adaptivität der Lehrtexte

Der Lehrtext wird an den Erfahrungswert des Benutzenden adaptiert: Je nachdem, wie hoch der errechnete Erfahrungswert des Benutzenden ist, wird ihm ein unterschiedlich großer Ausschnitt des gesamten Textes angezeigt. Der jeweilige Erfahrungswert des Benutzenden ist auf einer Skala von 1 bis 15 angesiedelt. Bei einem Erfahrungswert unter 6 werden alle drei Abschnitte angezeigt: Der Benutzende benötigt detaillierte Informationen und konkrete Beispiele, um die Aufgabe lösen zu können. Bei einem Erfahrungswert zwischen 6 bis 10 werden zwei Abschnitte angezeigt. Bei Werten darüber wird lediglich der erste Abschnitt dargestellt: Der Benutzende kennt bereits die Paradigmen aus anderen Sprachen und benötigt lediglich knappe Informationen über die JavaScript spezifische Verwendung. Meint der Benutzende mehr oder weniger Informationen zu benötigen, kann er den Text weiter ein- bzw. ausklappen. Der Erfahrungswert errechnet sich aus nachfolgenden Punkten:

- Informationen des Fragebogens (als initialer Startwert)
    - Allgemeine Programmiererfahrung (1-5) (doppelt gewichtet)
    - JavaScript Programmiererfahrung (1-5)
- Benötigte Versuche in der aktuellen Lektion
    - Bei mehr als fünf Versuche wird der Erfahrungswert um 5 verringert.
    - Bei zwei oder weniger Versuchen wird der Erfahrungswert um 3 erhöht.
- Manuelle Veränderung des Ausschnitts
    - Wird mehr angezeigt, wird der Erfahrungswert um 1 verringert.
    - Wird weniger angezeigt, wird der Erfahrungswert um 1 erhöht.

Der Erfahrungswert dient lediglich zur internen Berechnung und wird dem Benutzenden nicht angezeigt. Da sowohl nach erstmaligem Login als auch fortlaufend auf der Grundlage des Fragebogens Daten zum Benutzenden gesammelt und analysiert werden, handelt es sich hierbei

nach Vandewaetere et al. (2011) um einen mikro-adaptiven Ansatz. Damit soll Benutzenden mit Vorerfahrungen zu viel Text erspart bleiben und Programmieranfänger*innen die benötigten Informationen zur Verfügung gestellt werden.

3.4 Framework des Serious Games

Nachdem Lehrtexte und Aufgabenstellung vom Benutzenden gelesen wurden, kann er/sie mit dem neu gewonnenen Wissen eine Lösung für das Problem programmieren und den Protagonisten so zum Erfolg verhelfen. Der für die bereits vorhandene Spielmechanik benötigte Code wird dabei in – für den Benutzenden nicht sichtbare – Dateien ausgelagert. Die Benutzenden sehen lediglich die für sie bestimmte Datei UserCode, welche befüllt werden muss, um die Aufgabe zu lösen. Damit soll sichergestellt werden, dass der Benutzende nicht von der Codefülle überfordert ist, sondern sich auf das Wesentliche konzentrieren kann. Die Datei UserCode enthält initial zwei Funktionen: userCreate und userUpdate. userCreate wird einmalig bei der Initialisierung des Spiellevels ausgeführt, während userUpdate fortlaufend in der Spielschleife einmal pro Bilddarstellung (frame) aufgerufen wird. Der Benutzende kann so Codes für die zwei wichtigsten Zustände des Spiels schreiben. Die Funktionen können leer sein oder aber schon vorgegebene Codes vom Kursautor /der Kursautorin enthalten, die vom Benutzenden ergänzt werden müssen.

Nachdem der Benutzende seinen Lösungscode geschrieben hat, kann er/sie auf „Play“ drücken. Sollten syntaktische Fehler**,** wie zum Beispiel falsch gesetzte Klammern, vorhanden sein, werden diese festgestellt und die Ausführung wird unterbrochen. Fehler werden dann mit Zeilennummern und entsprechender Fehlermeldung angezeigt. Andernfalls wird der programmierte Code serverseitig gespeichert, bereitgestellt und das so modifizierte Level gestartet. Der Benutzende kann dabei das Problem auf verschiedene, kreative Weise lösen, solange die semantischen Anforderungen erfüllt sind. Diese werden vom Kursautor oder von der Kursautorin definiert. Einerseits können Werte und Zustände überprüft werden, beispielsweise die Position des Spielenden oder der Zustand der aktuellen Spielwelt. Andererseits kann auch die Implementierung des Programmierenden auf bestimmte Schlagwörter untersucht werden. Das ist besonders nützlich, wenn überprüft werden soll, ob tatsächlich das zu lernende Programmierkonstrukt angewandt wurde. Beispielsweise lässt sich das Ergebnis einer for-Schleife womöglich auch erreichen, indem der Inhalt der Schleife um die Anzahl der angegebenen Ausführungen dupliziert wird. Dann wurde das inhaltliche Ziel zwar erreicht, das Lernziel wurde

jedoch vermutlich verfehlt. Die definierten Tests werden jeweils nach der userCreate- und userUpdate-Funktion des Benutzenden ausgeführt. Der genaue Ablauf des Frameworks wird im folgenden Diagramm detailliert dargestellt (Abbildung 2).

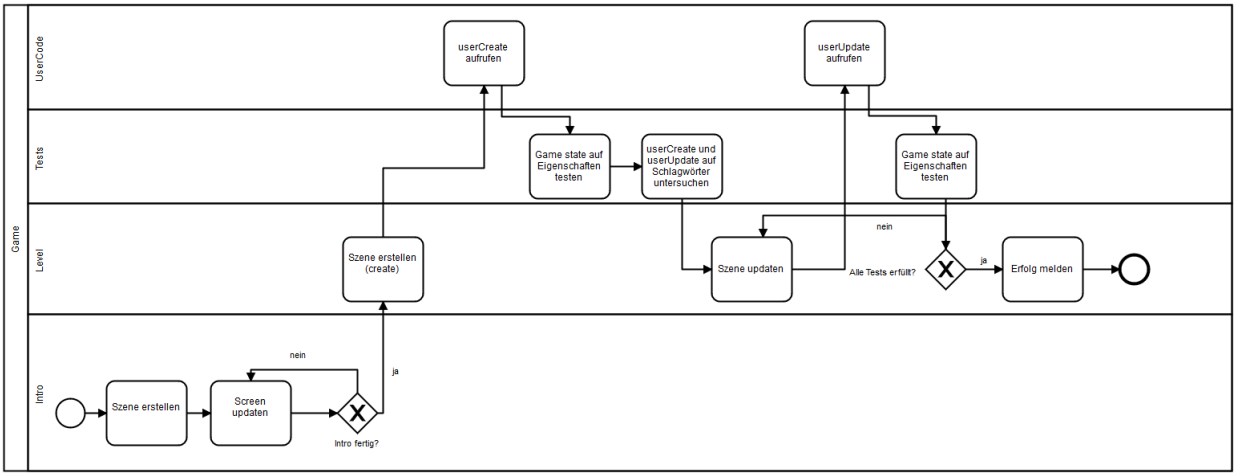

Abbildung 2: Ablauf eines Spiels und einzelne Zustände

3.5 Gamification-Elemente

Durch das Lösen weiterer Probleme durch zusätzliche, optionale Programmierung erscheinen in einigen Leveln Diamanten, welche der Benutzende einsammeln kann und welche sich auf sein Ergebnis auswirken: Zum Abschluss jeder Lektion werden virtuelle Trophäen vergeben. Je nachdem ob und wie viele Diamanten eingesammelt wurden, wird eine Bronze-, Silber oder Goldmedaille vergeben. In Leveln ohne Diamanten werden Goldmedaillen vergeben. Im Benutzerprofil können gewonnene Trophäen betrachtet werden. Zusätzlich wird ein Prozentsatz von Spieler*innen angezeigt, die diese Trophäe ebenfalls erhalten haben. Diese Gamification-Elemente sollen zum spielerischen Wettstreit zwischen den Anwendenden führen und dadurch die Motivation zusätzlich steigern. Zusammen mit dem Serious Game deckt das System somit die wichtigsten Game-Elemente nach Hamari et al. (2014) und Dignan (2011) ab:

- Auszeichnungen,
- Mehrere Level,
- Feedback,
- Fortschrittsanzeigen (Kursübersicht),
- Rahmengeschichte,
- Ziele.

3.6 Erweiterbarkeit durch API

Das System kann, neben kleineren Administrationsmöglichkeiten über das eigene Benutzerprofil, vor allem über eine REST-API (Representational State Transfer API) modifiziert, erweitert und administriert werden. APIs, welche den von Fielding (2000) vorgestellten Architekturstil REST verwenden, haben den Vorteil, dass sie eine einheitliche Schnittstelle darstellen und die Verwendung somit simpel ist. Jede Information (wie ein Kurs, eine Lektion, aber auch Lektionen eines bestimmten Kurses), die über eine eindeutige URL adressiert und identifiziert werden kann, wird als Ressource bezeichnet. Durch die Umsetzung des HATEOAS Prinzip (Hypermedia as the engine of application state) wird zusätzlich eine einheitliche Darstellung der Ressourcen und deren Beziehungen sichergestellt: Subressourcen einer Ressource X werden direkt in der Darstellung von X verlinkt, sodass der Anwendende diesem Link nur folgen muss. Ressourcen unterstützen zudem die HTTP-Methoden wie GET und POST als Operationen zum Abruf und zur Modifikation. Aufgrund dieser Prinzipien benötigt der Benutzende keinerlei Vorwissen über die API, sondern kann direkt mit der Schnittstelle interagieren.

3.7 Beispielkurs Crocos Bruder

Zur Demonstration und Evaluierung der Lernplattform wurde der Beispielkurs „Crocos Bruder“ implementiert, mit welchem grundlegende Programmierparadigmen von JavaScript vermittelt werden. Der Benutzende spielt dabei ein 2D-Adventurespiel und muss dem Protagonisten Croco bei der Rettung seines Bruders helfen:

> **Geschichte**
>
> Es ist ein sonniger Tag. Croco und sein Bruder sind gerade in ihrem Garten als plötzlich eine Horde dunkler Gestalten erscheint und beide angreift. Croco hat keine Chance und fällt in Ohnmacht. Als er wieder zu sich kommt, sieht er, dass sein Bruder entführt wurde. Verzweifelt überlegt er, wie er gegen einen so mächtigen Gegner ankommen kann. Da kommt der Dorfälteste herbeigeeilt und übergibt Croco die Codeeingabe, mit welcher er die Monster besiegen kann. Ermutigt zieht Croco los. Mithilfe seines neuen Werkzeugs will er seinen Bruder retten.

Der Kurs richtet sich vor allem an Programmier- und JavaScript-Anfänger*innen. In jedem Level wird Croco vor ein neues Problem gestellt, welches durch ein neu erlerntes Programmierkonstrukt lösbar ist. Jede Lektion baut auf den Lerninhalten der vorherigen Lektionen auf. Die

Level wurden grafisch jeweils unterschiedlich entworfen, um ein abwechslungsreiches und interessantes Spielerlebnis zu schaffen. Im unteren Bildschirmbereich wird dem Benutzenden die Gesamtanzahl der zu lösenden Teilprobleme und der aktuelle Fortschritt angezeigt (siehe Abbildung 3).

