Informatische Bildung mithilfe eines MOOC

Diplomarbeit

von

MSc Stefan Janisch

Technische Universität Graz

Fakultät für Informatik und Biomedizinische Technik

Institut Interactive System and Data Science

Priv.-Doz. Dipl.-Ing. Dr.techn. Martin Ebner

Graz, im April 2017

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

Graz, April 2017 .....

Statutory Declaration

I declare that I have authored this thesis independently, that I have not used other than the declared sources / resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

Graz, April 2017 .....

Danksagung

Während des Studiums und der Erstellung dieser Diplomarbeit haben mich viele Menschen unterstützt. In dieser Danksagung möchte ich mich bei ihnen für diese Hilfe bedanken.

Besonderer Dank gilt meiner Familie, die mir immer Rückhalt gegeben hat und ohne die mein Studium nicht möglich gewesen wäre.

In weiterer Folge möchte ich mich noch bei Martin Ebner sowie Wolfgang Slany für ihre fachliche und persönliche Unterstützung bedanken.

Kurzfassung

Das Ziel dieser Arbeit ist es, zu untersuchen, ob es möglich ist, informatische Bildung mithilfe von sogenannten Massiv-Open-Online-Kursen (MOOC) zu vermitteln. Dazu wurde der MOOC „Learning to Code – Programmieren mit Pocket Code“ entwickelt, der vor allem Kindern und Jugendlichen erste Programmiererfahrungen ermöglichen sollte. Dieser Kurs wurde auf der MOOC-Plattform iMooX angeboten und auf zwei verschiedene Arten durchgeführt: Im ersten Setting wurde der Kurs von den TeilnehmerInnen rein online bearbeitet. Im anderen Setting wurde das Inverse-Blended-Learning-Konzept angewendet. Der MOOC wurde mit einer Schulklasse während des Unterrichts durchgeführt. Aufgrund der Anwesenheit des Kursleiters sowie des möglichen Austausches der SchülerInnen untereinander sollte eine Verknüpfung der virtuellen mit der realen Welt der KursteilnehmerInnen geschaffen werden.

Obwohl es im Online-Setting eine für MOOC typische Dropout-Rate von 89% gab, zeigte eine Analyse der Quizbearbeitungen, Videos, Feedbacks und abgegebenen Programme eine starke Auseinandersetzung der aktiven TeilnehmerInnen mit den Kursinhalten. Auch im Schul-Setting ergab die Analyse eine intensive Beschäftigung der SchülerInnen mit dem Thema Programmieren. Die Abschlussprogramme sowie die Auswertung eines Post-Tests verdeutlichten eine Verbesserung der Programmierfähigkeiten. Es zeigte sich, dass MOOCs mit entsprechender Aufbereitung und Durchführung eine Abwechslung und Alternative sein können, um zusätzliche (informatische) Lerninhalte anzubieten und zu vermitteln.

Abstract

The goal of this thesis is to analyze the possibility of using Massive Open Online Courses (MOOC) as a way to impart computer science education. For this reason a new MOOC “Learning to Code – Programmieren mit Pocket Code” was developed. This MOOC should particularly help kids and youths to gain first experiences with programming. The course was offered at the Austrian MOOC platform iMooX and was executed in two different ways. In the first setting the course was done by the participants purely online. In the other setting the so-called “Inverse Blended Learning” concept was applied. The MOOC was executed in classroom with high school students. Due to the presence of the course instructor as well as the possibility of exchange between the students themselves a link between the virtual and the real world should be established.
In the Online setting a high dropout rate of 89% was carried out, which is typical for MOOC courses. However an analysis of the Quiz, videos, feedback and submitted programs revealed a strong involvement of the active participants with the course contents. Also in the school setting the analysis showed an intense engagement of the students with the topic of programming. The evaluation of the final programs as well of the conducted post-test showed an improvement in programming skills. It therefore appears that MOOCs could be used as an alternative and variation to offer and impart additional (computer science) learning content if prepared and implemented appropriately.

.

Inhaltsverzeichnis

1. Einleitung

Diese Arbeit befasst sich mit Massive-Open-Online-Kursen (auch MOOCs genannt) und der Frage, ob sich diese für die Vermittlung von informatischer eignen.

Dazu wurde ein MOOC entwickelt, der in zwei verschiedenen Settings durchgeführt wurde und das Thema „Learning to Code - Programmieren mit Pocket Code“ behandelte. Folgende Forschungsfrage soll beantwortet werden: Inwieweit ist ein MOOC geeignet, informatische Kompetenzen, insbesondere Programmierkompetenzen, zu fördern?

Am Anfang dieser Arbeit soll der Informatik-Unterricht in Österreich durchleuchtet werden, um die Notwendigkeit dieser Arbeit zu verdeutlichen. Die Wichtigkeit von informatischer Bildung und insbesondere die Kompetenz des Programmierens soll erläutert werden sowie Tools wie Pocket Code, die es Kindern erlauben, erste Programmiererfahrungen zu sammeln. In weiterer Folge wird der Begriff und der Aufbau von MOOCs erklärt.

Das 3. Kapitel befasst sich mit der Entwicklung des Online-Kurses. Neben Aufbau und Entwicklung der verschiedenen Materialien soll auch der Inhalt des Kurses vorgestellt werden. Anschließend werden die Durchführung des Kurses und der Evaluationsplan beschrieben. Die Ergebnisse der Auswertung des Kurses in den zwei Settings findet sich im Kapitel Evaluation und werden im darauffolgenden Kapitel diskutiert. Letztendlich soll die Forschungsfrage beantwortet sowie ein Ausblick auf zukünftige Entwicklungen gegeben werden.

2. Hintergrund

„Learning to Code“ – Programmieren mit Pocket Code ist ein Massive-Open-Online-Kurs (MOOC), der auf iMooX.at[1] angeboten wird. Informatische Bildung und hierbei im Speziellen die Fähigkeit des Programmierens gewinnt in der heutigen Gesellschaft immer mehr an Bedeutung. In diesem Kapitel soll deswegen der Informatikunterricht in Österreich durchleuchtet werden sowie die Tatsache, warum Programmieren eine wichtige Rolle in der informatischen Bildung spielt. In weiterer Folge sollen Tools und Programmiersprachen vorgestellt werden, die es Kindern ermöglichen, erste Erfahrungen mit dem Programmieren zu sammeln. Unter anderem gehört hierzu die visuelle Programmiersprache Pocket Code, die auch für den Kurs verwendet wurde. Anschließend soll noch der Begriff des MOOCs erklärt werden, der im Bereich des technologiegestützten Lehrens und Lernens zunehmend eingesetzt wird.

2.1 Informatikunterricht in Österreich

Informatik ist in den Lehrplänen der allgemein bildenden Schulen nicht sehr stark verankert. Lediglich in den Lehrplänen der allgemein bildenden höheren Schulen wird Informatik als Pflichtfach angeführt. Dieses Pflichtfach findet allerdings nur in der 9. Schulstufe statt (AHS-Oberstufe) und beinhaltet zusammengefasst folgende Punkte (Bundesministerum für Bildung, 2017a):

- Informationsmanagement und Lernorganisation
- Strukturierung und Systematisierung von Inhalten und deren Präsentation
- Sicherer Umgang mit Standardsoftware
- Erstellung von Kalkulationsmodellen sowie Interpretation und Präsentation der Ergebnisse
- Technische und theoretische Grundlagen der Informatik

[1] iMooX.at (letzter Abruf am 27.01.2017) ist die erste österreichische MOOC-Plattform

- Rechtliche Grundlagen bezüglich Datenschutz, Urheberrecht und Datensicherheit sowie deren Auswirkung

Programmieren wird im Lehrstoff der 5. Klasse somit nicht explizit erwähnt, eine erste Erwähnung findet sich im Lehrplan des Wahlfach-Unterrichts der 6.-8. Klasse. Hier beinhaltet der Lehrstoff unter anderem den Punkt „Konzepte von Programmiersprachen“ (Bundesministerium für Bildung, 2017b).
In den Lehrplänen der Unterstufe findet man zwar Hinweise auf eine Verwendung von Informationstechnologien im Unterricht; eine konkrete Erwähnung, welche Kompetenzen vermittelt werden sollen, gibt es allerdings nicht (Bundesministerium für Bildung, 2017c).
Es kann somit festgehalten werden, dass es in Österreich momentan keinen verpflichtenden Informatikunterricht bis zur 9. Schulstufe gibt.

2.2 Informatische Bildung und digitale Kompetenzen

Aufgrund der zunehmenden Digitalisierung der Lebenswelt spielt informatische Bildung eine immer wichtigere Rolle in der Ausbildung. Informatische Bildung und digitale Kompetenzen werden zwar in den österreichischen Lehrplänen erwähnt, über welche digitalen Kompetenzen SchülerInnen verfügen sollen, wird dabei aber offen gelassen.
Um die Wichtigkeit von digitalen Kompetenzen in der heutigen Lebenswelt zu unterstreichen, wird Computerkompetenz von der Europäischen Union als eine der acht Schlüsselkompetenzen genannt, die alle Menschen für ihre persönliche Entfaltung, soziale Integration, ihren Bürgersinn und ihre Beschäftigung benötigen (Europäische Union, 2006). Zusätzlich wurde 2013 von der EU das „Digital Competence Framework for Citizens“ auch bekannt als DigComp herausgegeben. Dieser Kompetenzrahmen für digitale Kompetenzen wurde mittlerweile überarbeitet (DigComp 2.0) und besteht aus 5 Bereichen, die 21 Kompetenzen umfassen. Die Bereiche sind: Umgang mit Informationen und Daten, Kommunikation und Zusammenarbeit, Erzeugen digitaler Inhalte, Sicherheit und Problemlösen (Vuorikari, Punie, Carretero, & Van den Brande, 2016).

Auf österreichischer Ebene gibt er hier unter anderem die Initiative digi.komp[2]. In dieser Initiative werden die digitalen Kompetenzen, die inhaltlich auch auf dem DigComp-2.0-Modell basieren, in drei Kompetenzstufen unterteilt (Bauer & Waba, 2016):

- digi.komp 4 (Grundstufe, Volksschule)
- digi.kom 8 (Sekundarstufe)
- digi.komp 12/13 (Sekundarstufe 2, AHS/BMHS)

Für jede der drei Stufen gibt es entsprechende Kompetenzraster und Unterrichts-/Begleitmaterialien für LehrerInnen.

Dass die Stärkung von MINT-Fächern, insbesondere der Informatik, ein wichtiges und auch politisches Anliegen ist, zeigt auch das erst kürzlich ins Leben gerufene Regierungsprojekt die Digital Roadmap[3] sowie der Regierungsplan zu Bildung 4.0[4].

2.3 Programmieren als Kompetenz

Programmieren wird als 4. Punkt im Bereich „Erzeugen digitaler Inhalte“ im DigComp-Framework der EU genannt. Es wird beschrieben als das Planen und Entwickeln einer Folge von verständlichen Anweisungen für ein Computersystem, um ein gegebenes Problem zu lösen oder eine bestimmte Aufgabe auszuführen (Vuorikari et al, 2016).

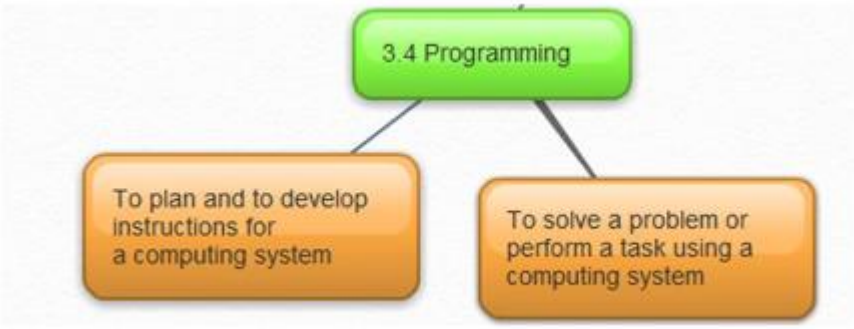


Abbildung 1: Kompetenz Programmieren (Vuorikari et al, 2016).

[2] www.digikomp.at (letzter Abruf am 14.02.2017)

[3] https://www.digitalroadmap.gv.at/ (letzter Abruf am 06.03.2017)

[4] https://www.bmb.gv.at/schulen/schule40/index.html (letzter Abruf am 06.03.2017)

Diese Problemlösungsfähigkeit, die beim Programmieren auftaucht, wird auch häufig als sogenanntes „Computational Thinking“ bezeichnet. Computational Thinking inkludiert dabei nicht nur die Fähigkeit, Probleme zu lösen, sondern auch den ganzen Prozess, der dafür notwendig ist. Hier wird zum Beispiel das Automatisieren von Lösungen durch algorithmisches Denken, Organisieren und Analysieren von Daten oder auch das Verallgemeinern und Anwenden dieser Problemlösungsprozesse auf verschiedene andere Problemstellungen genannt (Futschek, 2016). Laut Wing (2006) ist Computational Thinking eine fundamentale Fähigkeit nicht nur für InformatikerInnen. Neben Lesen, Schreiben und Arithmetik sollte Computational Thinking zu den analytischen Fähigkeiten eines Kindes dazugehören.

Durch Programmieren können Kinder auch ihre eigenen kreativen Ideen umsetzen; sie lernen, was es heißt, sich mit dem Computer auszudrücken, und werden so von passiven zu aktiven Nutzern von Technologien.

2.4 Programmieren für Kinder

Als in den späten 1970er und 1980er Jahren die ersten Computer eingeführt wurden, herrschte zunächst eine große Begeisterung, Kindern Programmieren beizubringen. Viele Schulen, besonders in den USA, brachten ihren Schülern bei, einfache Programme mit der Programmiersprache Logo oder Basic zu schreiben. Ein Vorreiter auf diesem Gebiet war Seymour Pappet, der in seinem 1980 erschienen Buch *Mindstorms* die Programmiersprache Logo als Eckpfeiler präsentierte, um Ansätze des Lernens zu überdenken (Pappert, 1980). Obwohl einige SchülerInnen und LehrerInnen von diesen neuen Möglichkeiten sehr angetan waren, wechselten die meisten Schulen zu einem anderen Einsatz von Computern. Seit dieser Zeit sind Computer im Leben von Kindern allgegenwärtig, wobei die wenigsten Programmieren lernen. Heutzutage wird Computerprogrammieren von vielen Leuten als eng begrenzte, komplexe technische Aktivität gesehen, die nur einem kleinen Teil der Bevölkerung zumutbar ist (Resnick, et al., 2009).

Es stellt sich die Frage, was mit dem anfänglichen Enthusiasmus für Programmieren passiert ist. Warum konnten sich Initiativen und Programmiersprachen

wie zum Beispiel Logo nicht durchsetzen? Dafür werden einige Faktoren genannt (Resnick, et al., 2009):

- Frühe Programmiersprachen waren schwierig zu verwenden und viele Kinder kamen einfach nicht mit der Syntax des Programmierens zurecht.
- Programmieren war oft verknüpft mit Aktivitäten, wie zum Beispiel der Erzeugung von Primzahlen-Listen oder Herstellung einfacher Linienzeichnungen, die nicht an die Lebenswelt und Interessen von Kindern angepasst waren.
- Programmieren wurde oft in dem Kontext eingeführt, dass keiner weiterhelfen konnte, wenn etwas nicht funktionierte, beziehungsweise dass keiner ermutigt wurde, sich weiter damit zu beschäftigen, wenn es funktionierte.

Laut Papert (1980) sollten Programmiersprachen einen „low floor“ (leicht zu beginnen) und „high ceiling“ (Möglichkeiten, um immer komplexere Projekte im Laufe der Zeit zu machen) haben. Zusätzlich brauchen diese Sprachen auch noch „wide walls“ (viele unterschiedliche Projekte sollen unterstützt werden, sodass viele Personen mit unterschiedlichen Interessen und Lernstilen angesprochen werden können). Diese drei Bedingungen zu erfüllen, war nicht einfach (Guzdial, 2004).

2003 wurde schließlich vom MIT-Media-Lab das Scratch-Projekt ins Leben gerufen. Daraus entstand 2007 die Programmiersprache Scratch, die sich an Personen ohne jegliche Programmiererfahrung richtete und in diesem Feld mittlerweile weltweit eingesetzt wird (Maloney, Resnick, Rusk, Silverman, & Eastmond, 2010). Aus diesem Vorreiter entwickelten sich schließlich zahlreiche andere Initiativen und Programmiersprachen wie zum Beispiel Pocket Code.

2.4.1 Scratch

Scratch ist eine visuelle Block-basierte Programmiersprache, die entwickelt wurde, um NutzerInnen einen leichten Einstieg in die Programmierung zu ermöglichen. Scratch hat als Schwerpunkt die Manipulation von Medienelementen (Fotos, Musik, Grafiken, Sounds, etc.) und unterstützt Programmier-Aktivitäten, die sich an den Interessen von Kindern und Jugendlichen orientie-

ren, wie der Erstellung von animierten Geschichten, Spielen, Musik-Videos oder interaktiven Präsentationen (Maloney, Peppler, Kafai, Resnick, & Rusk, 2008).
Die Grammatik der Programmiersprache basiert auf einer Sammlung von grafischen „Programmier-Blöcken“, die Kinder zusammenstecken, um Programme zu erzeugen (Abbildung 2). Gleich wie bei Legosteinen gibt es auch bei diesen Blöcken Verbindungen, die anzeigen, wie die Blöcke zusammengesteckt werden können. Somit können Kinder anfangen mit diesen Blöcken zu experimentieren, indem sie verschiedene Blöcke in verschiedenen Sequenzen und Kombinationen zusammenstecken (Skripte). Es gibt keine undurchsichtige Syntax und das Ergebnis ist sofort sichtbar. Dieses „Sofort-Sichtbare“ ist ein wesentlicher Bestandteil von Scratch. Das Programm muss nicht kompiliert werden und NutzerInnen können jederzeit auf ein Programmfragment klicken, um zu sehen, was passiert. Genau genommen können sogar Parameter verändert werden oder Blöcke zu einem Skript hinzugefügt werden, während dieses ausgeführt wird. Somit können kleine Codesegmente erstellt und getestet werden, die später zu einem großen zusammengeführt werden können.
Zusätzlich sind die Blöcke so entworfen, dass sie nur mit den Blöcken zusammenpassen, die syntaktisch Sinn machen. So suggerieren zum Beispiel Schleifen-Blöcke („forever“, „repeat“) durch ihre Form („c-shaped“), dass andere Blöcke zwischen ihnen platziert werden sollen (Resnick, et al., 2009).

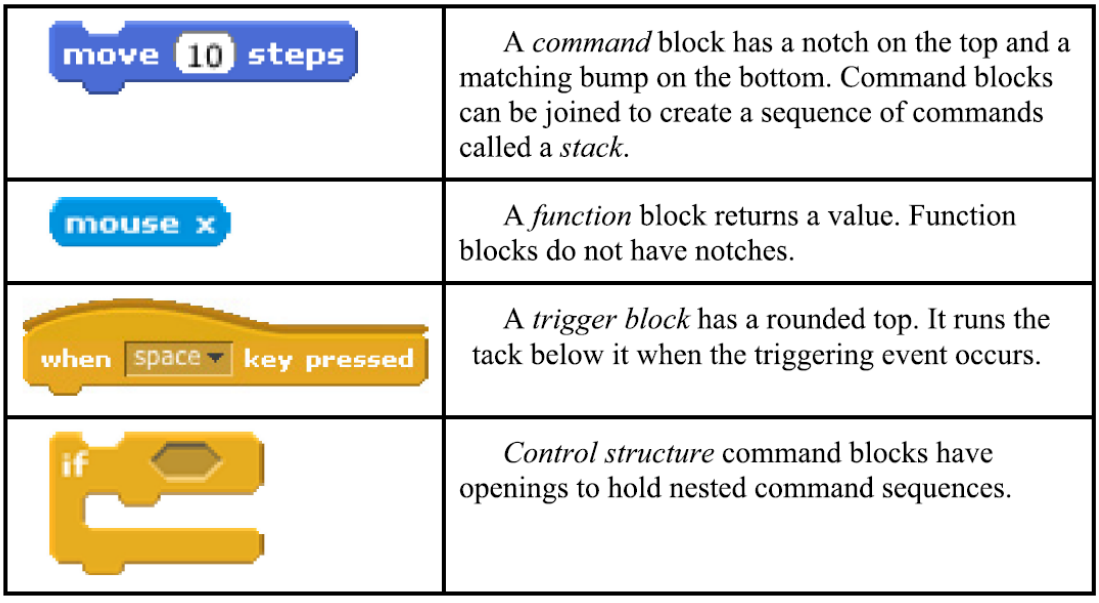


Abbildung 2: Verschiedene Blocktypen in Scratch (Resnick, et al., 2009)

Jedes Scratch-Projekt besteht grundsätzlich aus einer fixen Bühne (Hintergrund) und einer bestimmten Anzahl an verschiebbaren sogenannten „Sprites“ (Objekte). Jedes dieser Objekte hat einen eigenen Satz von Bildern, Sounds, Variablen und Skripten (Abbildung 3). Diese Organisation ermöglicht einen einfachen Export und Austausch von Sprites, da jeder Sprite unabhängig ist. Durch diesen Austausch wird die Wiederverwendung von Code und Kollaboration gefördert. So sind seit dem Start von Scratch schon mehr als eine Million Projekte von mehr als 120000 Nutzern auf deren Website hochgeladen worden. Jeden Tag werden 1500 neue Projekte hochgeladen, was im Durchschnitt ein neues Projekt pro Minute bedeutet (Maloney et al, 2010).

Abbildung 3: Die Scratch Benutzeroberfläche (Maloney et al, 2010)

2.4.2 Pocket Code

Nicht nur Erwachsene, sondern auch Kinder und Jugendliche besitzen heutzutage ihr eigenes Smartphone. So haben schon Studien von 2014 festgestellt, dass der Großteil der 10- bis 18-Jährigen im Besitz eines eigenen Smartphones ist (Feierabend, Plankenhorn, & Rathgeb, 2015) (Nagler, Ebner, & Schön, 2016).