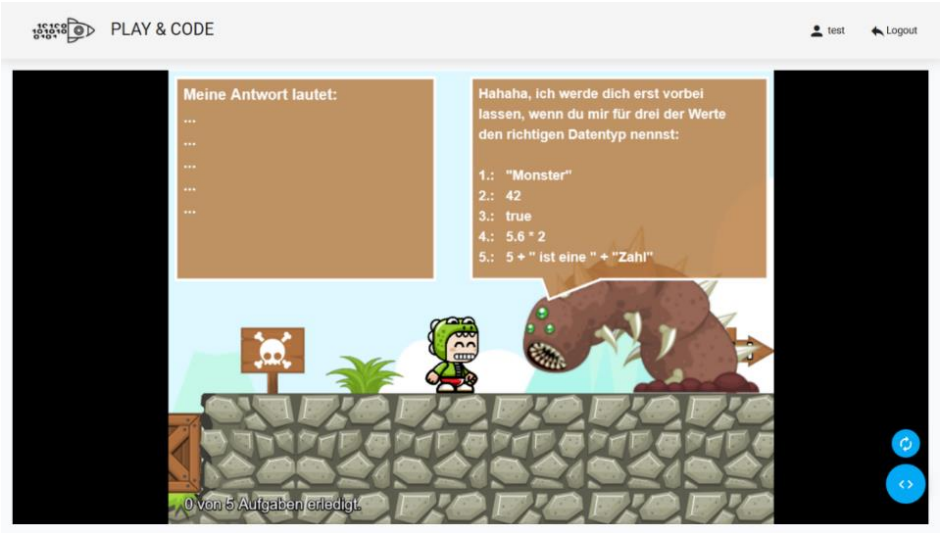

Abbildung 3: Screenshot aus dem dritten Level von "Crocos Bruder"

Folgende JavaScript-Themen werden in „Crocos Bruder“ behandelt:

1. Syntax: Was ist unter Syntax zu verstehen und wie ist ein JavaScript Programm aufgebaut?
2. Kommentare: Nutzen von Kommentaren
3. Datentypen: strings, numbers, booleans
4. Variablen: Definition und Verwendung
5. If-Abfragen: Bedingte Ausführungen
6. If-Else
7. Komparatoren
8. logische Operatoren
9. Funktionen
10. For-Schleifen

Die Themenwahl, deren Reihenfolge und die Lehrtexte sind an das Lehrbuch „JavaScript for Kids: A Playful Introduction to Programming“ (Morgan 2014) angelehnt und für die Lernplattform angepasst. Die Themen decken zudem einen Teil der von Kazimoglu et al. (2012) und

Muratet et al. (2009) identifizierten Programmiergrundlagen, wie konditionale Logik, ab. Morgan beschreibt in seinem Lehrbuch anschaulich und einfach die JavaScript-Grundlagen. Es richtet sich an Kinder, kann aber ebenso von Erwachsenen zum Lernen verwendet werden. Auf amazon.de wurde das Buch mit 4.3 von 5 Sternen (Stand Januar 2018) sehr positiv bewertet. Es eignet sich daher gut zur Beschreibung dieses Kurses.

3.8 Zusammenfassung des Kapitels

In diesem Kapitel wurden das Konzept des Serious Games basierten Lernsystems PLAY & CODE sowie die Anforderungen und Prämissen an dieses vorgestellt: Als didaktische Modelle werden das ARCS-Modell von Keller (1987) sowie die „First Principles of Instruction“ von Merrill (2002) verwendet. Damit wurden Motivation, Kreativität und Spaß beim Lösen als wichtige Anforderungen identifiziert. Da die Aufgaben und Inhalte zusätzlich praktisch relevant sein sollten, wurde JavaScript als geeignete zu lehrende Sprache ausgewählt.

Das Lernsystem kann unterschiedliche Kurse mit jeweils mehreren Lektionen enthalten. Jeder Kurs stellt ein abgeschlossenes Spiel dar. Mit jeder Lektion wird der Held des Spiels vor ein neues Problem gestellt. Dieses kann gelöst werden, indem die Benutzer*innen das neu erworbene Wissen aus dieser Lektion anwenden. Beispielhaft wurde der Kurs „Crocos Bruder“ vorgestellt, mit welchem einige JavaScript-Grundlagen gelehrt werden.

4 Implementierung von PLAY & CODE

Die Implementierung der Lernplattform PLAY & CODE unterteilt sich in drei Module:

- Datenbank
- Backend
- Frontend

In der Datenbank werden alle Daten, wie Kurse, Lektionen und User gespeichert. Das Backend verwaltet diese Daten und stellt sie mit einer REST-API bereit. User werden authentifiziert und Anfragen autorisiert. Das Frontend stellt die Webseite dar. Angefragte Daten werden hier visualisiert und die Benutzenden können Kurse durchführen. Diese Module werden im Folgenden detailliert beschrieben, um einen Einblick in die technische Funktionsweise von PLAY & CODE zu erlangen.

4.1 Entitäten und die Datenbankarchitektur

In PLAY & CODE können Benutzende aus einem Kursangebot wählen. Diese Kurse bestehen jeweils aus unterschiedlichen Lektionen, welche abgeschlossen und bewertet werden können. Benutzende füllen zu Beginn der Benutzung des Systems einen Fragebogen zu ihren Vorerfahrungen aus. Aus diesem und aus kontinuierlich erfassten Daten während der Benutzung wird der Erfahrungswert errechnet, mit welchem die Länge des angezeigten Lehrtextes angepasst wird. Dieser Wert wurde im vorherigen Kapitel bereits genau beschrieben. Jedem Benutzenden wird außerdem eine bestimmte Benutzerrolle zugeordnet. Um diese Daten festhalten zu können, wurden folgende Entitätstypen und deren Datenfelder herausgearbeitet:

- User
    - Eindeutiger Benutzername
    - Name, Vorname, Mailadresse
    - Passwort
    - Erfahrungswert (user_knowledge)
    - Benutzerrolle (user_role)
    - Liste von abgeschlossenen Lektionen mit der jeweils erreichten Trophäe (user_completed_lessons)

    - Liste mit der Anzahl der Versuche und der investierten Zeit für jede begonnene Lektion (user_lesson_data)
    - Daten vom initialen Fragebogen (self_assessment)
- Kurs
    - Titel, Beschreibung
    - Enabled-Flag
    - Eine Liste von zugewiesenen Lektionen
    - Eindeutiger String-Identifier
- Lektion
    - Titel, Beschreibung, Aufgabenstellung, Lehrtexte
    - Verweise auf die vorhergehende und auf die nachfolgende Lektion
    - Verweis auf den zugeordneten Kurs
    - Enabled-Flag
    - Eindeutiger String-Identifier
    - Liste von Benutzerbewertungen für diese Lektion (lesson_review)
    - Liste von Benutzern, die diese Lektion abgeschlossen haben

Mit den Enabled-Flags können Lektionen und Kurse temporär deaktiviert werden. Das ist nützlich, wenn beispielsweise ein größerer inhaltlicher Fehler festgestellt wird. Die Unterscheidung zwischen der Liste user_completed_lessons und der Liste user_lesson_data wurde getroffen, da im Gegensatz zu user_completed_lessons in user_lesson_data Daten zu jeder begonnenen, aber nicht notwendiger Weise abgeschlossenen, Lektion festgehalten werden. Diese Daten sind für die Evaluierung wichtig und sollen dabei helfen zu verstehen, welche Lektionen für die Benutzende schwieriger waren als andere. Die String-Identifier stellen den Namen des Ordners auf der Festplatte dar, in welchem die jeweiligen Ressourcen für Kurse und Lektionen abgelegt sind. Hierauf wird im Abschnitt zum Backend näher eingegangen. Die exakten Tabellendefinitionen und deren Beziehungen untereinander können in der folgenden Abbildung 4 betrachtet werden.

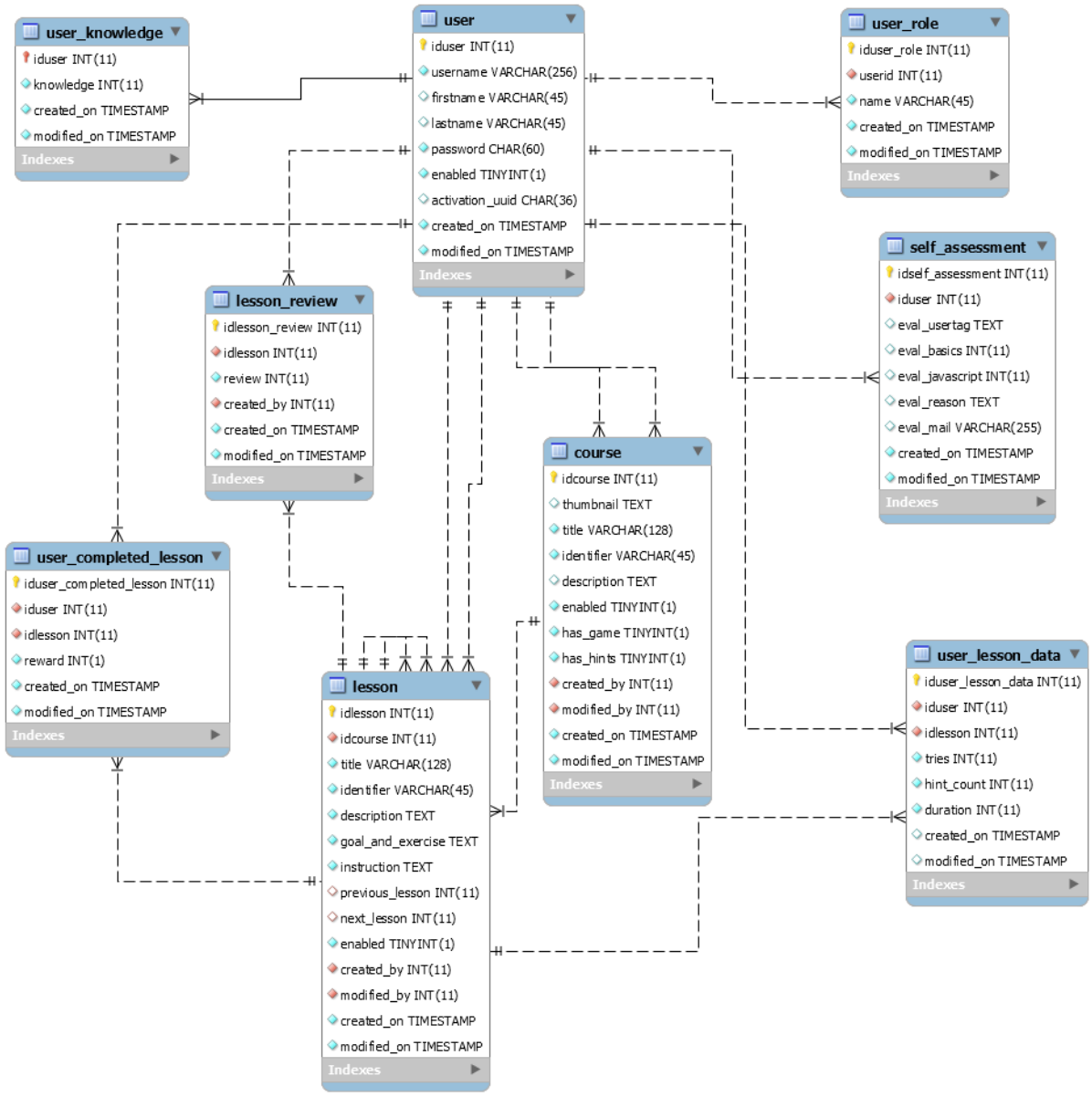

Abbildung 4: Darstellung der Entitäten und deren Beziehungen in der Datenbank

4.2 Backend

Mithilfe des Backends werden die oben genannten Daten verwaltet und über eine REST API bereitgestellt. Es wurde mit dem Java Framework Spring und Spring Boot 1.5 entwickelt. Das Spring Framework stellt eine Reihe von Modulen, wie Spring Security, zur Verfügung, welche die Implementierung von Standardfunktionalitäten, hier insbesondere von Webservices, vereinfachen. Spring Boot stellt einige Konfigurationen für das Framework bereit und ermöglicht es einen Tomcat-Server direkt einzubetten.

Um auf Daten zugreifen zu können, müssen sich Benutzende zunächst authentifizieren. Dafür wird einmalig mit der HTTP-Authentifizierungsmethode Basic Authentication das eigene Benutzerprofil abgefragt. Die Verbindung ist per SSL verschlüsselt, sodass Daten, wie der Benutzername und das Passwort, bei der Übertragung nicht ausgelesen werden können.

Das Backend überprüft dann mittels Benutzername und Passwort die Anfrage. Dafür muss das Passwort zunächst verschlüsselt werden, da die Passwörter nicht im Klartext in der Datenbank abgelegt, sondern mit bcrypt verschlüsselt sind.[4] Die Passwörter sind somit auch in der Datenbank nicht auslesbar und sicher abgespeichert. Sollte die Authentifizierung erfolgreich verlaufen sein, wird dem Benutzenden im Header x-auth-token ein Authentifizierungstoken übergeben, welches für alle weiteren Anfragen in dieser Session verwendet werden kann. Passwort und Benutzername müssen so nur einmal übertragen werden. Die Tokens werden in einer Redis-Datenbank gehalten und haben eine Gültigkeit von 30 Minuten. Danach wird der Benutzende automatisch ausgeloggt. Im Frontend wird dies überprüft.

Abgesehen von einigen wenigen Ausnahmen wird bei jeder API-Anfrage die Autorisierung des Benutzenden überprüft. Je nach Anfrage wird dafür eine unterschiedliche Benutzerrolle benötigt. Benutzende mit der Userrolle „User“ dürfen nur Informationen hinzufügen und verändern, welche sie selbst betreffen. Beispielsweise kann jeder Anwendende seine eigenen Profildaten verändern. Daten von anderen Usern können nur Administrator*innen verändern. Für jeden Entitätstyp sind Schnittstellen zur Erstellung, Veränderung und Löschung, sowie zum Lesen von Einträgen hinterlegt (CRUD). Über die Adresse /logout kann sich der Benutzende frühzeitig ausloggen und den Server anweisen die Session zu invalidieren.