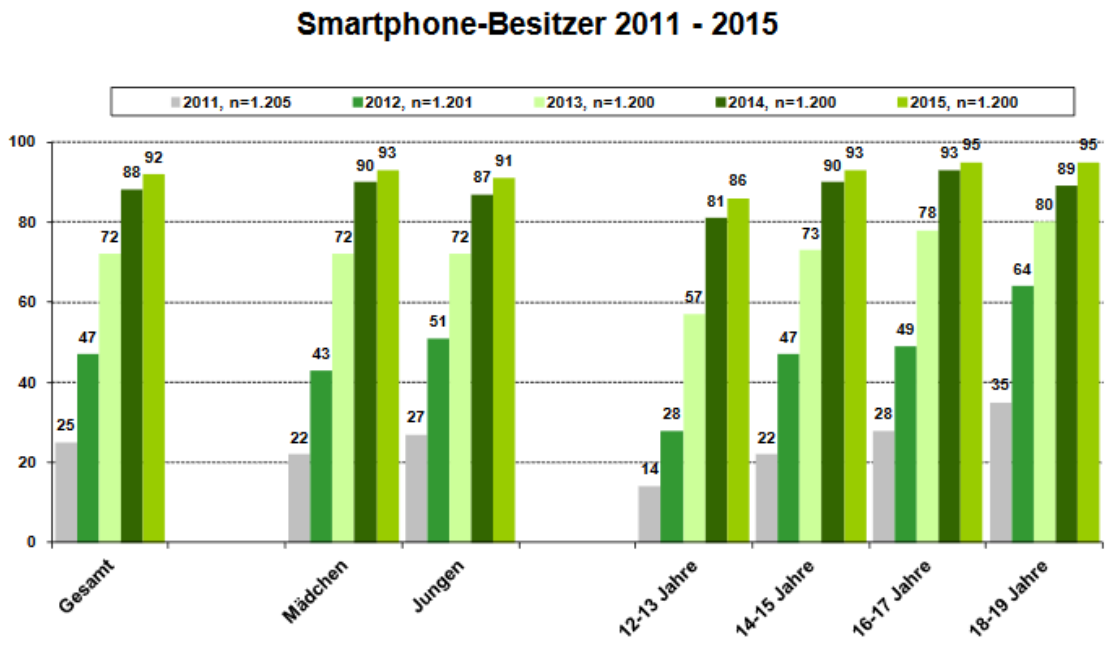


Abbildung 4: Prozent der Smartphone Besitzer in Deutschland nach Altersklasse eingeteilt (Feierabend et al, 2015)

Selbst unter vielen Kindern im Alter von 6 bis 7 Jahren wird der Griff zum Smartphone eine alltägliche Gewohnheit (Grimus & Ebner, 2014).

Unter diesem Gesichtspunkt wurde an der TU Graz eine Gratis-App mit Namen „Pocket Code“ entwickelt, die es Kindern und Jugendlichen ermöglichen soll, erste Programmiererfahrungen direkt am Smartphone zu sammeln. Pocket Code ist gratis im Google-Play-Store erhältlich und erlaubt es, Spiele, Animationen oder andere Apps mit dem eigenen mobilen Gerät zu erstellen. Ähnlich wie Scratch ist auch Pocket Code eine visuelle objektbasierte Programmiersprache, in der der Focus auf die Semantik des Programmierens gelegt wird. Durch den Einsatz von legoartigen Blöcken sollen besonders die syntaktischen Probleme von Programmiersprachen eliminiert werden (Janisch, Slany, & Ebner, 2016).

Die Funktionsweise von Pocket Code soll an folgendem Beispiel demonstriert werden (Abbildung 5):

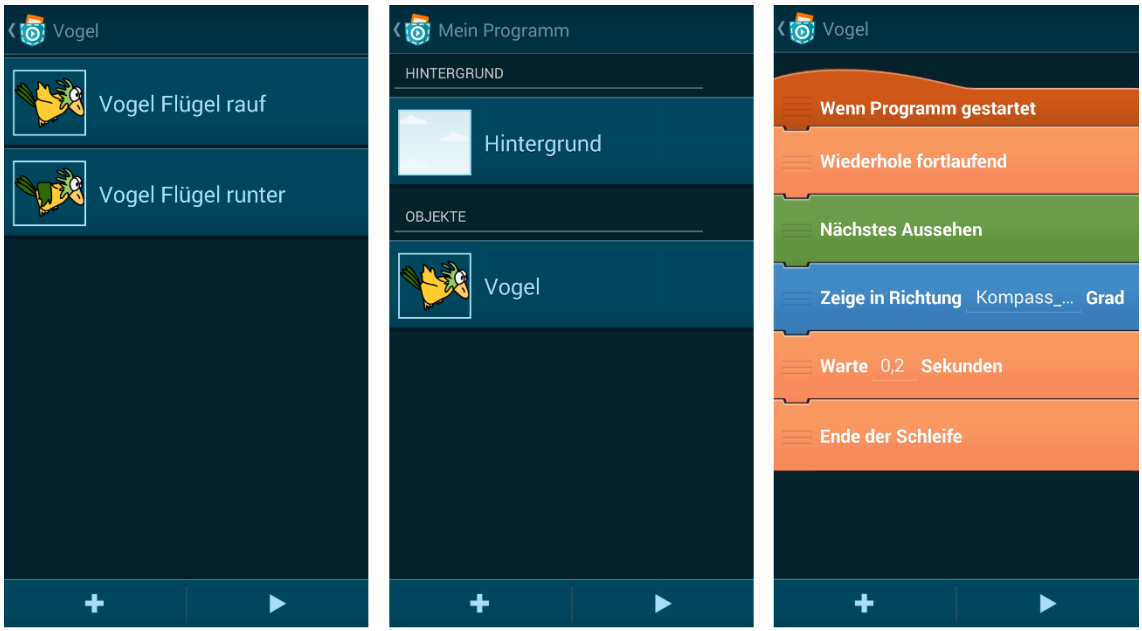


Abbildung 5: Pocket Code Beispielprogramm

Jedes Programm besteht aus einer beliebigen Anzahl an Objekten und einem Hintergrund (der ein spezielles Objekt ist). Jedes Objekt hat a) *Skripten*, die das Verhalten des Objekts steuern b) *Aussehen*, die bestimmen, wie das Objekt dargestellt wird c) *Klänge*, um Musik in das Programm zu integrieren. Das Verhalten des Objekts sowie das Aussehen und die Klänge werden mithilfe von Skripten gesteuert. In dem gezeigten Beispielprogramm (Abbildung 5) ist das Ziel, einen Vogel zu programmieren, der mit den Flügeln schlägt sowie immer nach Norden zeigt, egal wie das Smartphone gehalten wird. Dieses Programm mit Namen „Mein Programm“ besteht aus zwei Elementen, einem blauen Himmel-Hintergrund-Objekt und einem Vogel-Objekt (Abbildung 5 (links)). Das Objekt Vogel hat 2 verschiedene Aussehen (Abbildung 5 (Mitte)), die verwendet werden, um das Flügelschlagen des Vogels zu animieren. Der Skriptbereich des Objekts Vogel beinhaltet ein einzelnes Skript, das den Flügelschlag des Vogels sowie die Ausrichtung nach Norden steuert (Abbildung 5 (rechts)). Das Skript besteht aus verschiedenfarbigen Blöcken, die anzeigen, aus welcher Kategorie die Blöcke kommen (Ereignisse, Steuerung, Bewegung, Klang, Aussehen, Malstift, Daten). Der erste Block „Wenn Programm gestartet“ startet die Ausführung des Skripts immer dann, wenn das Programm vom dem oder der Userin gestartet wird. Der „Wiederhole Fortlaufend“-Baustein mit dazugehörigem „Ende der Schleife“-Block repräsentiert eine endlose Schleife; jeder Block,

der sich dazwischen befindet, wird so lang ausgeführt, bis das Programm beendet wird. Der „Nächstes Aussehen“-Block wechselt das Aussehen des Objekts von „Flügel rauf“ auf „Flügel runter“ und der „Zeige in Richtung *Kompass_Richtung* Grad“- Baustein aktualisiert die Richtung, in die das Objekt zeigt. Um die Kompass_Richtung auf das Objekt anzuwenden, greift Pocket Code auf die Sensoren des Smartphones zu. Da ein Smartphone verschiedenste Sensoren hat (Beschleunigung, Neigung, Lautstärke, Touchscreen, Kamera, etc.) kann ein Programm sehr interaktiv gestaltet werden, was bei Desktop-Anwendungen nicht in diesem Ausmaß gegeben ist. Der letzte Block in der Endlos-Schleife hat die Funktion, die Animationsgeschwindigkeit des Flügelschlages zu verlangsamen. Aus diesem Grund ist dieser Block ein „Warte 0,2 Sekunden“-Block, der eine 0,2s-Verzögerung in der Schleife auslöst (Spieler, et al., 2016). Anhand dieses Beispiels ist ersichtlich, wie verschiedene Blöcke eingesetzt werden, um Skripte zu erzeugen, mit denen man letztendlich ein Programm kreieren kann. Gleich wie Scratch können die Skripte eines Objektes parallel laufen und Objekte können durch verschiedene Blöcke miteinander kommunizieren (Petri et al, 2015).

Ein Programm kann jederzeit getestet werden und fertige Programme können direkt als App am Smartphone gespeichert sowie auf die Pocket-Code-Community-Seite[5] geladen werden. Auf dieser Plattform ist es möglich, sich die Programme von anderen anzuschauen, zu kommentieren, herunterzuladen sowie in weiterer Folge auch zu verändern. Zum derzeitigen Zeitpunkt (Jänner 2017) gibt es auf dieser Community-Seite mehr als 20000 Projekte von Nutzern aus der ganzen Welt. Diese Projekte umfassen unter anderem animierte Geschichten, interaktive Präsentationen, Musikvideos, Spiele und viele andere Arten von Apps.

2.5 MOOCs

Ein MOOC (massive open online course) ist ein frei zugänglicher Online-Kurs (ohne Zugangsvoraussetzungen), an dem unbegrenzt viele Personen teilnehmen können (Kaplan & Haenlein, 2016). Zusätzlich zu Kurs-Materialien wie Vi-

[5] https://share.catrob.at/pocketcode/ (letzter Abruf am 24.02.2017)

deos, Vorträgen oder Aufgaben gibt es häufig Foren oder Chats, die den Austausch zwischen den TeilnehmerInnen ermöglichen. Der Begriff MOOC setzt sich aus den Wörtern **M**assive **O**pen **O**nline **C**ourse zusammen.

Massive beschreibt dabei die Teilnehmerzahl von solchen Kursen; ein Kurs wird als „massive“ bezeichnet, wenn die sogenannte Dunbar-Zahl überschritten wird. Open bedeutet, dass es keine Zugangsvoraussetzungen sowie einen freien Zugriff auf alle Kursmaterialien gibt. Online bezieht sich auf die Abwicklung und Kommunikation eines solchen Kurses; diese erfolgt über eine zentrale Anlaufstelle wie eine Website (Wedekind, 2013). Die Bezeichnung Course ergibt sich aus der Organisation. Den TeilnehmerInnen wird eine eigenständige Bearbeitung des Kursthemas ermöglicht, meistens gibt es aber einen definierten Start- und Endtermin. Die Inhalte werden von einem bzw. mehreren Lehrenden vorgegeben und zu bestimmten Zeiten werden Inhalte freigeschaltet (Kaplan & Haenlein, 2016). Die Einteilung von MOOCs erfolgt oft in 2 Typen: cMOOCs und xMOOCs.

Gerade für Schulen könnten MOOCs eine zukünftige Alternative sein, da sie kostenlose, zusätzliche Lehrinhalte anbieten, für deren Produktion spezialisierte Lehrkräfte nötig wären, die an der Schule fehlen. Die Kombination von Präsenzlehre mit MOOCs unter der Berücksichtigung von kollaborativen Elementen lässt neue didaktische Möglichkeiten zu (Dreisibner, Ebner, & Kopp, 2014). In den USA bekommen SchülerInnen die Absolvierung von MOOCs teilweise schon zu den Credits angerechnet (Jackson, 2013). Auch LehrerInnen könnten MOOCs für die Aus- und Weiterbildung nutzen.

3. Entwicklung des Online-Kurses

In diesem Kapitel sollen die Überlegungen wiedergegeben werden, die maßgeblich für die Entwicklung des Kurses verantwortlich waren. Einerseits wird auf generelle Überlegungen und Aufbau eingegangen, anderseits werden kurz die Inhalte der einzelnen Wochen vorgestellt.

3.1 Struktur von iMooX

Da der Kurs auf der MOOC-Plattform iMooX angeboten wurde, mussten die Inhalte und der Aufbau an die entsprechende Struktur angepasst werden. Jeder Kurs auf iMooX besteht aus einer gewissen Anzahl an Kapiteln, wobei jedes Kapitel wochenweise freigeschaltet wird. Ein Kurs bestehend aus 6 Kapiteln, hat somit eine Dauer von 6 Wochen. Herzstück von jedem Kapitel sind eingebettete Videos, die das Thema des Kapitels behandeln. Neben den Videos ist es auch möglich, Zusatzmaterialien in Form von Bildern, Dokumenten oder Links auf externe Materialien einzubauen. Am Ende jedes Kapitels gibt es ein Quiz, mit dem die TeilnehmerInnen ihr Wissen überprüfen können. Jedes Quiz besteht aus einer gewissen Anzahl an Fragen, die entweder vom Format Multiple Choice oder Single Choice sein können. Ein Quiz kann öfter absolviert werden, wobei der beste Versuch zählt. Um den Kurs abzuschließen, anders ausgedrückt, um die Teilnahmebestätigung zu erhalten, muss jedes Quiz positiv absolviert werden (= 75% der Fragen richtig beantwortet). Ein wichtiger Bestandteil jedes Kurses ist ein Forum. Hier können verschiedene Threads erstellt werden, in denen unterschiedliche Themen behandelt werden können. Es dient einerseits zum Austausch der Kursleitung mit den TeilnehmerInnen (z.B.: bei Fragen), anderseits können sich die TeilnehmerInnen untereinander austauschen.

3.2 Aufbau des Kurses

Als Inhalt des MOOC wurde das Thema Programmieren gewählt. Wie aus dem vorherigen Kapitel ersichtlich ist, ist Programmieren ein wesentlicher Bestandteil informatischer Bildung. Da Programmieren erst konkret im Lehrplan des

Wahlfaches der 6.-8. Klasse AHS-Oberstufe erwähnt wird, stellte sich die Frage nach einer früheren Bearbeitung dieses wichtigen Themas. „Learning to Code“ – Programmieren mit Pocket Code“ wurde deswegen speziell für Kinder und Jugendliche im Alter von 10 – 14 Jahren entwickelt.

Pocket Code ist eine visuelle Programmiersprache, mit der auf mobilen Endgeräten programmiert werden kann. Durch ihre grafische Benutzeroberfläche eignet sie sich sehr gut für Kinder und Jugendliche, um erste Programmiererfahrungen zu sammeln. Da die meisten Kinder und Jugendlichen schon ihr eigenes Smartphone haben (siehe Kapitel 2), kann hier direkt am privaten Handy programmiert werden.

Eine große Herausforderung dieses Kurses war es, die Inhalte und den Aufbau an die entsprechende Altersgruppe anzupassen. Ein häufiges Problem von MOOCs ist die hohe „Dropout“-Rate. Viele TeilnehmerInnen, die einen Kurs beginnen, beenden diesen nicht. Als Ursache dafür werden oft folgende Gründe genannt (Khalil & Ebner, 2014):

- Zeitmangel
- Motivation der Lernenden
- Fehlendes Hintergrundwissen
- Gefühl der Isolation und das Fehlen von Interaktivität

Zusätzlich muss beachtet werden, dass Kinder und SchülerInnen andere Bedürfnisse und Anforderungen haben als Universitätsstudenten oder Erwachsene in Weiterbildungskursen, die meistens im Fokus von MOOCs stehen (Boxser & Agarwal, 2014). In Großbritannien gibt es einige MOOCs, die für 11- bis 19-Jährige zur Verfügung stehen sowie auf der bekannten MOOC-Plattform edX[6]. Ebner & Khalil (2015) haben einen 10-wöchigen MOOC für Schulkinder untersucht, der auf iMooX angeboten wurde. Daraus leiteten sie unter anderem folgende Empfehlungen für die Implementierung eines STEM-MOOCs[7] für Schulkinder ab:

- Der MOOC soll eine didaktische Vorgehensweise im Klassenzimmer haben, um die Motivation der SchülerInnen zu fördern.

[6] https://www.edx.org/high-school

[7] MOOC im Bereich Science, Technology, Engineering und Mathematics

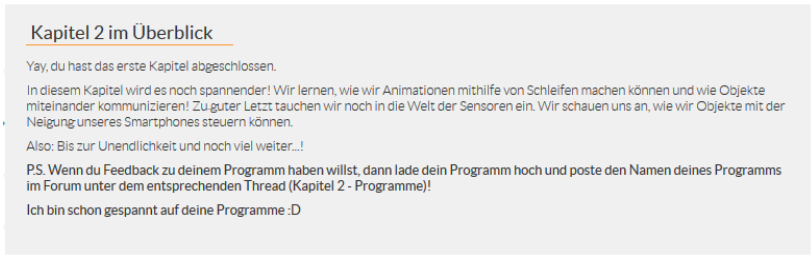
- Die Ausdehnung der Rolle des Lehrers im Klassenzimmer spielt eine wichtige Rolle.
- Kürzere MOOCs für eine höhere Abschlussrate.

Aus diesen Überlegungen wurde schließlich der Aufbau des Kurses entwickelt. Als Dauer wurde 5 Wochen festgelegt. In diesen 5 Wochen sollen die TeilnehmerInnen von ersten Programmiererfahrungen bis hin zur Entwicklung eines eigenen Programms mithilfe von Pocket Code geführt werden. Dabei wurde das Augenmerk auf die Vermittlung von Programmierkonzepten, wie Objekte, Schleifen, Parallelismus, Variable und Anweisungen, gelegt. Da man Programmieren am besten durch eigenes Üben lernt, wurden zusätzliche Aufgaben für jede Woche entwickelt, die auf dem zuvor behandelten Stoff aufbauten. Diese Aufgaben sollten den TeilnehmerInnen die Möglichkeit geben, ihr Wissen kreativ einzusetzen und gelernte Konzepte einzusetzen. Der Lernstoff sowie die Aufgaben wurden mit Videos vermittelt, wobei es bei den Aufgaben jeweils immer eine zusätzliche schriftliche Beschreibung der Aufgabe sowie einen Tipp und Lösungsvorschlag gab. Als Begleitmaterialen in schriftlicher Form gab es sogenannte „Pocket-Karten“; welche einen Einblick in die Funktionsweise der verschiedenen Blöcke gaben.

Am Ende jeder Woche gab es auch ein Quiz, das aus 5 Fragen bestand. Die Fragen richteten sich dabei an Inhalte, die in dem jeweiligen Kapitel behandelt wurden. Jedes Quiz konnte 5-mal absolviert werden, wobei nur der beste Versuch zählte. In jeder Frage gab es nur eine richtige Lösung, und um ein Quiz positiv abzuschließen, mussten 75% der Fragen richtig beantwortet werden. Nach positivem Abschluss eines Quiz wurde das Kapitel als abgeschlossen gewertet und ein Badge wurde ausgehändigt (Abbildung 6).

Abbildung 6: Badge der dritten Woche

Für Fragen an die Kursleitung und den Austausch zwischen den TeilnehmerInnen wurde das Forum verwendet. Hier wurden Threads zu allgemeinen Themen erstellt (technische Probleme, Spielewiese, Sonstiges) sowie Threads für die einzelnen Wochen. In jeder Woche wurde den TeilnehmerInnen angeboten die Namen ihrer erstellten Programme im Forum zu posten, um ein Feedback des Kursleiters zu bekommen. Am Anfang jedes Kapitels gab es einen kurzen Begrüßungstext, der das Kapitel kurz vorstellte. Für den positiven Abschluss des gesamten Kurses wurden ein spezieller Badge sowie eine Teilnahmebestätigung ausgegeben. Abbildung 7 verdeutlicht den typischen Aufbau eines Kapitels.


Kapitel 2 im Überblick

Yay, du hast das erste Kapitel abgeschlossen.

In diesem Kapitel wird es noch spannender! Wir lernen, wie wir Animationen mithilfe von Schleifen machen können und wie Objekte miteinander kommunizieren! Zu guter Letzt tauchen wir noch in die Welt der Sensoren ein. Wir schauen uns an, wie wir Objekte mit der Neigung unseres Smartphones steuern können.

Also: Bis zur Unendlichkeit und noch viel weiter...!

P.S. Wenn du Feedback zu deinem Programm haben willst, dann lade dein Programm hoch und poste den Namen deines Programms im Forum unter dem entsprechenden Thread (Kapitel 2 - Programme)!