Hochgeladene Dateien werden in einem von dem Serveradministrator bzw. der Serveradministratorin frei wählbaren Ordner auf dem Server gespeichert. Der Pfad zu diesem Ordner und weitere Parameter, wie die Datenbankanbindung, können in einer Konfigurationsdatei spezifiziert werden.[5] Kursdateien liegen dabei im Unterordner „resources“ jeweils separat in einem

[4] bcrypt wird als Verschlüsselung für Passwörter vom Open Web Application Security Project (OWASP) empfohlen. OWASP ist eine Non-Profit-Organisation, welche zum Ziel hat, die Sicherheit von Software zu erhöhen. Siehe: https://www.owasp.org/index.php/Cryptographic_Storage_Cheat_Sheet (aufgerufen am 30.04.2018 um 15:16 Uhr)

[5] Diese „application.properties“-Datei muss in einem config-Ordner im gleichen Verzeichnis wie die Anwendung liegen.

Ordner. Zu jeder Lektion eines Kurses existiert ein Unterordner und eine html-Datei. Kursordner, Lektionen-Ordner und Lektionen-HTML-Dateien sind jeweils mit ihren eindeutigen String-Identifiern benannt.

Codeabgaben von Benutzer*innen werden im Ordner „users“ abgelegt. Dort wird für jeden Benutzer ein extra Ordner mit Benutzername erstellt, in welchem zu den jeweiligen Kursen und Lektionen die JavaScript-Dateien abgelegt werden. Die Struktur von Kurs- und Lektionsordnern ist dabei im Benutzerordner gleich. Wird für die gleiche Lektion nochmals ein Code hochgeladen, wird der vormals hochgeladene Code in den Ordner „old“ verschoben. Die vollständige Ordnerstruktur ergibt sich somit wie folgt (Abbildung 5):

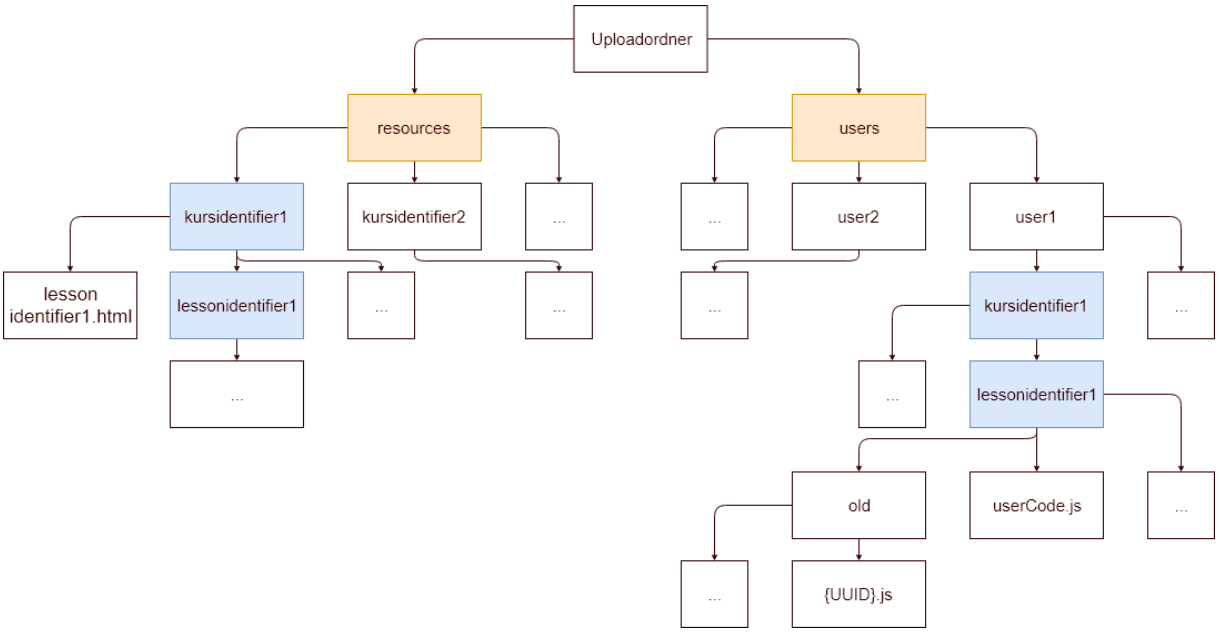

Abbildung 5: Ordnerstruktur für hochgeladene Dateien

Neue Kurse können von Administratoren über die API-Adresse /resources/files als zip-Datei hochgeladen werden. Diese zip-Datei muss die Lektionsordner und HTML-Dateien enthalten. Zusätzlich wird eine json-Datei benötigt, welche die einzupflegenden Datenbankinformationen zu den Lektionen und dem Kurs enthält. Diese json-Datei wird nach erfolgreichem Upload und Dateneinpflege gelöscht.

Benutzer*innen können über die API-Adresse /resources/users unter Angabe vom Code sowie dem Kurs- und Lektionsidentifier Abgaben hochladen, welche automatisch ihrem Benutzeraccount zugeordnet werden. Die Daten können über die Adresse /api auch visualisiert abgerufen und modifiziert werden.

4.3 Frontend

Das Frontend wurde mit dem Framework Angular 5 von Google entwickelt. Neben einigen Vereinfachungen ermöglicht das Framework, Abschnitte des Frontends in sogenannte Komponenten zu zerlegen. Diese Komponenten bestehen jeweils aus CSS, HTML und TypeScript-Dateien. Sie können an mehreren Stellen wiederverwendet werden und andere Komponenten aufrufen. So entsteht ein kompakter, übersichtlicher Code. Durch die Verwendung von TypeScript können außerdem Features von ECMAScript 6, wie zum Beispiel Lambda-Funktionen, bereits jetzt schon eingesetzt werden. Da ECMAScript 6 noch nicht vollständig von gängigen Browsern unterstützt wird, übersetzt ein Compiler den TypeScript-Code dann in JavaScript-Code. Als Design-Framework wird das, ebenfalls von Google entwickelte, Material Design eingesetzt.

Das Frontend besteht aus mehreren, sogenannten Routen. Eine Route in Angular gibt an, welche Komponente unter welchem Pfad angezeigt wird. Folgende Routen bestehen (Tabelle 1):

Pfad	**Inhalt**
/register	Registrierungs-Bildschirm
/legal	Rechtliche Hinweise
/privacy	Datenschutzhinweise
/licenses	Verwendete Lizenzen
/dashboard*	Übersicht über die Kurse
/list/:id*	Aufgelistete Lektionen eines Kurses mit :id
/lesson/:id*	Lektionsansicht mit Lektion :id
/profile*	Benutzerprofil mit Trophäen

*Tabelle 1: Frontend-Routen und deren Bedeutung. Pfade, welche mit einem * markiert sind, werden erst erreichbar, wenn sich der Benutzer eingeloggt hat.*

Ruft eine Benutzerin oder ein Benutzer die Webseite auf, so wird zunächst der login-Bildschirm angezeigt. Beim Klick auf den Login-Button werden die eingetragenen Daten per GET-Anfrage an /api/users/search/findByUsername?username={username} gesendet. Sollte der

Login-Vorgang erfolgreich verlaufen sein, werden die Benutzerdetails und das Authentifizierungstoken aus dem Header im localStorage gespeichert.[6] Alle weiteren Anfragen werden, wie im Abschnitt zuvor beschrieben, mit diesem authentifiziert.

In der Lektionsansicht wird die HTML-Datei dieser Lektion als iFrame eingebunden. Diese Datei stellt den Spielinhalt dar. Sobald der Code abgegeben wurde, muss das Spiel neugeladen werden. Außerdem muss die Lernplattform reagieren, sobald sich der Spielzustand verändert. Um dies bewerkstelligen zu können, müssen Daten zwischen dem Spiel und der Lernplattform ausgetauscht werden. Diese Kommunikation findet über den localStorage des Browsers statt. So kann der Zustand auch nach einem Schließen des Browsers wiederhergestellt werden, solange die Session noch nicht abgelaufen ist (Tabelle 2).

Felder	**Bedeutung**	**Wert**
game.success	Wird vom Spiel auf 1 gesetzt, sobald das Level abgeschlossen wurde.	0 oder 1
game.reward	Gibt an, je nachdem wie viele Aufgaben gelöst wurden, welche Trophäe zu vergeben ist. Wird vom Spiel gesetzt.	1: Bronze, 2: Silber, 3: Gold
game.skipIntro	Wird von der Lernplattform auf 1 gesetzt, sobald die Coding-Ansicht geöffnet wird. Außerdem setzt das Spiel auf 1, sobald das Intro betrachtet wurde. Das Spiel überspringt dann bei weiteren Starts das Intro.	0 oder 1
game.userCode-Path	Wird von der Lernplattform gesetzt, sobald ein Code abgegeben wurde. Das Spiel wird diesen Code dann bei der nächsten Ausführung einbinden.	https:// playandcode.de/ re- sources/users/...

Tabelle 2: LocalStorage-Felder

[6] Als localStorage wir eine Schlüssel-Wert-Tabelle bezeichnet, welches auch nach Schließen des Browsers erhalten bleibt. Siehe https://developer.mozilla.org/en-US/docs/Web/API/Window/localStorage (aufgerufen am 30.04.2018 um 15:17 Uhr)

Sobald die Felder game.success auf 1 und game.reward gesetzt sind, zeigt die Lernplattform einen Dialog mit der Trophäe an, welcher den Benutzenden darüber informiert, dass er diese Lektion abgeschlossen hat.

Das Frontend ist sowohl auf Deutsch als auch auf Englisch verfügbar und kann über https://playandcode.de bzw. https://playandcode.com aufgerufen werden.

4.4 Spielframework

Um mit der Lernplattform kompatibel zu sein, müssen Serious Games bestimmte Anforderungen erfüllen:

- Es müssen vereinbarte localStorage-Felder berücksichtigt werden.
- Der Benutzercode muss vom Rest des Codes getrennt sein.
- Der Benutzercode muss austauschbar sein.

Optimaler Weise sollten die Spiele außerdem die Struktur von Abbildung 2 berücksichtigen, um eine klare Strukturierung des Codes sicherzustellen und einheitliche Abläufe der Spiele untereinander zu gewährleisten:

- Das Spiel sollte einen Introzustand, in welchem die Geschichte fortgeführt wird, und einen Levelzustand haben, welcher veränderbar ist.
- Es sollte eine Methode userCreate, die in der Initialisierung des Levels aufgerufen wird, und eine Methode userUpdate existieren, die im Updateprozess aufgerufen wird.
- Es sollte eine Test-Klasse vorhanden sein, welche den Spielzustand sowohl nach der Initialisierung als auch fortlaufend im Updateprozess überprüft. Außerdem sollte der Code optional auf Schlagwörter untersucht werden können.

Um das Erstellen von Kursen unter Berücksichtigung der oben genannten Aspekte zu vereinfachen, wurden einige JavaScript-Klassen und Interfaces geschrieben, welche zusammen ein Framework bilden und universal eingebunden sowie benutzt werden können.[7] Mit diesen wird

[7] Das Framework setzt aktuell noch die 2D-Spieleengine Phaser als Game-Engine voraus (der Kurs „Crocos Bruder“ wurde mit Phaser implementiert) und benötigt die JavaScript-Bibliothek jQuery, um den Code dynamisch einzubinden. In zukünftigen Versionen werden diese Abhängigkeiten entfallen.

auch eine einheitliche Benutzeroberfläche, z.B. hinsichtlich der Aufgabenanzeige und den Fehlernachrichten, sichergestellt.

5 Methodik

Um die eingangs vorgestellte Forschungsfrage: Können mithilfe eines Serious Games basierten Lernsystems Motivation und Lernerfolg für grundlegende Programmierkenntnisse gesteigert werden? beantworten zu können, wurde das Lernsystem PLAY & CODE an mehreren Schulen in Berlin mit Schüler*innen unterschiedlichen Alters und mit wenig JavaScript-Erfahrungen evaluiert. Quantitative Daten wurden schriftlich durch standardisierte Online-Fragebögen und zudem automatisiert durch das System erfasst. Zudem wurden qualitative Daten mittels der Methode der teilnehmenden Beobachtung während der Durchführung gesammelt.