Ich bin schon gespannt auf deine Programme :D

Begrüßungstext

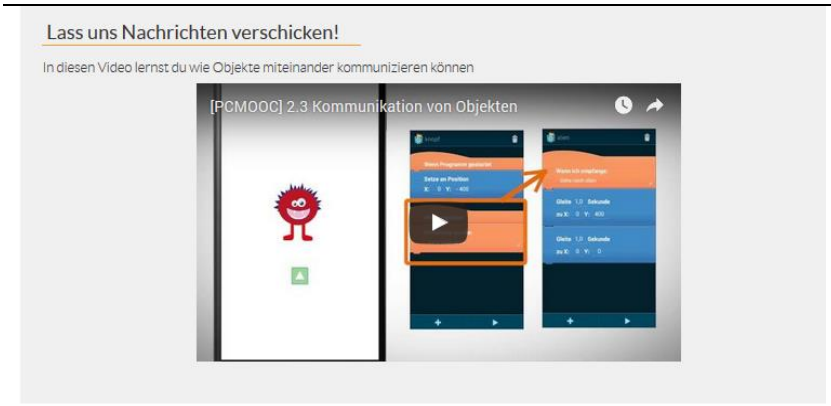
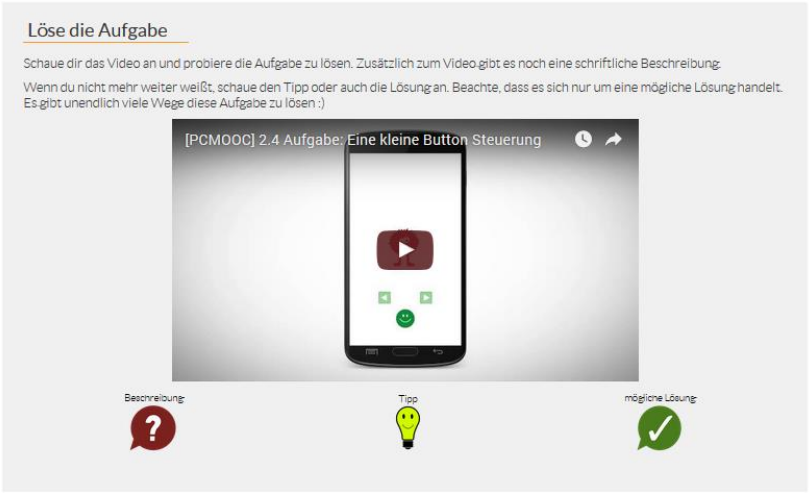
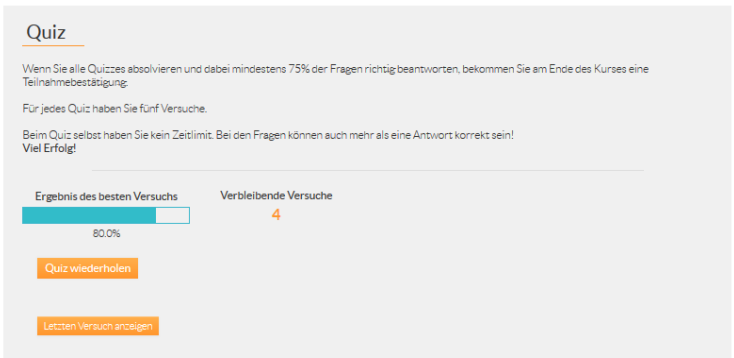
	**Lernvideo**
	**Aufgabe**
	**Pocket Karten**
	**Quiz**

Abbildung 7: Überblick über die verschiedenen Komponenten eines Kapitels

3.3 Videos

Videos sind das Herzstück eines MOOCs. So hat eine Untersuchung des ersten Kurses (6002x, Circuits and Eletronics) auf der MOOC-Plattform edX ergeben, dass TeilnehmerInnen den Großteil ihrer Zeit damit verbrachten, Videos anzuschauen (Breslow, et al., 2013) (Abbildung 8). Auch eine Studie von 3 Kursen der Plattform Coursera hat festgestellt, dass viele TeilnehmerInnen sich hauptsächlich mit Videos beschäftigen, während Kurskomponenten wie Aufgaben, Online-Diskussionen oder andere interaktive Komponenten einfach übersprungen wurden (Kizilcec , Piech, & Schneider , 2013). Aufgrund ihrer Wichtigkeit wird in diesem Kapitel die Entwicklung der Videos beschrieben.

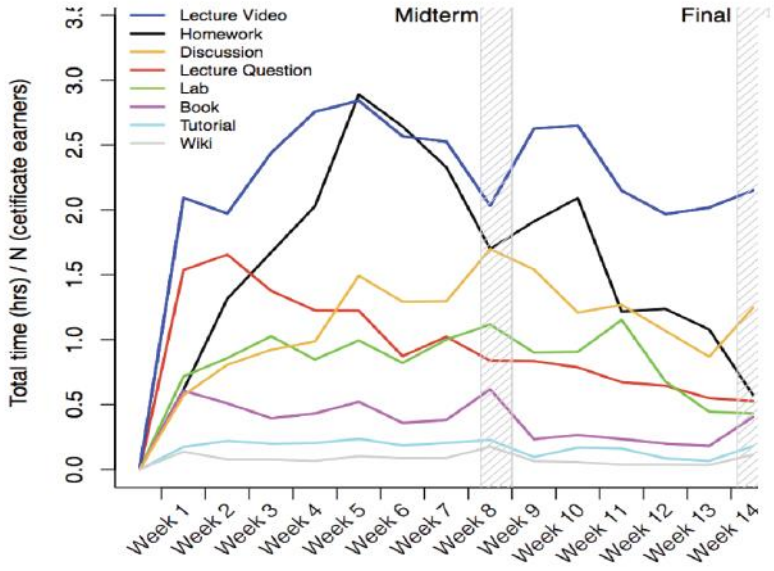


Abbildung 8: Durchschnittliche Zeit pro Woche die Teilnehmer, die Kurs abgeschlossen haben, mit den verschiedenen Kurskomponenten verbrachten (Breslow, et al., 2013).

Bei MOOCs werden häufig verschiedene Stile von Videos eingesetzt (Abbildung 9). Es stellt sich somit die Frage, wie sich diese Stile auf das Verhalten der TeilnehmerInnen auswirken.

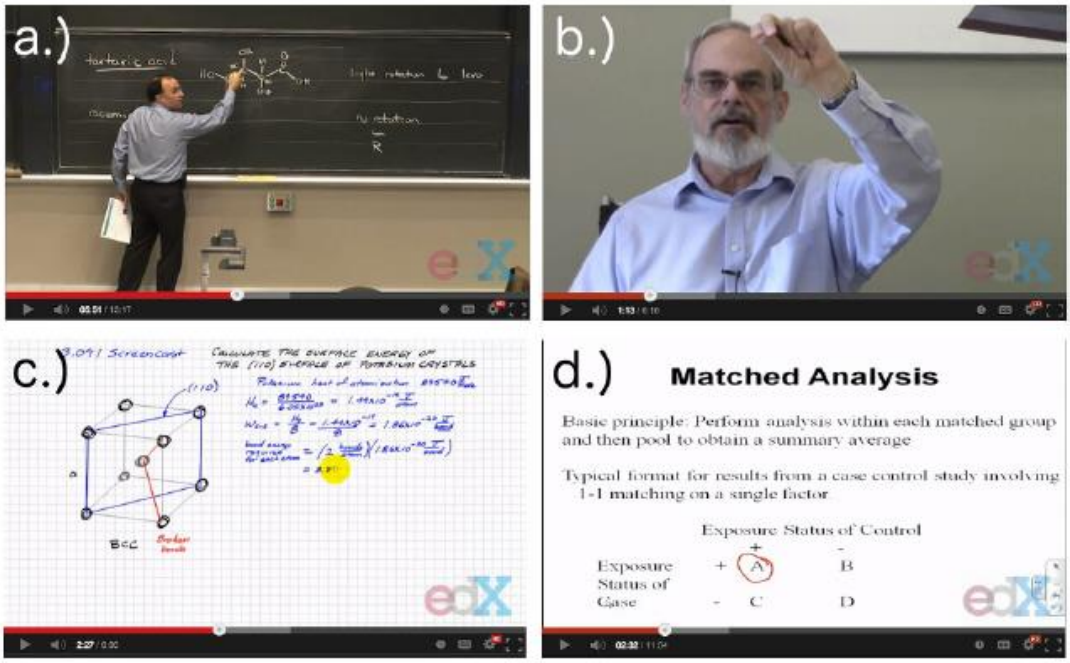


Abbildung 9: Typische Video Stile in MOOCs: a.) Lehrveranstaltung Aufzeichnung b.) „talking head" – Close Up Aufnahme eines Unterrichtenden c.) Interaktive Zeichnung am Tablet bekannt durch Khan Academy d.) PowerPoint Folien Präsentation (Guo, Kim, & Rubin, 2014).

Guo, Kim, & Rubin haben in einer empirischen Studie von 2014 6.9 Millionen „video watching sessions" von vier Kursen der edX-Plattform untersucht, um Rückschlüsse des Einflusses von Video-Produktion auf das Verhalten von MOOC-TeilnehmerInnen schließen zu können. Dabei untersuchten sie, wie lange TeilnehmerInnen ein bestimmtes Video anschauten und ob sie im Anschluss probierten Video-spezifische Fragestellungen zu lösen. Einige wichtige Hauptuntersuchungsergebnisse dieser Studie samt daraus resultierenden Empfehlungen werden in Tabelle 1 gelistet:

Untersuchungsergebnisse	***Empfehlung***
Die Länge der Videos war der stärkste Indikator für eine Beteiligung der TeilnehmerInnen. Kurze Videos waren viel fesselnder als lange Videos. Bei den kürzesten Videos (0-3 min) dauerten 75% der Sessions länger als dreiviertel der Video -länge.	Planung spielt eine wesentliche Rolle bei MOOC-Videos. Es sollte probiert werden, die Inhalte in den Videos in kurze Einheiten (max. 6 Minuten) aufzuteilen.

Videos, in denen die Lehrpersonen schnell und mit viel Enthusiasmus redeten, waren fesselnder.	Lehrpersonen sollten instruiert werden enthusiatisch zu reden; eine absichtliche Verlangsamung der Redegeschwindigkeit ist nicht notwendig.
„Khan-style“-Videos, in denen interaktiv am Tablet gezeichnet wurde, waren fesselnder als PowerPoint-Präsentationen oder Code Screencasts.	Es sollen Bewegung und Animationen in den Videos eingeführt werden.
Vorlesungs- und Tutorial-Videos wurden von den TeilnehmerInnen unterschiedlich behandelt.	Für Vorlesungen sollte mehr das erste Anschauen im Vordergrund stehen; Tutorials sollten so gestaltet werden, dass sie leichtes Wiederanschauen und Durchschauen unterstützen.

Tabelle 1: Einige Ergebnisse sowie Empfehlungen bezüglich Videoproduktion (Guo et al. 2014)

In der Entwicklung und Produktion der Videos für den Kurs auf iMooX wurde probiert auf die genannten Empfehlungen einzugehen. Da jedes Video nicht länger als max. 5 Minuten dauern sollte, wurden die Inhalte aufgespaltet und in mehrere kürzere Videos pro Kapitel verpackt. Zur Übersichtlichkeit wurde trotzdem probiert, dass jedes Video nur ein Thema beinhaltet. Aufgrund dieses thematischen Schwerpunkts lassen sich die verschiedenen Längen der Videos erklären (von einer bis max fünf Minuten). Da sich die Videos mit den Funktionen von Pocket Code beschäftigen sowie Codesegmente erklären, können sie als Tutorial-Videos bezeichnet werden. Als Stil der Videos wurden Bildschirmaufnahmen (Screen Recordings) von Pocket Code gemacht, zu denen gleichzeitig gesprochen wurde und - wenn notwendig - zusätzliche Animationen hinzugefügt wurden (Abbildung 10). Es wurde probiert, möglichst einfach und in einem normalen Tempo zu sprechen. Einige wenige Videos, wie zum Beispiel das Einleitungsvideo, hatten keine Bildschirmaufnahme, da hier Allgemeines zum Kurs erklärt wurde. Hier wurde der sogenannte „talking-head“-Stil verwendet, wobei eine Person in die Kamera spricht. Diese Person war allerdings nicht real, sondern wurde animiert, um zur Stimmung des Kurses zu passen (Abbildung 11).