Um den Lernerfolg der Schüler*innen feststellen zu können, wurde ein Pre-/Posttest durchgeführt. Hierbei wurden Fragen (16 Fragen im Pretest, 17 Fragen im Posttest)[8] zu folgenden grundlegenden Themenbereichen gestellt, welche auch im evaluierten PLAY & CODE Kurs „Crocos Bruder“ behandelt wurden:

- Kommentare,
- Variablen,
- Datentypen,
- If-Else-Abfragen,
- For-Schleifen,
- Funktionen,
- Komparatoren und
- logische Operatoren.

Diese Themenwahl orientiert sich an den von Kazimoglu et al. (2012) und Muratet et al. (2009) identifizierten Programmiergrundlagen. Im Pretest wurden zudem folgende Benutzerdaten abgefragt:

- Alter,
- Geschlecht und
- Spielhäufigkeit pro Woche bzw. Monat.

[8] Komplette Fragebögen im Anhang.

Es wurde außerdem pro Themenbereich eine Möglichkeit zum Überspringen der Fragen angeboten, falls der Benutzende über keine Vorerfahrungen verfügte.

Nach Abschluss des Posttests beantworteten die Schüler*innen zudem einen Evaluationsfragebogen, in dem sie folgende Eigenschaften der Lernplattform bewerten und Fragen beantworten sollten:

- Bedienbarkeit,
- Verständlichkeit der Lerntexte,
- Schwierigkeitsgrad der Aufgaben,
- Länge der Aufgaben,
- Spaß beim Lernen,
- Motivation durch Trophäen,
- Interesse an weiteren Lerninhalten,
- Gesamtbenotung von PLAY & CODE.

Zudem hatten die Schüler*innen die Möglichkeit, mit eigenen Worten, d.h. textbasiert, zu beschreiben, was ihnen gut und was ihnen nicht gut am Spiel gefallen hat.

Die Fragebögen wurden online mit der Umfragewebsite SurveyMonkey erstellt und dort von den Schüler*innen beantwortet. Der durchschnittliche Zeitaufwand für die Beantwortung lag beim Pretest und beim Evaluationsfragebogen bei je zwei Minuten, beim Posttest bei sieben Minuten. Am Pretest nahmen 136 und am Posttest 101 Schüler*innen teil. Der Evaluationsfragebogen wurde von 108 Schüler*innen ausgefüllt. Während der Benutzung von PLAY & CODE wurden, wie im Kapitel 4 bereits erwähnt, folgende Daten je Schüler/Schülerin für die Evaluation erfasst:

- Bearbeitungsdauer und Versuchsanzahl pro Lektion,
- Gewonnene Trophäe pro Lektion,
- Bewertung der Lektion,
- Codeabgaben,
- Grund, warum der Benutzende bisher noch nicht bzw. nicht mehr programmiert,
- Selbsteinschätzung Programmierwissen allgemein / JavaScript.

Bei PLAY & CODE haben sich im gesamten Untersuchungszeitraum 140 eindeutige Benutzer*innen registriert.[9] Um die Datensätze der unterschiedlichen Fragebögen und der Systemdaten verknüpfen zu können, sollten die Schüler*innen jeweils ihr Namenskürzel in ein entsprechendes Textfeld schreiben. Dieses Kürzel setzte sich aus ihrem Vornamen und dem ersten Buchstaben ihres Nachnamens zusammen (z.B. MaxM). In der Lernplattform musste dieses Feld beim erstmaligen Login gefüllt werden.

Kontaktiert wurden alle dreizehn Partnerschulen der Einrichtung „Informatik und Gesellschaft und Didaktik der Informatik“ des Instituts für Informatik der Humboldt-Universität sowie die katholische Schule St. Marien auf Empfehlung einer Kommilitonin. Von insgesamt 14 angeschriebenen Schulen haben sich vier Schulen zu einer Teilnahme bereit erklärt (siehe Tabelle 3). Die Evaluation wurde vom 15.02.2018 bis zum 09.03.2018 an folgenden Schulen durchgeführt:

[9] Diese Zahl enthält auch Benutzer*innen, die nicht an der Befragung teilgenommen hatten und unaufgefordert oder auf Empfehlung von Anderen die Seite besuchten.

Datum	Schule	Teilnehmeranzahl	Klasse	Schultyp
15.02.	Melanchton-Gymnasium	20 (w: 4, m: 16)	11. und 12. Klasse	Gymnasium
19.02. / 05.03.	Katholische Schule St. Marien	11 (w: 2, m: 9)	8. Klasse	Integrierte Sekundarstufe
19.02.	Katholische Schule St. Marien	10 (w: 1, m: 9)	12. Klasse	Gymnasium
23.02. / 02./ 09.03.	Friedrich-Ebert-Gymnasium	16 (w: 8, m: 8)	9. Klasse	Gymnasium
26.02.	Käthe-Kollwitz-Gymnasium	7 (w: 2, m: 5)	8. Klasse	Gymnasium
27.02.	Katholische Schule St. Marien	14 (w: 3, m: 10, fehlend: 1)	9. Klasse	Integrierte Sekundarstufe
28.02.	Friedrich-Ebert-Gymnasium	10 (w: 1, m: 9)	12. Klasse	Gymnasium
28.02.	Käthe-Kollwitz-Gymnasium	11 (w: 2, m: 9)	7. Klasse	Gymnasium
06.03.	Friedrich-Ebert-Gymnasium	7 (w: 1, m: 6)	10. Klasse	Gymnasium
08.03.	Friedrich-Ebert-Gymnasium	11 (w: 2, m: 9)	9. Klasse	Gymnasium

Tabelle 3: Übersicht über Termine, Schulen und Teilnehmende; w=weiblich, m=männlich

Als Anreiz wurden unter den Schüler*innen, die alle Fragebögen ausgefüllt hatten, fünf Amazon-Gutscheine im Wert von 20 Euro verlost. Diese wurden am 15.03.2018 an die Schüler*innen verschickt.

Nach der Erhebung der Daten wurden diese aus der MySQL-Datenbank von PLAY & CODE sowie aus SurveyMonkey exportiert und mit dem Software-Programm SPSS (Statistical Programme for Social Sciences) ausgewertet. Benutzer*innen, die älter als 20 Jahre zum Zeitpunkt der Erhebung waren oder im Pretest bereits mehr als 80 Prozent aller Fragen korrekt beantwortet hatten, wurden herausgefiltert, um die Daten und die Ergebnisse nicht zu verzerren. Um die Differenz zwischen Pre- und Posttest festzustellen, wurde zu den Antworten jedes Themengebiets in beiden Datensätzen ein gerundeter Durchschnittswert ermittelt. Hatte

der/die Schüler*in die Hälfte oder mehr der Antworten richtig beantwortet, wurde das Themengebiet als korrekt bearbeitet markiert, andernfalls als falsch. Aus den so entstandenen acht Variablen pro Datensatz wurden Summen berechnet, die über den Anteil von „richtig“ beantworteten Themenbereichen Aufschluss geben. Die Differenz stellt den Wissenszuwachs dar.

Da die Teilnehmeranzahl pro Fragebogen unterschiedlich ist und die Angabe des Namenkürzels teilweise fehlt, konnten nicht alle Fälle miteinander verknüpft werden. Die Zusammenführung von Pretest, Posttest und den Evaluationsfragebogen ergab 50 eindeutige, bereinigte Fälle. Auch wenn im Ergebnis der Zusammenführung deutlich weniger Fälle zur Auswertung geeignet sind, so ist die Reduktion notwendig, um aussagekräftige Ergebnisse zu erzielen.

Um den statistischen Einfluss der Reduktion der Fälle zu prüfen, wurde die Differenz A der Mittelwerte aller Ergebnissummen im Pre- und Posttest (als Durchschnitts-Wissenszuwachs, über alle Benutzer*innen) und anschließend der Mittelwert B über jeden Wissenszuwachs pro Benutzer*in im zusammengeführten Datensatz berechnet. Der Vergleich beider Mittelwerte (A und B) ergibt eine Differenz von lediglich sieben Prozent. Das heißt, im zusammengeführten Datensatz (mit 50 Fällen) kann lediglich ein sieben Prozent größerer Wissenszuwachs festgestellt werden, was einer einzelnen zusätzlich korrekt beantworteten Frage entspricht. Für die folgende Analyse kann dieser Unterschied aufgrund der geringen Abweichung zugunsten valider Daten und Ergebnisse vernachlässigt werden. Die folgenden Analysen beziehen sich deshalb auf die 50 vollständigen Fälle des Pre- und Posttests sowie des Evaluationsfragebogens.

6 Ergebnisse der Datenauswertung

Um eine hinreichende Antwort auf die Forschungsfrage geben zu können, orientierte sich die Datenauswertung und Ergebnisdarstellung an folgenden Kategorien:

- Qualität und Adaptivität der Lehrtexte,
- Motivation, Kreativität, Spaß,
- Lernerfolg,
- Zielgruppe,
- Usability der Oberfläche.

Bevor auf die empirischen Ergebnisse näher eingegangen wird, soll zunächst die Stichprobe von 50 Schüler*innen beschrieben werden (siehe Tabelle 4).

Variable	Ausprägung	Häufigkeit	Prozent
Geschlecht			
	Männlich	39	78%
	Weiblich	11	22%
Alter			
	12 Jahre	4	8%
	13 Jahre	7	14%
	14 Jahre	4	8%
	15 Jahre	8	16%
	16 Jahre	9	18%
	17 Jahre	14	28%
	18 Jahre	4	8%
Spielhäufigkeit			
	Selten	4	8%
	Mehrmals im Monat	1	2%
	Mehrmals pro Woche	26	52%
	Jeden Tag	19	38%
Schultyp			
	Integrierte Sekundarstufe	10	20%
	Gymnasium	40	80%

Tabelle 4: Eigenschaften der Stichprobe, N = 50

Wie zu sehen, reicht die Altersspanne der Schüler*innen von 12 bis 18 Jahre. Bemerkenswert ist zudem, dass 90 Prozent der Schüler*innen mehrmals pro Woche oder häufiger Videospiele spielen. Auffällig ist ebenfalls der hohe Anteil männlicher Teilnehmer.

6.1 Qualität und Adaptivität der Lehrtexte

Die Lehrtexte sind bei PLAY & CODE essentiell für den Lernerfolg des Benutzenden. Sie sind die Quelle für Informationen über das zu lernende Thema. Es stellt sich daher die Frage, wie die Qualität die Lehrtexte des Evaluationskurses „Crocos Bruder“ eingeschätzt wird und wie sich diese auf den Lernerfolg der Benutzenden auswirkt. Hängt womöglich auch das Alter des Benutzenden mit der Verständlichkeit der Texte und deren Erfolg zusammen?

PLAY & CODE passt zudem die Länge der Lehrtexte an den geschätzten Wissensstand des Benutzenden an. Es sollte daher überprüft werden, wie die Textlänge bewertet wurde. Folgende Hypothesen wurden untersucht:

- Je besser die Textlänge von den Benutzenden bewertet wurde, desto besser wurden die Texte angepasst.
- Je älter die Benutzenden sind, desto besser wurden die Texte verstanden und desto bessere Ergebnisse wurden erzielt.
- Wurden die Texte von den Benutzenden als zu lang bewertet, hatte das auch Einfluss auf eine eher schlechte Bewertung der Verständlichkeit der Texte.
- Wurde die Textverständlichkeit als schlecht bewertet, dann wurden auch die Aufgaben als zu schwer bewertet.
- Je schlechter die Textverständlichkeit bewertet wurde, desto schlechter wurde auch der eigene Lernzuwachs bewertet.

Die Datenanalyse ergab, dass die Textlänge von 72 Prozent der Schüler*innen als „genau richtig“, von 24 Prozent als „zu lang“ und von 4 Prozent als „zu kurz“ bewertet wurde (vgl. Abbildung 6).