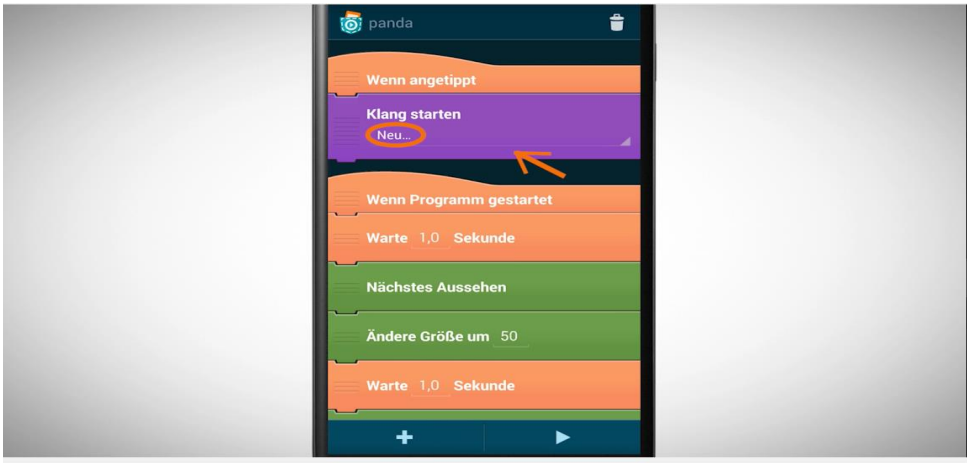


Abbildung 10: Typischer Ausschnitt aus einem Video: Bildschirmaufnahme mit gesprochenen Text und zusätzlicher kleiner Animation

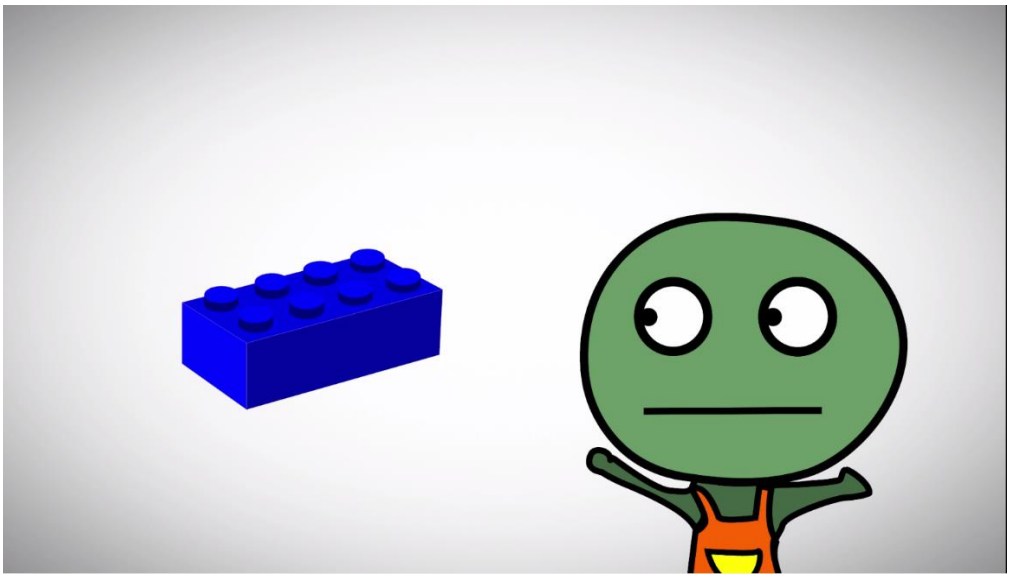


Abbildung 11: Animierter "talking head"

Die generelle Entwicklung eines Videos lässt sich in folgende Phasen unterteilen:

1. Planung des Inhalts des Videos
2. Planung und Niederschrift des gesprochenen Textes
3. Einsprechen des Textes (aufgenommen mit Adobe Audition)
4. Screen Recording am Smartphone, das ungefähr mit Text übereinstimmt (aufgenommen mit der App: Recordable)
5. Screen Recording und gesprochenen Text in Schnittprogramm bringen (Adobe Premiere Pro)

6. Zusammenschneiden, sodass gesprochener Text mit Screen Recording übereinstimmt (eventuell Pausen hinzufügen oder Abschnitte kürzen)
7. Zusätzliche Animationen hinzufügen, wenn notwendig
8. Export des Videos in MP4-Format (.h264 Codec, Auflösung: 1920 x 1080, hohe Bitrate, 24 fps)

Die Transkripte der einzelnen Videos befinden sich im Anhang.

3.4 Begleitmaterial

Neben den Videos wurde auch Begleitmaterial in schriftlicher Form für den Kurs entwickelt. Zu den einzelnen Übungsaufgaben, die mittels Video vorgestellt wurden, wurden jeweils immer eine kurze Beschreibung, ein Tipp sowie ein Lösungsvorschlag entwickelt. Die Beschreibung der Aufgabe hatte den Zweck, den TeilnehmerInnen einen Überblick über die Fragestellung zu verschaffen, ohne sich dabei wieder das Video anschauen zu müssen. Obwohl es im Video der Vorstellung der Aufgabe meistens einige Tipps zum Lösen gab, wurde ein zusätzliches Dokument mit Tipps zur Aufgabe angeboten. Diese Tipps hatten ähnlich wie die Beschreibung den Sinn, den TeilnehmerInnen eine schnelle Hilfestellung zu geben. Der Lösungsvorschlag, der verwendet werden kann, wenn gar nicht mehr weitergewusst wird, war als eine weitere Hilfe gedacht oder auch als Kontrolle. In diesem Dokument wurde eine konkrete Lösung der Aufgabe vorgestellt (Abbildung 12).

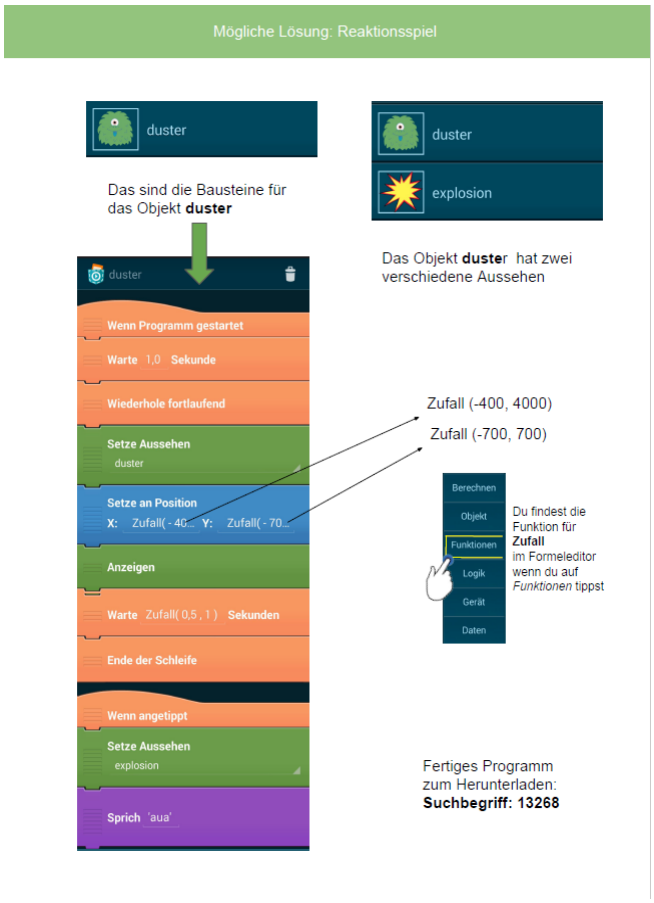


Abbildung 12: Mögliche Lösung zur Aufgabe Reaktionsspiel

Als weiteres Begleitmaterial wurden sogenannte Pocket-Karten entwickelt. Ähnlich wie es bei anderen Programmiersprachen Dokumentationen zu Funktionen, Methoden oder auch Attributen gibt, sollten Pocket-Karten die Funktion einzelner Blöcke beschreiben. Auf der Vorderseite einer Pocket-Karte wurde grafisch die Funktionsweise eines bestimmten Blockes gezeigt. Auf der Hinterseite der Karte gab es eine kurze Beschreibung dieser Funktionsweise sowie ein kurzes Beispiel, wie dieser Block verwendet werden kann (Abbildung 13). Die Pocket Karten sollten als Zweck haben einzelne wichtige Bausteine und deren möglichen Einsatz zu beschreiben. Vor allen in der Durchführung in der Schule kann damit der Trainer bzw. Lehrer entlastet werden, da sich die SchülerInnen eigenständig einen Überblick über die verschiedenen Funktionsweisen und Einsatz der Bausteine verschaffen können.