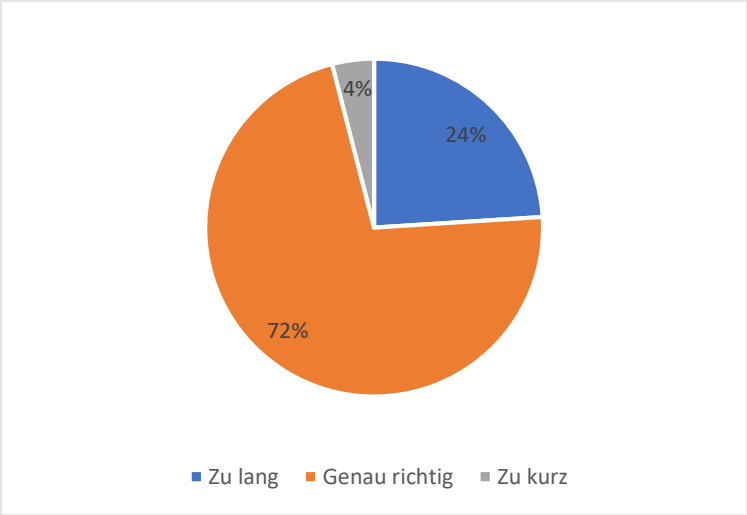

Abbildung 6: Bewertung der Länge der Lerntexte
Bemerkung: Die Frage hierzu lautete: „Wie fandest du die Länge der Lerntexte?“, N = 50

Die Befunde machen deutlich, dass für den Großteil der Benutzenden die Texte die richtige Länge hatten. Das könnte einen Hinweis darauf geben, dass die Schätzung des Wissensstands des Benutzenden und die Adaptivität der Lerntexte durch das System das gewünschte Ergebnis erzielten. Für immerhin 24 Prozent waren die Texte jedoch zu lang. Für diese Gruppe sollten möglicherweise die Texte optimiert und in der Gesamtlänge gekürzt als auch eine feinstufigere Adaptivität der Texte eingeführt werden.

Nachdem die Textlänge untersucht wurde, wird nun die Bewertung der Textverständlichkeit betrachtet: 20 Prozent der Benutzenden haben die Textverständlichkeit mit der Schulnote „sehr gut“, 62 Prozent mit der Schulnote „gut“ bewertet. 10 Prozent empfanden die Textverständlichkeit als „befriedigend“. Jeweils 4 Prozent gaben die Note „ausreichend“ und „mangelhaft“ (Abbildung 7).

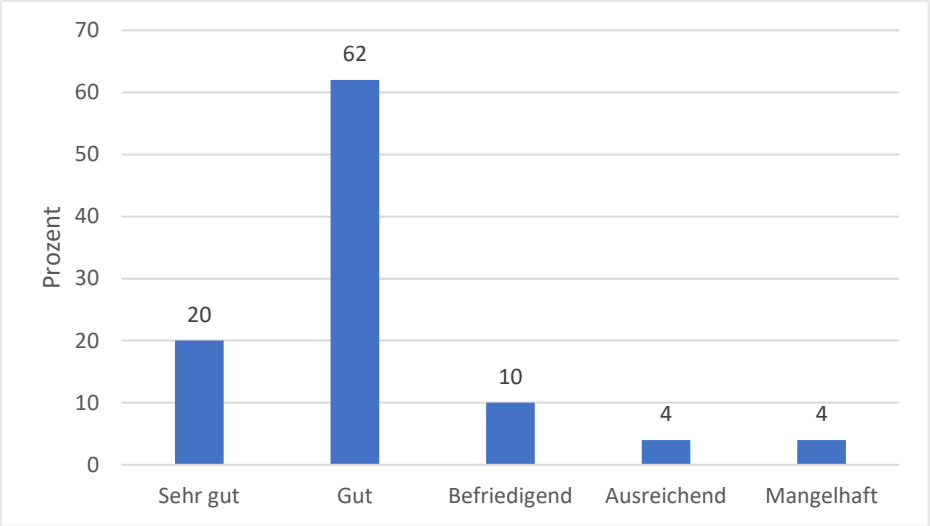

Abbildung 7: Bewertung der Textverständlichkeit
Bemerkung: Die Frage hierzu lautete: „Wie verständlich fandest du die Lerntexte?", N = 50

Anders als zunächst angenommen, hat die Textlänge keinen Einfluss auf die Verständlichkeit. Die beiden Variablen sind voneinander unabhängig und korrelieren nicht. Mit einer Durchschnittsbewertung von 2,1 wurde die Textverständlichkeit überwiegend als „gut" bewertet. Die bereits genannten Textoptimierungen können diesen Wert womöglich noch verbessern.

Ein weiterer Befund macht deutlich, dass mit steigendem Alter die Summe korrekt beantworteter Fragen im Pretest steigt (Korrelation nach Pearson mit Signifikanz von 0,002). Da außerdem angenommen werden kann, dass mit steigendem Alter auch die kognitiven Fähigkeiten (wie z. B. die Übertragbarkeit vorhandenen Wissens bzw. die Vorerfahrungen) zunehmen, war zu erwarten, dass die Textverständlichkeit und die Summe an Trophäen steigt. [10] Die Analyse macht jedoch deutlich, dass eine signifikante Korrelation zwischen dem Alter bzw. der Vorerfahrung und dem Textverständnis, sowie zwischen dem Alter bzw. der Vorerfahrung und der Gesamtsumme an Trophäen nicht besteht. Vorerfahrungen und Alter haben somit keinen Einfluss auf die Bewertung der Textverständlichkeit. Die Texte scheinen sich daher für alle Benutzer*innen gleichermaßen zu eignen. Dies könnte an der Strukturierung der Texte sowie deren Anpassung an den Wissensstand des Benutzenden liegen: Je weniger Erfahrungen der Benutzende mitbringt, desto mehr Text wird ihm oder ihr angezeigt. Mit steigender Länge wird der Text detaillierter und enthält mehr Beispiele (1. Absatz: Kurze Beschreibung, 2. Absatz: Details,

[10] Höherwertige Trophäen wurden höher gewichtet.

3. Absatz: konkrete Beispiele). Somit wird eine gleichbleibende Textverständlichkeit bei sinkendem Erfahrungswert durch steigenden Detailierungsgrad und steigender Länge des Textes erreicht.[11]

Zusammenfassend lässt sich festhalten: Für die meisten Schüler*innen hatten die Lehrtexte die richtige Länge. Das lässt den Schluss zu, dass die Adaptivität der Lehrtexte aufgrund des errechneten Wissensstands des Benutzenden korrekt funktionierte und zum gewünschten Ergebnis führte. Auch die Textverständlichkeit wurde überwiegend mit „gut“ bewertet.

6.2 Motivation, Kreativität und Spaß

Motivation und Spaß sind, wie im Kapitel 2 beschrieben, wichtige Faktoren für den Lernerfolg. Keller (1987) stellt mit seinem ARCS-Modell vier Aspekte vor, wie Motivation erzeugt werden kann (siehe Kapitel 2):

- Die Aufmerksamkeit des Lernenden muss erzeugt werden. (Attention)
- Die Relevanz des Lerninhalts durch Kontextbeziehung sollte verdeutlicht werden. (Relevance)
- Zuversicht sollte erzeugt werden, indem Aufgaben fordernd, aber machbar sind. (Confidence)
- Ein Lernerfolg muss sichtbar sein, sodass sich Zufriedenheit des Lernenden einstellen kann. (Satisfaction)

Auch die Möglichkeit, Probleme kreativ lösen zu können, ist eine Prämisse für erfolgreiches Lernen. In PLAY & CODE sollen diese Faktoren vor allem durch den Einsatz von Serious Games und Trophäen als Gamification-Element erzeugt werden. Der Erfolg dieser Komponenten wurde durch folgende Hypothesen überprüft:

- Je mehr Spielversuche benötigt wurden und je länger die Bearbeitungszeit dauerte, desto weniger Spaß hat es den Benutzenden gemacht.
- Je größer die Summe erreichter Trophäen war, desto mehr Spaß hat es den Benutzenden gemacht.

[11] Mögliche Korrelationen zwischen Textverständlichkeit und Bewertung des Schwierigkeitsgrads der Aufgaben sowie zwischen Textverständlichkeit und Lernzuwachs wurden aufgrund der geringen Varianz von 0,827 und einer Standardabweichung von 0,909 der Variable „Textverständlichkeit“ nicht weiter untersucht.

- Je höher der Wissenszuwachs war, desto mehr Spaß hat es den Schüler*innen gemacht.
- Je kreativer die Benutzenden waren, desto mehr Spaß hatten sie und desto bessere Ergebnisse haben sie erreicht.

Außerdem soll untersucht werden, wie schwierig die Schüler*innen die Aufgaben empfanden und wie sie die Relevanz des Gelernten bewerten. Wie bedeutsam waren die Inhalte für Folgelektionen und für die Anwendung? Hierzu werden die Antworten auf die qualitativen Fragen „Was hat dir besonders gut gefallen?“ und „Was hat dir nicht so gut gefallen?“ ausgewertet.

6.2.1 Dauer der Bearbeitung

Eine Korrelation zwischen der Dauer der Bearbeitung und dem Spaß wurde mit 97,4 prozentiger Wahrscheinlichkeit festgestellt.[12] Anders als zunächst angenommen, nahm der Spaß mit steigender Dauer nicht ab, sondern zu: Bei einer einseitigen Korrelationsüberprüfung betrug der Korrelationskoeffizient -0,295. Dies könnte einen Hinweis darauf geben, dass die Schüler*innen daran interessiert waren, die Level, trotz der damit verbundenen Herausforderung, zu lösen. Hieraus lässt sich schlussfolgern, dass im Sinne Kellers (1987) durch das Spiel Aufmerksamkeit (Attention) und Zuversicht (Confidence, d.h. die Aufgaben fordernd und machbar sind) erzeugt wurden. Vermutlich sind diese Effekte aber bis zu einer gewissen Dauer begrenzt und eine Frustration stellt sich danach dennoch ein.

Insgesamt haben die Schüler*innen im Durchschnitt 70 Minuten zur Bearbeitung des Kurses „Crocos Bruder“ benötigt, mit einem Minimalwert von 38 Minuten und einem Maximalwert von 130 Minuten. Die Standardabweichung vom Mittelwert beträgt 25 Minuten. Bereinigt um Ausreißer ergibt sich für die Bearbeitung eine durchschnittliche Dauer von 65 Minuten und einer Standardabweichung vom Mittelwert von 16 Minuten. Die Kursdurchführungsdauer betrug daher inklusive der Bearbeitung der Fragebögen im Durchschnitt 77 Minuten. Das entspricht der geplanten Durchführungsdauer: Der Kurs, inklusive der Pretest- und Postest-Fragebögen sowie der Evaluation, sollte innerhalb einer Doppelunterrichtsstunde von 90 Minuten durchführbar sein. Das Ausfüllen der Fragebögen hat durchschnittlich, wie bereits er-

[12] Da fünf Schüler*innen ein leeres Namenskürzel im System angegeben haben, ist hier N = 45.

wähnt, 11 Minuten in Anspruch genommen. Die Schüler*innen konnten in dieser Zeit durchschnittlich 10 von 12 Lektionen mit einer Standardabweichung von 2,043 absolvieren (N = 45) (vgl. Abbildung 8).

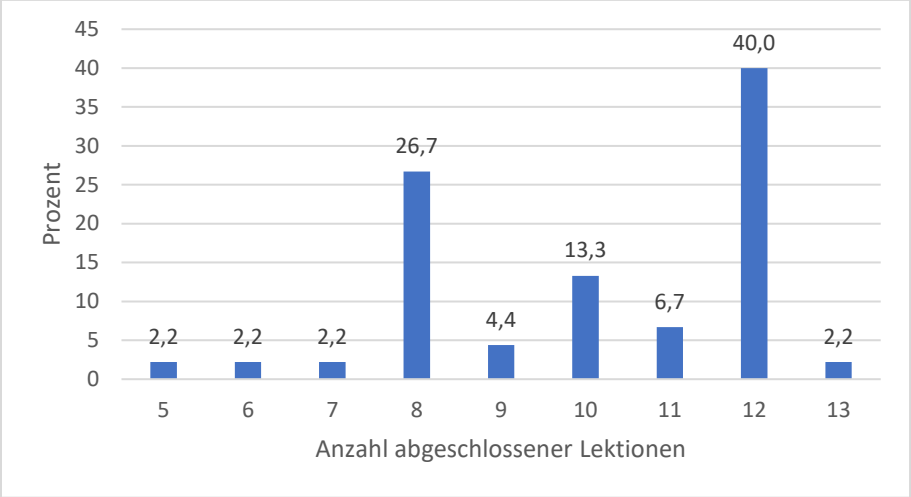

Abbildung 8: Anzahl abgeschlossener Lektionen, N = 45.
*Bemerkung: Es gab keine Schüler*innen, die weniger als fünf Lektionen absolvierten.*

Da die 12. Lektion eine zusammenfassende Aufgabe darstellt, haben die Schüler*innen im Durchschnitt zehn der elf Themenbereiche bearbeitet, lediglich for-Schleifen (Lektion 11) wurden von 60 Prozent der Teilnehmenden nicht gelernt.[13] Schüler*innen, die mehr als zwölf Lektionen abschließen konnten, haben den Kurs „Crocos Bruder“ nach Abschluss auf Englisch erneut gestartet.

6.2.2 Spaß

Nach dem Spaß gefragt, gaben 36 Prozent der Schüler*innen an, dass sie „sehr viel Spaß“ hatten, weitere 54 Prozent hatten „viel Spaß“. Nur 8 Prozent der Benutzenden gaben an, dass sie „wenig Spaß“ hatten. Die Antwortvorgabe „überhaupt kein Spaß“ wurde von keinem Teilnehmenden ausgewählt. Im Durchschnitt wurde der Spaß mit 1,8 bewertet (Skala von 1 – „sehr viel Spaß“ bis 4 – „überhaupt kein Spaß“).[14]

[13] Betrachtet man die Schüler*innen mit acht oder weniger abgeschlossenen Lektionen (15 Schüler*innen insgesamt), so fällt auf, dass acht von ihnen die integrierte Sekundarstufe der Katholischen Schule St. Marien besuchen. Sieben von ihnen besuchten den Kurs am 27.02.2018. Kurse für diesen Schultyp sollten daher womöglich in Zukunft für eine längere Dauer angesetzt oder mit einem angepassten Inhalt angeboten werden.

[14] Gestellt wurde die Frage „Wieviel Spaß hat Dir das Lernen mit Spielen gemacht?“ mit folgenden Antwortmöglichkeiten „sehr viel Spaß“, „viel Spaß“, „wenig Spaß“, „überhaupt kein Spaß“.

6.2.3 Trophäen

Gefragt nach der Motivation durch Trophäen, gaben 13 Schüler*innen „sehr motivierend“ und 23 Schüler*innen „motivierend“ an. Elf Schüler*innen fanden die Trophäen „wenig motivierend“, drei „überhaupt nicht motivierend“ (vgl. Abbildung 9).

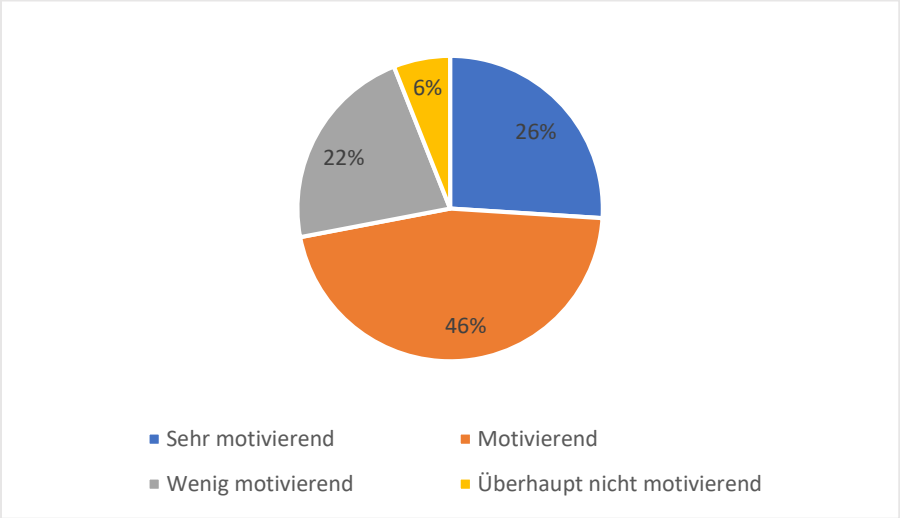

Abbildung 9: Motivation durch Trophäen
Bemerkung: Die Frage hierzu lautete: „Wie motivierend empfandst Du die Trophäen (Bronze, Silber, Gold)?", N = 50

Im Durchschnitt empfanden die Schüler*innen die Trophäen als „motivierend“ (2.3 bei Bewertungsskala 1 bis 4) mit einer Standardabweichung von 1,2. Die lediglich „gute“ Bewertung verwundert insofern, als dass 90 Prozent aller Teilnehmenden mehrmals pro Woche oder häufiger Videospiele spielen und daher angenommen wurde, das Gamification-Elemente, wie sie auch in Videospielen vorkommen, mit einer deutlich positiveren Rückmeldung der Benutzenden verbunden ist. Auffällig ist auch, dass kein Zusammenhang zwischen der Anzahl der Trophäen und der Bewertung des Spaßes festgestellt werden konnte. Möglicherweise war der unmittelbare Nutzen der Trophäen (Die Möglichkeit des Vergleichens der eigenen Leistung mit der Leistung Anderer) nicht klar und deutlich ersichtlich. Dieser Zusammenhang sollte zukünftig weiter untersucht werden. Bemerkenswert ist jedoch, dass Schüler*innen, die durch Trophäen motiviert wurden, auch mehr Spaß empfanden.

Mit dem Spaßempfinden korreliert auch der Wissenszuwachs. Schüler*innen, die selbst einschätzen, viel dazugelernt zu haben, hatten nach eigene Aussagen auch mehr Spaß beim Spiel (Skala 1 – “sehr viel Spaß“ bis 5 „überhaupt kein Spaß“, Signifikanz: 0,07, Korrelation: -0,38).

Dies gibt einen Hinweis darauf, dass sich die von Keller (1987) beschriebene Satisfaction eingestellt hat: Durch sichtbare Erfolge wurde mehr Spaß empfunden.

6.2.4 Kreativität

Um zu ermitteln, wie kreativ die Benutzenden bei der Lösung der Aufgaben waren, wurden Levenshtein-Distanzen zwischen der Abgabe des Benutzenden und der Musterlösung für die jeweilige Aufgabe errechnet und summiert (N = 45). Je höher die Gesamtsumme, desto kreativer war, nach diesem Modell, der Benutzende bei der Lösung der Aufgaben. Ein Zusammenhang zwischen der Kreativität der Benutzenden und dem Spaß konnte zwar nicht festgestellt werden, jedoch hängen Kreativität und die jeweilige Ergebnissumme zusammen: Je kreativer die Schüler*innen bei der Lösung ihrer Aufgaben waren, desto bessere Ergebnisse konnten erzielt werden (Korrelation 0,42 mit Signifikanz von 0,002). Insbesondere bei Aufgaben mit optionalen Zielen kreierten viele Schüler*innen individuelle Lösungen, welche ebenfalls zum bestmöglichen Ergebnis führten.

6.2.5 Schwierigkeitsgrad der Aufgaben

Die Aufgabenschwierigkeit wurde von 80 Prozent der Schüler*innen als „genau richtig“ bewertet. 8 Prozent fanden die Aufgaben „zu leicht“, 12 Prozent „zu schwer“. Die Aufgaben waren im Sinne Kellers (1987) „Confidence“ demnach fordernd, aber machbar (vgl. Abbildung 10).

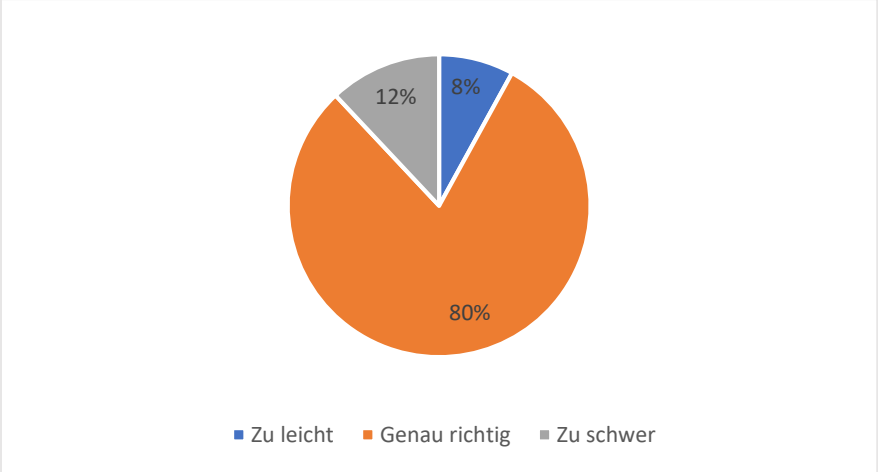

Abbildung 10: Schwierigkeitsgrad der Aufgaben
Bemerkung: Die Frage hierzu lautete: „Wie empfandst Du die Aufgaben?", N = 50

6.2.6 Qualitative Befunde

Für eine weitere Betrachtung der in diesem Kapitel behandelten Aspekte Motivation, Kreativität und Spaß soll nun noch näher auf die Befunde der qualitativen Fragen eingegangen werden. 44 Schüler*innen haben mindestens eine der qualitativen Fragen „Was hat dir besonders gut gefallen?“ und „Was hat dir nicht so gut gefallen?“ beantwortet. Diese Antworten wurden mit folgenden Schlagwörtern kodiert und ausgewertet. In der Klammer ist jeweils die Anzahl der positiven und negativen Antworten dargestellt.

- Spielauftritt: Bewertung des Designs und der Spielmechanik (15 positiv, 2 negativ)
- Gamification-Element: Trophäen (5 positiv, 0 negativ)
- Spielgeschichte (10 positiv, 0 negativ)
- Spielerisches Lernen (17 positiv, 1 negativ)
- Lehrtexte (6 positiv, 6 negativ)
- Relevanz: Aufeinander aufbauender Lerninhalt, Praxisrelevanz und direktes Feedback (4 positiv, 0 negativ)
- Schwierigkeitsgrad der Aufgaben (4 positiv, 3 negativ)
- Plattformauftritt: UI und UX von PLAY & CODE (5 positiv, 3 negativ)
- Nicht aussagekräftige Aussagen: 5

Auch wenn die Kurzantworten der Schüler*innen häufig nur schlagwortartig waren und nicht alle Schüler*innen zu den beiden Fragen Auskunft gaben, so lässt sich doch erkennen, dass vor allem das spielerische Lernen, der Spielauftritt und die Spielgeschichte deutlich positiv bewertet wurden und damit die Motivation, Kreativität und den Spaß als Bedingungen des Lernerfolgs maßgeblich beeinflussen und unterstützen.

Obige Ergebnisse des Kapitels 6.2 lassen den Schluss zu, dass der Großteil der teilnehmenden Schüler*innen für die Aufgaben motiviert werden konnte. Darauf deutet sowohl die Bewertung der Trophäen als auch die gute Bewertung des Spaßes hin. Zudem schienen die Aufgaben meist den richtigen Schwierigkeitsgrad gehabt zu haben. Die Bedeutung des Lernstoffes generell und für folgende Lektionen schien klar.

6.3 Lernerfolg

Nachdem im vorherigen Abschnitt die Faktoren untersucht wurden, die einen Einfluss auf den Lernerfolg haben, soll nun der Lernerfolg der Schüler*innen selbst untersucht werden. Hierzu werden die Ergebnisse des Posttest-Quiz näher betrachtet und die Frage gestellt, ob die Schüler*innen das Erlernte im Posttest-Quiz anwenden konnten.

Der Wissenszuwachs wurde, wie im Kapitel 5 beschrieben, berechnet als die Differenz zwischen dem Pretest-Ergebnis und dem Posttest-Ergebnis. Die Ergebnisse der Tests konnten jeweils Werte zwischen 0 und 8 annehmen. Wurden 8 Punkte erreicht, wurden alle Fragen zu den acht Themenbereichen korrekt beantwortet. Bei 0 Punkten wurde keine Frage korrekt beantwortet. Angestrebt wurde eine möglichst große Differenz als Wissenszuwachs zwischen dem Pretest- und dem Posttest-Ergebnis. In folgendem Diagramm sind die Ergebnisse dargestellt (Abbildung 11).

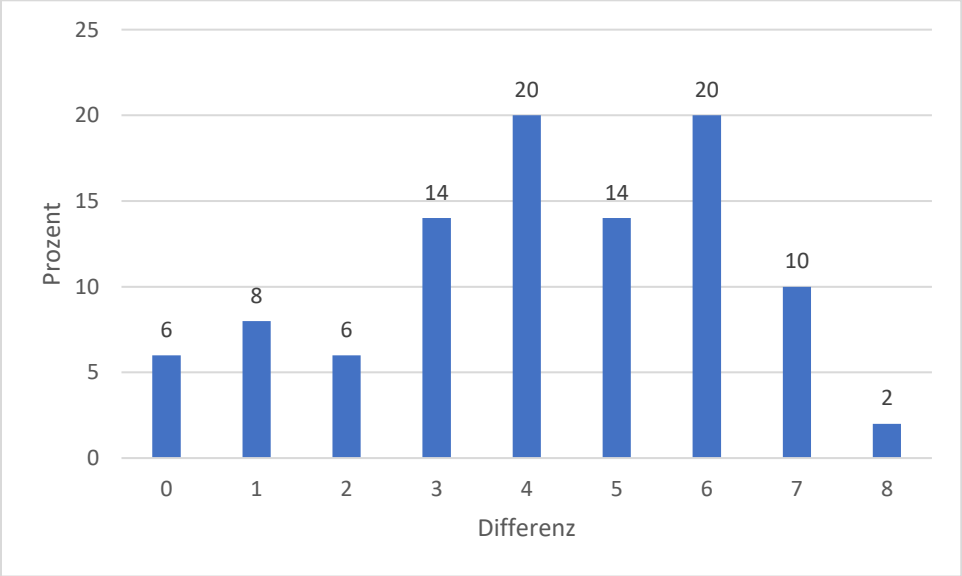

Abbildung 11: Differenz der Pre- und Posttestergebnisse als Wissenszuwachs, N = 50

Im Durchschnitt haben die Schüler*innen im Posttest 4,2 Fragenkategorien mehr als im Pretest korrekt beantworten können und damit ihr Ergebnis des Pretests um 50 Prozent verbessert. Dabei wurden insbesondere Fragen zu Kommentaren, Vergleichsoperatoren, Datentypen und If-Abfragen richtig beantwortet.[15] Die Schüler*innen haben somit eine grundsätzliche

[15] Kommentare: 40 korrekte und 10 falsche Antworten; Vergleichsoperatoren: 40 korrekte und 10 falsche Antworten; Datentypen: 43 korrekte und 7 falsche Antworten; If-Abfragen: 40 korrekte und 10 falsche Antworten;

Vorstellung von Programmierung und Wissen um die Datentypen und konditionalen Verzweigungen in JavaScript durch das Spiel erlangen können.