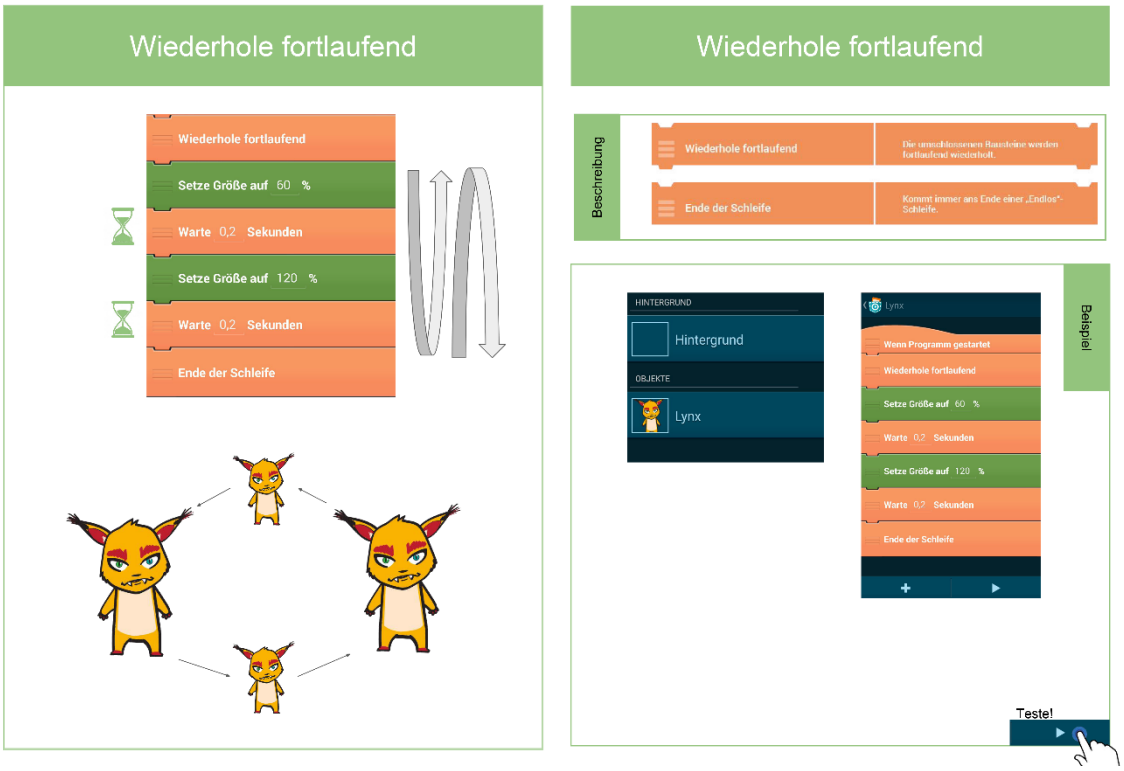


Abbildung 13: Vorder- und Rückseite der Pocket-Karte für den Block "Wiederhole fortlaufend“

3.5 Zielsetzung des Kurses

Das Ziel des Kurses war es, Kindern und Jugendlichen, aber auch allen anderen Interessierten einen Einblick in das Programmieren zu ermöglichen. Neben dem Erreichen von möglichst vielen Personen war das Hauptziel, die TeilnehmerInnen für das Programmieren und die Konzepte dahinter zu begeistern. Da es doch oft gewisse Vorurteile und Ängste gegenüber dem Programmieren gibt, sollte die Angst genommen und die TeilnehmerInnen sollten dazu angeregt werden, sich über den Kurs hinaus damit zu beschäftigen.

3.6 Beschreibung des Kurses

Der Kurs wurde wird folgt beschrieben:

Bezüglich Programmieren bestehen viele Vorurteile und Ängste. Mithilfe von Pocket Code sollen vor allem Kinder erste Erfahrungen mit dem Programmieren sammeln. Durch eine einfache und visuelle Benutzeroberfläche wird eine spielerische Umsetzung eigener Ideen ermöglicht. Der Kurs richtet sich somit an

Kinder und Jugendliche (Altersgruppe 10-14 Jahre) und hat als Hauptinhalt das Erstellen eigener Spiele, interaktiver Animationen und Apps mithilfe von Pocket Code. Primär werden dabei Struktur und Funktionsweise der App vorgestellt, im Hintergrund werden „Computational Thinking"-Konzepte erarbeitet, wie zum Beispiel Konditionale, Variablen, Events oder Parallelismus. Dabei ist es den Kindern überlassen, ob sie den Kurs selbstständig oder gemeinsam mit ihren Eltern machen.

3.7 Inhalt

Der Inhalt wurde für die Pocket Code-Version 0.9.25 entwickelt, die zum Start des Kurses (Oktober 2016) im Google-Play-Store erhältlich war. Da der Kurs in zwei verschiedenen Settings durchgeführt wurde (rein online bzw. in einer Schulklasse, siehe Kapitel 4), gab es geringe Unterschiede im Inhalt des Kurses. Im Folgenden sollen die Inhalte des Kurses beschrieben werden.

3.7.1 Kapitel 1 – Mein erstes eigenes Programm

In diesem Kapitel ging es darum, Pocket Code kennenzulernen und ein eigenes Programm damit zu entwickeln. Das erste Video beschäftigte sich mit dem Kurs selbst und gab einen kurzen Überblick über das Thema Programmieren. Es wurde kurz erklärt, warum Programmieren wichtig ist, was Pocket Code ist und wie der Kurs aufgebaut ist. Als zusätzliche Motivation wurde auch noch auf den Galaxy Game Jam hingewiesen[8], ein Wettbewerb, in dem man mit seinem Pocket-Code-Programm Preise wie Smartphones oder Roboter gewinnen konnte (Abbildung 15). Eine Grafik wurde gezeigt, um das Lernsetting darzustellen. (Abbildung 14).

[8] www.galaxygamejam.com/ (letzter Abruf am 28.01.2017)

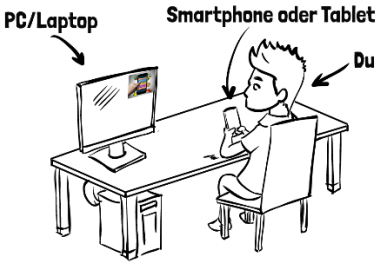


Abbildung 14: Lernsetting

Abbildung 15: Galaxy Game Jam

In den darauffolgenden acht Lernvideos wurden die Installation von Pocket Code und erste Schritte damit gezeigt. Es wurde auf die Benutzeroberfläche und den Aufbau von Pocket Code eingegangen sowie die wichtigsten ersten Schritte durchgenommen. Dazu gehörte:

- Ein neues Programm erstellen
- Ein Objekt in diesem Programm erzeugen und dieses programmieren
- Das Aussehen des Objekts verändern
- Ein Ereignis und Klang hinzufügen
- Die Position von einem Objekt verändern

Neben diesen Videos wurden noch schriftliche Kurzanleitungen angeboten (Abbildung 16), die mit dem Inhalt der Videos übereinstimmten. Aufgrund der Wichtigkeit dieser Schritte sollten diese Anleitungen als zusätzliche Hilfe hergenommen werden.

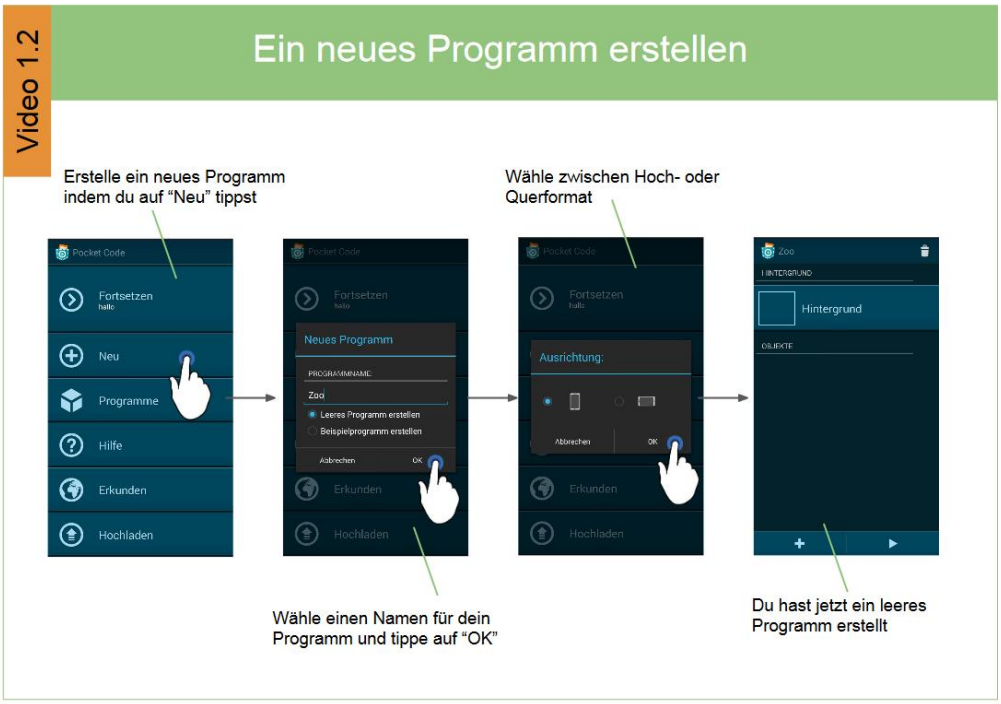


Abbildung 16: Schriftliche Kurzanleitung für Video 1.2

Nach diesen Videos sollten die TeilnehmerInnen probieren das Gelernte im Rahmen einer Aufgabe selbst umzusetzen. Als Aufgabe dazu gab es einen Zoo zu programmieren. Als Begleitmaterial wurden Pocket-Karten erstellt. Am Ende des Kapitels gab es noch ein Video, das erklärte, wie man sein Programm als App erstellen oder auf die Community-Seite hochladen kann. Des Weiteren wurde ein Thread im Forum erstellt, in dem die TeilnehmerInnen den Namen ihrer hochgeladenen Programme posten konnten. Der Kursleiter bot an, diese Programme anzuschauen, um sein Feedback dazu zu geben.

3.7.2 Kapitel 2 – Bis zur Unendlichkeit

Nachdem es im Kapitel 1 hauptsächlich um den Umgang mit Pocket Code ging, lag der Schwerpunkt von diesem Kapitel auf verschiedenen Konzepten, wie Schleifen, der Kommunikation zwischen Objekten und dem Einsatz von Sensoren. In den darauffolgenden Aufgaben mussten diese Konzepte wieder eingesetzt werden. Auch hier konnten die TeilnehmerInnen wieder den Namen ihres hochgeladenen Programms im Forum posten.

3.7.3 Kapitel 3 – Lass uns spielen!

Das Ziel dieses Kapitels war es, schon erste kleine Spiele zu programmieren. Dazu wurden Funktionen (im Speziellen die Zufallsfunktion), „If“-Anweisungen und die physikalischen Blöcke erklärt. Dazu gehörte auch, wie es möglich ist, Kollisionen zwischen Objekten zu erkennen. Die Aufgaben waren hier schon komplexer, in der letzten Aufgabe sollte schon eine simples „Pong“-Spiel programmiert werden (Abbildung 17). Als Begleitmaterial gab es die bekannten Pocket-Karten.