6.4 Zielgruppe

Ursprünglich wurden als Zielgruppe Schüler*innen ab der 11. Klasse ohne oder mit nur geringer Programmiererfahrungen angesehen. Ab diesem Alter, so wurde angenommen, ist ein gewisses Grundinteresse an Programmierung vorhanden und die Schüler*innen dieser Altersgruppe verfügen über kognitive Fähigkeiten, die Themen adäquat zu verstehen. Es zeigte sich allerdings bei der Kontaktaufnahme mit den Schulen, dass viele Schüler*innen bereits ab der 7. Klasse über Programmiererfahrungen verfügen. Damit stellt sich für die Datenanalyse die Frage, ob Schüler*innen unterschiedlicher Altersgruppen zur Zielgruppe von PLAY & CODE gehören, sofern sie gewisse Vorerfahrungen in Programmierung mitbringen.

Gesucht wurden Teilnehmende, für die PLAY & CODE sehr effektiv war, das heißt, mit einem deutlichen Lernzuwachs verbunden war. Hierzu wurden diejenigen Teilnehmenden gesondert betrachtet, bei denen die Differenz des jeweiligen Pretest- und Posttestergebnisses sechs oder mehr korrekt beantwortete Themenbereiche betrug. Die 15 ausgewählten Schüler*innen zeichnen sich durch nachfolgende Eigenschaften aus:

- Die Schüler*innen sind zwischen 12 und 18 Jahre alt. Ein Drittel davon ist 15 Jahre oder jünger, zwei Drittel 16 Jahre oder älter.[16] Der Mittelwert liegt bei 15,7 Jahre mit einer Standardabweichung von 2,13. Damit liegt die durchschnittliche Altersspanne zwischen (gerundet) 14 und 18 Jahren.
- Sie spielen überwiegend mehrmals die Woche Videospiele, ein Drittel von ihnen spielt jeden Tag.[17]

Variablen: 24 korrekte und 26 falsche Antworten; For-Schleifen: 16 korrekte und 34 falsche Antworten; Funktionen: 21 korrekte und 29 falsche Antworten; Logik-Operatoren: 11 korrekte und 39 falsche Antworten.

[16] Frage: „Wie alt bist du?“, N = 15. 12 Jahre: 2 Nennungen, 13 Jahre: 2 Nennungen, 15 Jahre: 1 Nennung, 16 Jahre: 2 Nennungen, 17 Jahre: 6 Nennungen, 18 Jahre: 2 Nennungen.

[17] Frage: „Wie oft spielst du Videospiele (z.B. auf Konsolen, Smartphones, Computer)?, N = 15. „Mehrmals pro Woche“: 10 Nennungen, „Jeden Tag“: 5 Nennungen.

- Von den 15 betrachteten Teilnehmenden sind zwei weiblich und dreizehn männlich. Somit sind 26 Prozent aller männlichen Teilnehmenden und 18 Prozent aller weiblichen Teilnehmenden in dieser Gruppe (bei N = 50 gesamte Teilnehmerzahl: 11 weiblich, 39 männlich).
- Im Durchschnitt konnten diese Schüler*innen im Posttest 6,4 Fragen mehr beantworten, mit einer Standardabweichung von 0,632 (N = 15).
- Sie konnten im Mittel 11,3 Lektionen abschließen (Standardabweichung 1,033).
- Alle 15 Schüler*innen gaben an, dass sie gerne weitere Programmierinhalte mit Spielen erlernen möchten.
- Alle Schüler*innen besuchen ein Gymnasium.

Für Gymnasialschüler*innen, die 14 bis 18 Jahre alt sind und eine Affinität zu Videospielen aufweisen, scheint PLAY & CODE also am effektivsten, d.h. mit dem größten Lernzuwachs verbunden zu sein. Durch vormals genannte Optimierungen lässt sich diese Zielgruppe womöglich noch erweitern. Zudem könnten unterschiedliche Kurse für unterschiedliche Altersstufen und Schulformen angeboten werden. Eine Gymnasialschullehrerin schlug zudem vor, konditionale Routen in die Kurse einzubauen: Je nachdem wie viele Aufgaben des aktuellen Levels die Schülerin oder der Schüler lösen konnte, kann er zum nächsten Level übergehen oder gleich mehrere Level des gleichen Themenblocks überspringen. Somit könnten nicht nur die Lehrtexte angepasst werden, sondern auch die Abfolge der Lektionen.

6.5 Usability der Oberfläche

Webseiten sollten intuitiv und einfach bedienbar sein, damit der Benutzende sein Ziel möglichst unkompliziert erreichen kann. Es ist daher wichtig zu wissen, wie die Schüler*innen die Usability von PLAY & CODE bewertet haben. Die Bedienbarkeit bewerteten die Schüler*innen in 34 Prozent der Fälle mit „sehr gut“, 48 Prozent gaben „gut“ als Antwort an. 12 Prozent fanden die Bedienung „befriedigend“, 2 Prozent „ausreichend“ und 4 Prozent „mangelhaft“.[18] Im Durchschnitt wurde die Bedienung mit 1,9 als „gut“ bewertet. Begründungen hierfür führ-

[18] Frage: „Wie verständlich war die Bedienung von PLAY & CODE für Dich?“, N = 50. Antworten: „sehr gut“: 17 Nennungen, „gut“: 24 Nennungen, „befriedigend“: 6 Nennungen, „ausreichend“: 1 Nennung, „mangelhaft“: 2 Nennungen, „ungenügend“: keine Nennungen .

ten die Schüler*innen teilweise bei der Frage „Was hat dir nicht so gut gefallen?“ an. Beispielsweise schrieb MoritzE (12. Klasse, St. Marienschule, 19.02.2018): „Das Layout: die Codeeingabe sollte immer offen sein, Keine Wall of Text, Spielbildschirm dafür kleiner...“.

Einige Schüler*innen bemängelten zudem, dass die Oberfläche nicht hinreichend erläutert wurde. Das Design wurde von den Benutzenden zu 62 Prozent als „sehr gut“, zu 32 Prozent als „gut“ und zu 6 Prozent als „befriedigend“ bewertet. Im Mittel wurde das Design mit 1,4 bewertet. Insgesamt wurde PLAY & CODE mit einer 1,7 bewertet (vgl. Abbildung 12).

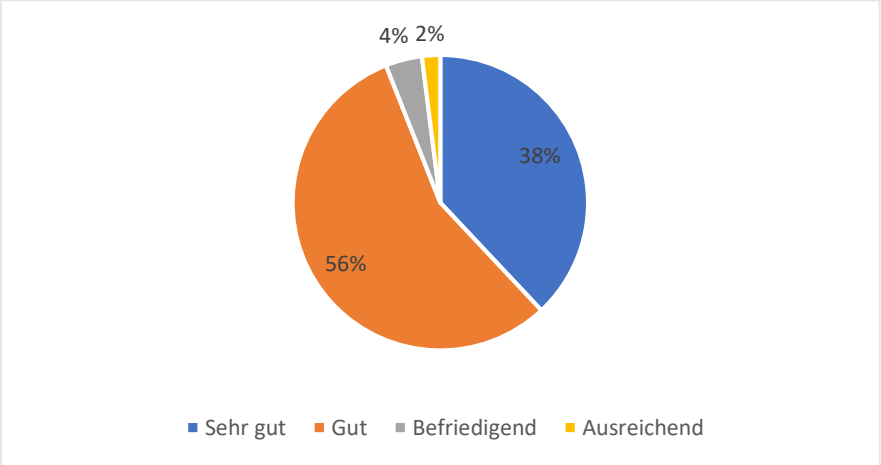

Abbildung 12: Gesamtbewertung des Spiels
Bemerkung: Die Frage hierzu lautete: „Welche Gesamtnote gibst du PLAY & CODE?“, N = 50

Danach gefragt, ob die Schüler*innen gerne weitere Programmierinhalte mit Spielen lernen würden wollen, gaben 92 Prozent (46 Teilnehmende) „Ja“ an, 8 Prozent gaben „Nein“ an.

7 Zusammenfassung und Fazit

Die gesellschaftliche Relevanz der Informatik steigt angesichts der zunehmenden Digitalisierung kontinuierlich. Gleichzeitig nimmt auch der Fachkräftemangel an Programmierer*innen zu. Vielen erscheint Programmierung als zu abstrakt und mühsam. Es müssen deshalb neue Wege gefunden werden, Programmieranfänger*innen zu motivieren und das notwendige Fachwissen praxisnäher zu vermitteln. Folgende Forschungsfrage wurde daher mit dieser Arbeit untersucht: „Können mithilfe eines Serious Games basierten Lernsystems Motivation und Lernerfolg für grundlegende Programmierkenntnisse gesteigert werden?“

Zur Beantwortung dieser Frage wurde die Online-Lernplattform PLAY & CODE entwickelt, die didaktisch auf dem Motivationsmodell ARCS von Keller (1987) sowie den „First Principles of

Instruction“ von Merrill (2002) beruht. Lerninhalte werden mittels eines Serious Games vermittelt und können direkt in diesem Spiel angewandt werden, sodass ein Effekt sofort sichtbar ist. Lehrtexte werden automatisch an den Wissensstand des Benutzenden angepasst. Zusätzliche Motivation soll durch den Einsatz von Trophäen als Gamification-Element erreicht werden.

Um die Wirksamkeit dieser Lernplattform zu überprüfen, wurde eine Evaluation an mehreren Schulen in Berlin durchgeführt. Den Schüler*innen wurden mit der Lernplattform einige Grundlagen der Programmiersprache JavaScript beigebracht. Dabei konnte gezeigt werden, dass die Schüler*innen motiviert waren und Spaß daran hatten, die Programmieraufgaben zu lösen. Mittels eines Pre- und Posttests konnte festgestellt werden, dass die Schüler*innen nach dem Kurs 50 Prozent mehr Fragen zu den gelehrten Themenbereichen korrekt beantworten konnten als davor. Die Schüler*innen haben nach dem Spiel eine genauere Vorstellung von Programmierung und sie wissen um die Datentypen sowie konditionalen Verzweigungen in JavaScript. Die Forschungsfrage kann daher positiv beantwortet werden: Serious Games basierte Lernsysteme wie PLAY & CODE sind ein wirkungsvolles Mittel zur Vermittlung von Programmiergrundlagen. Zudem wurde die Lernplattform zumeist sehr positiv an den Schulen angenommen: Mehrere Schulen sind an den Ergebnissen der Evaluation interessiert. Für die 12. Klasse der Katholischen-Schule St. Marien (Gymnasium) war PLAY & CODE laut der Kurslehrerin eine sehr gute Klausurvorbereitung. Das Friedrich-Ebert-Gymnasium will PLAY & CODE zukünftig für Vertretungsstunden und im einführenden Informatik-Unterricht einsetzen.

Es ist vom Autor der Arbeit geplant, die Lernplattform weiter zu verbessern und auszubauen. Hierfür sollten, wie bereits erwähnt, die Lehrtexte des bestehenden Kurses „Crocos Bruder“ überprüft und verbessert werden. Lektionsspezifische Änderungswünsche von Schüler*innen und Lehrer*innen wurden hierfür ebenfalls erfasst und werden nach Überprüfung einfließen. Zudem konnten einige wiederkehrende Fehler der Schüler*innen beobachtet werden. Beispielsweise wurden Boolean-Werte häufiger in Anführungszeichen geschrieben. Diese Fehler werden in Zukunft in Texten stärker berücksichtigt. Auch eine feinstufigere Adaptivität der Texte sollte in Erwägung gezogen werden. Ebenso werden optionale, konditionale Pfade zum Überspringen von Lektionen eingebaut, falls die Schüler*innen bereits Erfahrung mit dem Themengebiet haben.

Untersucht werden sollte auch, wie die Motivation durch Trophäen weiter gesteigert werden kann. Möglicherweise muss hier das User Interface angepasst werden, sodass der Nutzen der Trophäen offensichtlicher ist. Hier sollten auch die von den Benutzer*innen gewünschten, verbesserten Erläuterungen zur Oberfläche helfen.

Mehrere Schüler*innen hatten zudem nachgefragt, ob, abgesehen von „Crocos Bruder“, noch weitere Kurse eingepflegt werden. Dies ist ebenfalls geplant. Auch ein „Zurückrollen“-Feature wurde gewünscht, mit welchem man den Code auf einen früheren, abgegebenen Zustand zurücksetzen kann. Dies ist mit den bereits vorhandenen Backups der UserCode-Uploads im Ordner „old“ einfach umzusetzen. Zudem hatten sich erfahrenere Schüler*innen einen Einblick in die nicht angezeigte Spielmechanik gewünscht. Diese könnte in zukünftigen Versionen optional dazu geschaltet werden.

Auch bei den Erhebungsinstrumenten gibt es Optimierungsbedarf. Zukünftig sollten in den Fragebögen die Namensfelder erzwungen werden, sodass eine höhere Anzahl an vollständigen Datensätzen erzielt werden kann. Außerdem werden in kommenden Evaluierungen im Pre- und Posttest jeweils die gleiche Anzahl an Fragen pro Themenbereich gestellt. Somit kann der Wissenszuwachs genauer bestimmt werden.

Neben diesen Weiterentwicklungen soll PLAY & CODE auch in Zukunft weiter an Schulen evaluiert werden. Hierfür sind bereits weitere Evaluierungstermine vereinbart. Auch die zusätzlichen Daten vom Friedrich-Ebert-Gymnasium werden sicherlich hilfreich sein.

Neue Bedürfnisse und Techniken fördern neue Ansätze. Der Ansatz der spielerischen Wissensvermittlung kann einen Beitrag zur Motivation und Vermittlung von Inhalten in der Informatik leisten. Durch die Evaluation wurde deutlich, dass dieser Ansatz nicht mit bestehenden konkurrieren muss, sondern diese sich ergänzen können: So kann PLAY & CODE nicht nur alleinstehend, sondern auch erfolgreich als Ergänzung zum Unterricht eingesetzt werden. Auch in andere Themenbereichen wurden Serious Games bereits erfolgreich verwendet. Es bleibt daher spannend, ob der Ansatz des spielerischen Lernens als genereller Lernansatz zukünftig themenübergreifend an Bedeutung gewinnt und sich stärker als bisher etablieren wird.

8 Literaturverzeichnis

Amabile, Teresa M., und Steve J. Kramer. „What really motivates workers.“ *Harvard Business Review*, 2010: 44-45.

Bertelsmann Stiftung. *Gute Aussichten für Arbeitnehmer in Deutschland.* 28. Juli 2010. https://www.bertelsmann-stiftung.de/de/presse/pressemitteilungen/pressemitteilung/pid/gute-aussichten-fuer-arbeitnehmer-in-deutschland/ (Zugriff am 1. Mai 2018).

Boot, Walter R., Daniel P. Blakely, und Daniel J. Simons. „Do action video games improve perception and cognition?“ *Frontiers in Psychology*, 13. September 2011.

Carbonnelle, Pierre. *PYPL PopularitY of Programming Language.* 2018. http://pypl.github.io/PYPL.html (Zugriff am 1. Mai 2018).

Cheng, Meng-Tzu, Jhih-Hao Chen, Sheng-Ju Chu, und Shin-Yen Chen. „The use of serious games in science education: a review of selected empirical research from 2002 to 2013.“ *Journal of Computers in Education*, 19. Juli 2015: 353-375.

Claessen, Koen, und John Hughes. „QuickCheck: a lightweight tool for random testing of Haskell programs.“ *Proceedings of the Fifth ACM SIGPLAN International Conference on Functional Programming.* New York: ACM, 2000. 268-279.

Deterding, Sebastian, Dan Dixon, Rilla Khaled, und Lennart Nacke. „From game design elements to gamefulness: defining gamification.“ *Proceedings of the 15th international academic MindTrek conference: Envisioning future media environments.* 2011. 9-15.

Dignan, Aaron. *Game frame: Using games as a strategy for success.* Free Press, 2011.

Dörner, Ralf, Stefan Göbel, Wolfgang Effelsberg, und Josef Wiemeyer. *Serious Games: Foundations, Concepts and Practice.* Springer, 2016.

Entertainment Software Association. „Essential Facts about the Computer and Video Game industry.“ Washington, DC, April 2015.

Felder, Richard M., Linda K. Silverman, und others. „Learning and teaching styles in engineering education.“ *Engineering Education*, 1988: 674-681.

Fielding, Roy Thomas. *Architectural Styles and the Design of Network-based Software Architectures.* Oakland: University of California, 2000.

Gagné, Robert M., Leslie J. Briggs, und Walter W. Wager. *Principles of Instructional Design.* Bd. 44. Fort Worth: Harcourt Brace College Publishers, 1992.

Gesellschaft für Informatik. *Studienanfängerzahlen in der Informatik: Leichte Steigerung reicht nicht aus.* 25. November 2015. https://gi.de/meldung/studienanfaengerzahlen-in-der-informatik-leichte-steigerung-reicht-nicht-aus/ (Zugriff am 19. April 2018).

Girard, Coralie, Jean Ecalle, und Annie Magnan. „Serious games as new educational tools: how effective are they? A meta-analysis of recent studies.“ *Journal of Computer Assisted Learning*, Juni 2013: 207-219.

Hamari, Juho, Jonna Koivisto, und Harri Sarsa. „Does Gamification Work? - A Literature Review of Empirical Studies on Gamification.“ *47th Hawaii International Conference on System Science.* Waikoloa: IEEE, 2014. 3025-3034.

Heublein, U., et al. „Motive und Ursachen des Studienabbruchs an baden-württembergischen Hochschulen und beruflicher Verbleib der Studienabbrecherinnen und Studienabbrecher.“ Hannover, 2017.

Hubwieser, Peter. *Didaktik der Informatik: Grundlagen, Konzepte, Beispiele.* Springer-Verlag, 2007.

Jeuring, Johan, L. Thomas Binsbergen, Alex Gerdes, und Bastiaan Heeren. „Model solutions and properties for diagnosing student programs in Ask-Elle.“ *Proceedings of the Computer Science Education Research Conference.* Berlin: ACM, 2014. 31-40.

Kalelioğlu, Filiz. „A new way of teaching programming skills to K-12 students: Code.org.“ *Computers in Human Behavior*, 2015: 200-210.

Kazimoglu, Cagin, Mary Kiernan, Liz Bacon, und Lachlan Mackinnon. „A serious game for developing computational thinking and learning introductory computer programming.“ *Procedia-Social and Behavioral Sciences* (Elsevier) 47 (2012): 1991-1999.

Keller, John M. „Development and use of the ARCS model of instructional design.“ *Journal of instructional development* (Springer) 10 (1987): 2-10.

Kirschner, Paul A., John Sweller, und Richard E. Clark. „Why minimal guidance during instruction does not work: An analysis of the failure of constructivist, discovery, problem-based, experiential, and inquiry-based teaching.“ *Educational psychologist* (Taylor & Francis) 41 (2006): 75-86.

Koppel, Oliver. „MINT: Fachkräfte verzweifelt gesucht.“ *Institut der deutschen Wirtschaft Köln*, 2017.

Kozulin, Alex. „Vygotsky’s theory in the classroom: Introduction.“ *European Journal of Psychology of Education* (Springer) 19 (2004): 3-7.

Merrill, M. David. „First principles of instruction.“ *Educational technology research and development* (Springer) 50 (2002): 43-59.

Morgan, Nick. *JavaScript for Kids: A Playful Introduction to Programming.* San Francisco, Kalifornien: No Starch Press, 2014.

Muratet, Mathieu, Patrice Torguet, Fabienne Viallet, und Jean-Pierre Jessel. „Experimental feedback on Prog&Play: a serious game for programming practice.“ *Computer Graphics Forum.* 2011. 61-73.

Muratet, Mathieu, Patrice Torguet, Jean-Pierre Jessel, und Fabienne Viallet. „Towards a serious game to help students learn computer programming.“ *International Journal of Computer Games Technology* (Hindawi Publishing Corp.) 2009 (2009): 3.

O'Grady, Stephen. *The RedMonk Programming Language Rankings: June 2017.* 8. Juni 2017. http://redmonk.com/sogrady/2017/06/08/language-rankings-6-17/ (Zugriff am 1. Mai 2018).

Paramythis, Alexandros, und Susanne Loidl-Reisinger. „Adaptive Learning Environments and e-Learning Standards.“ *Second european conference on e-learning.* Glasgow, 2003. 369-379.

Reinmann, Gabi. „Didaktisches Design - Von der Lerntheorie zur Gestaltungsstrategie.“ 2011: 93-102.

Ritterfeld, Ute, Michael Cody, und Peter Vorderer. *Serious games: Mechanisms and effects.* New York: Routledge, 2009.

Savery, John R. „Overview of problem-based learning: Definitions and distinctions.“ *Interdisciplinary Journal of Problem-Based Learning*, 22. Mai 2006: 5-15.

Schäfer, Andreas, Jan Holz, Thiemo Leonhardt, Ulrik Schroeder, Philipp Brauner, und Martina Ziefle. „From boring to scoring--a collaborative serious game for learning and practicing mathematical logic for computer science education.“ *Computer Science Education* (Taylor & Francis) 23 (2013): 87-111.

Seaborn, Katie, und Deborah I. Fels. „Gamification in theory and action: A survey.“ *International Journal of Human-Computer Studies* (Elsevier) 74 (2015): 14-31.

Statistisches Bundesamt. *Ausstattung privater Haushalte mit Unterhaltungselektronik - Deutschland.* 2017. https://www.destatis.de/DE/ZahlenFakten/GesellschaftStaat/EinkommenKonsumLebensbedingungen/AusstattungGebrauchsguetern/Tabellen/Unterhaltungselektronik_D.html (Zugriff am 1. Mai 2018).

—. *Hochschulen auf einen Blick.* 31. Mai 2016. https://www.destatis.de/DE/Publikationen/Thematisch/BildungForschungKultur/Hochschulen/BroschuereHochschulenBlick0110010167004.pdf?__blob=publicationFile (Zugriff am 20. April 2018).

—. *In 61% der Haushalte mit Kindern gibt es Spielekonsolen.* 28. Januar 2014. https://www.destatis.de/DE/PresseService/Presse/Pressemitteilungen/zdw/2014/PD14_005_p002pdf.pdf;jsessionid=50B5112A7058F3D6ECE9B59778D37DCC.InternetLive2?__blob=publicationFile (Zugriff am 1. Mai 2018).

Stevenson, Daniel E., und Paul J. Wagner. „Developing real-world programming assignments for CS1.“ *Proceedings of the 11th annual SIGCSE conference on Innovation and technology in computer science education.* Bologna: ACM, 2006. 158-162.

TIOBE Programming Community Index. „TIOBE Index for for April 2018.“ April 2018. https://www.tiobe.com/tiobe-index/ (Zugriff am 5. Mai 2018).

Vandewaetere, Mieke, Piet Desmet, und Geraldine Clarebout. „The contribution of learner characteristics in the development of computer-based adaptive learning environments.“ *Computers in Human Behavior* (Elsevier) 27 (2011): 118-130.

Wiarda, Jan-Martin. *Das darf doch nicht wahr sein!* 15. Dezember 2016. http://www.zeit.de/2016/50/studienabbrecher-anstieg-fachhochschulen-studie (Zugriff am 1. Mai 2018).

9 Selbstständigkeitserklärung

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbstständig verfasst und noch nicht für andere Prüfungen eingereicht habe. Sämtliche Quellen einschließlich Internetquellen, die unverändert oder abgewandelt wiedergegeben werden, insbesondere Quellen für Texte, Grafiken, Tabellen und Bilder, sind als solche kenntlich gemacht. Mir ist bekannt, dass bei Verstößen gegen diese Grundsätze ein Verfahren wegen Täuschungsversuchs bzw. Täuschung eingeleitet wird.

Berlin, den .....