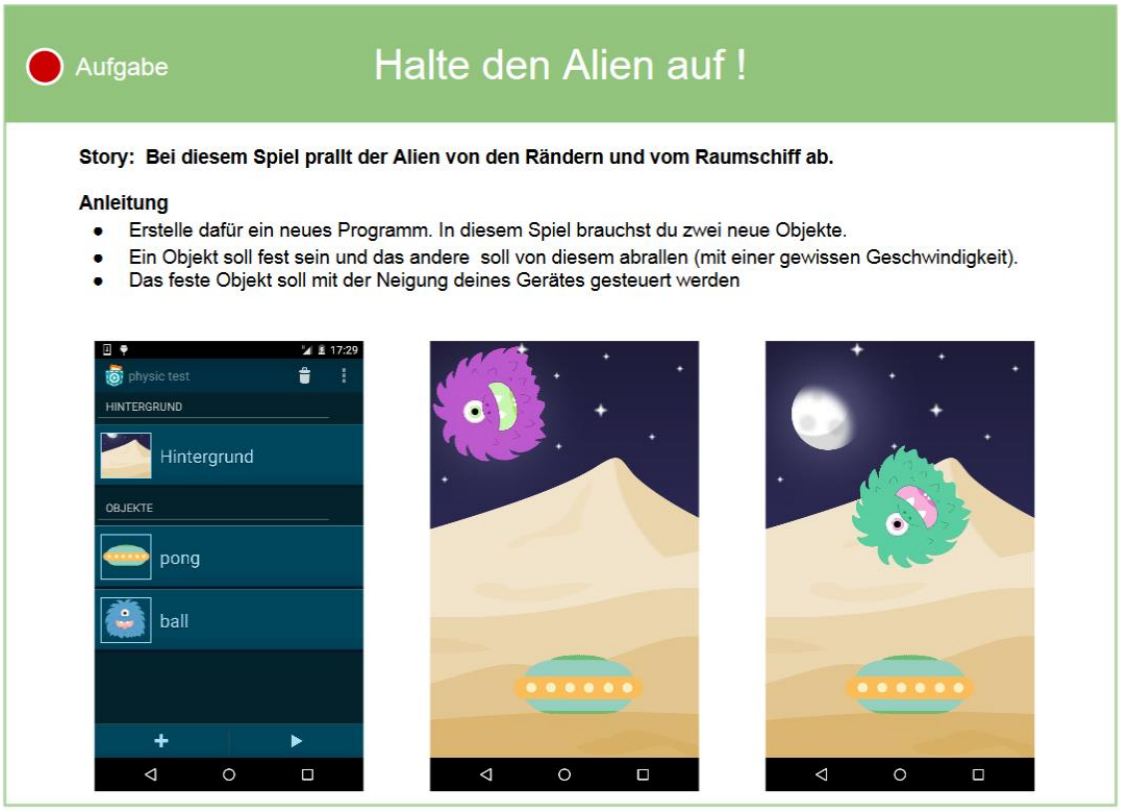


Abbildung 17: Aufgabe: "Halte den Alien auf"

3.7.4 Kapitel 4 – Punkte und Eigenschaften

Ein ganz großer Punkt in diesem Kapitel waren die Variablen. Diese wurden mithilfe von zwei verschiedenen Beispielen erklärt. Einmal wurde ein Punktestand verwendet, im anderen Video wurde gezeigt, wie es möglich ist, mit Variablen einen Countdown zu implementieren. Als Aufgabe galt es bei einem schon vorhandenen Spiel einen Punktestand einzubauen. Anschließend wur-

den noch Eigenschaften von Objekten behandelt. Da die anschließende Aufgabe schon komplexer war, galt es hier auch das Wissen von vorherigen Kapiteln einzusetzen. Die Programme konnten wieder im Forum genannt werden, um ein Feedback zu bekommen.

3.7.5 Kapitel 5 – Zielgerade

In diesem letzten Kapitel gab es inhaltliche Unterschiede zwischen den zwei Settings. Diese Unterschiede wurden aufgrund der verschiedenen Anforderungen gemacht. Für das Schulklasse-Setting wurde nur eine Aufgabe entwickelt, sodass die SchülerInnen noch einmal üben konnten. Aufgrund der zeitlichen Einschränkung (Doppelstunden) sollten sich die SchülerInnen auf die zweite Aufgabe konzentrieren. Hier war das Ziel, ein eigenes Programm zu entwickeln. Egal ob Spiel, Animation oder sonstige Apps. Es wurden keine Vorgaben gemacht, die SchülerInnen durften ihr Wissen kreativ einsetzen.
Für das reine Online-Setting wurden mehr Videos angeboten. Da in der Zwischenzeit eine neue Pocket-Version herausgekommen war (Version 0.9.26), wurden Videos angeboten, die die neuen Features zeigten. Zum Abschluss des Kapitels und des Kurses gab es auch hier die Aufgabe, ein eigenes Programm von Grund auf zu entwickeln.

3.8 Bekanntmachung des MOOCs

Da ein wesentlicher Bestandteil eines MOOCs eine hohe Teilnehmerzahl ist, galt es, den MOOC zu promoten und bekannt zu machen. Hierfür wurden verschiedene Medienkanäle genutzt. Es wurde ein kurzer Trailer produziert, mit dem auf Social Media Kanälen, wie Youtube[9] oder Facebook[10], der Kurs präsentiert wurde. Für die Printmedien wurden zwei Artikel geschrieben, einer davon wurde im Oktober-Sonderheft des Bundesministeriums für Bildung (Coding – Ein Baustein der informatischen Bildung) (Janisch, Ebner, & Slany, 2016) und der andere im L.A Multimedia 2016 (2) veröffentlicht (Janisch, Slany, & Ebner, Programmieren für Kinder, 2016). Als Werbematerial wurden Bierdeckel im Po-

[9] www.youtube.com (letzter Abruf am 15.01.2017)

[10] www.facebook.com/imoox.st (letzter Abruf am 15.01.2017)

cket-Code-Design produziert („Pocket- Deckel, Abbildung 18), die bei diversen Veranstaltungen und Gelegenheiten verteilt wurden.

Abbildung 18: Pocket-Deckel-Design (Vorder- und Rückseite)

Zusätzlich zu diesen Aktivitäten wurde noch Mundpropaganda betrieben sowie eine Information zum Kurs an die LehrerInnen der Teilnehmerklassen des „mobilen Klassenzimmers[11]“ (Workshop: Coding for Kids der Firma Samsung) ausgeschickt.

[11] http://www.samsung.com/at/microsite/digitale-bildung/coding-for-kids.html (letzter Abruf am 13.01.2016)

4. Durchführung des Kurses

4.1 Ablauf

Der Kurs wurde in zwei verschiedenen Settings durchgeführt; als „klassischer“ xMOOC (Online-Setting) und als Kurs in der Schule (Schule-Setting) (Abbildung 19). Diese beiden verschiedenen Abläufe sollen kurz beschrieben werden.

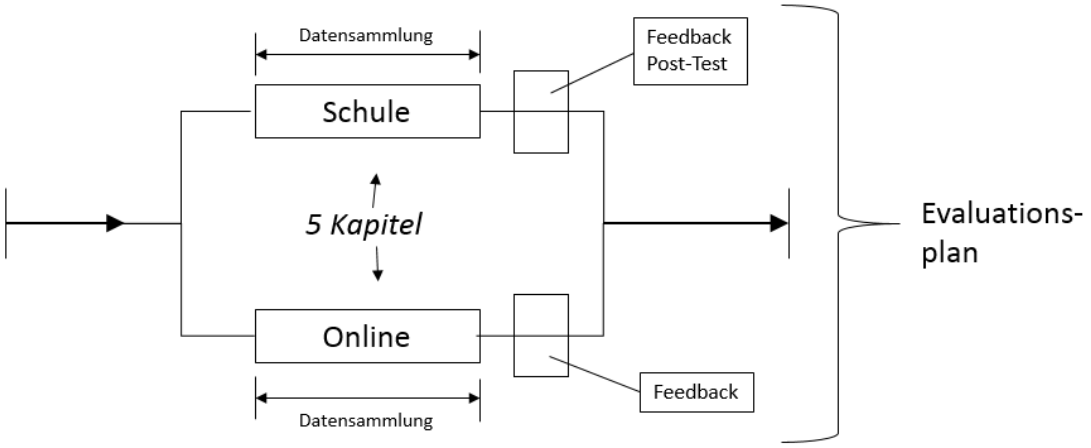


Abbildung 19: Durchführung des Kurses

4.1.1 Ablauf als reiner MOOC-Kurs

Der Kurs als reine xMOOC-Variante wurde auf der MOOC-Plattform iMooX durchgeführt. Start des Kurses war der 17. Oktober 2016. Jede Woche wurde ein weiteres Kapitel freigeschaltet, das letzte Kapitel somit am 14. November 2016. In jeder Woche gab es Videos, Aufgaben und ein Quiz zu lösen. Neben allgemeinen Themen im Forum wurde auch jede Woche ein neuer Thread erzeugt, in dem die TeilnehmerInnen ihre Programme posten konnten. Da es in der zweiten Woche von einer Teilnehmerin bemängelt wurde, dass der Schritt von Woche 1 zu Woche 2 zu groß wäre, wurde hier noch ein zusätzliches Video produziert (Video 2.0 Objekt zeichnen), um Hilfestellung zu geben. Wurden alle Quiz-Aufgaben positiv absolviert, musste am Ende noch ein Feedback-Formular ausgefüllt werden, um die Teilnahmebestätigung zu erhalten. Neben

der Teilnahmebestätigung gab es noch die Möglichkeit sich digitale Badges für einzelne Kapiteln oder den gesamten Kurs (Abbildung 20) erstellen zu lassen.

Abbildung 20: Pocket Code Abschluss Badge

4.1.2 Ablauf in der Schule

Im anderen Setting wurde der Kurs mit einer Schulklasse (1. Klasse AHS, Wahlfach Informatik) durchgeführt. Damit sollte das sogenannte „Inverse-Blended-Learning“-Konzept angewendet werden. Beim „Inversed-Blended-Learning“- Lernkonzept soll die virtuelle Welt des Onlinekurses mit der realen Lebenswelt der TeilnehmerInnen, in diesem Fall der SchülerInnen, verknüpft werden. Im Gegensatz zum Blended Learning, wo der traditionelle Unterricht im Klassenzimmer mit E-Learning-Phasen ergänzt wird, wird bei Inverse Blended Learning ein reiner Online-Kurs mit Offline-Angeboten ergänzt. Damit soll vor allem der hohen Dropout-Rate von MOOCs entgegengewirkt werden (Ebner, Schön, & Käfmuller, 2015).

Der MOOC-Kurs „Learning to Code – Programmieren mit Pocket Code“ wurde auf iMooX durchgeführt, das Offline-Angebot gab es in Form des anwesenden Kursleiters. Der Kursleiter befand sich während der gesamten Durchführung vor Ort und hatte die Aufgabe, eine Art Trainer zu sein, der diesen Online-Kurs offline begleitete und ergänzte. Dazu gehörten Aufgaben wie das Geben von Instruktionen, die Hilfe bei verschiedenen Fragestellungen oder auch die Motivation von SchülerInnen. Neben dem Trainer konnten die SchülerInnen natürlich auch untereinander kommunizieren. Aufgrund dieser Möglichkeit des realen

Austauschens mit Trainer und MitschülerInnen wurde das Forum in diesem Setting nicht benutzt.

Der Kurs wurde jeweils immer in einer Doppelstunde der Wahlfaches Informatik der 1. Klassen (Gymnasium) abgehalten und wurde von 14 SchülerInnen besucht. Die erste Doppelstunde fand am 4. Oktober statt; 2 Wochen, bevor der reine Online-Kurs startete. Dieser Kurs wurde somit geklont[12], um ausschließlich den SchülerInnen zur Verfügung zu stehen. Am Anfang der ersten Einheit galt es, den SchülerInnen den Ablauf näherzubringen. Um die Zeit möglichst gut nützen zu können, wurde vom Trainer für jeden Schüler und jede Schülerin ein Konto auf iMooX sowie für Pocket Code Community Seite erstellt. Jeder Schüler und jede Schülerin bekamen daraufhin ein kleines Kärtchen, auf dem ihre Benutzerdaten aufgelistet waren (Abbildung 20).

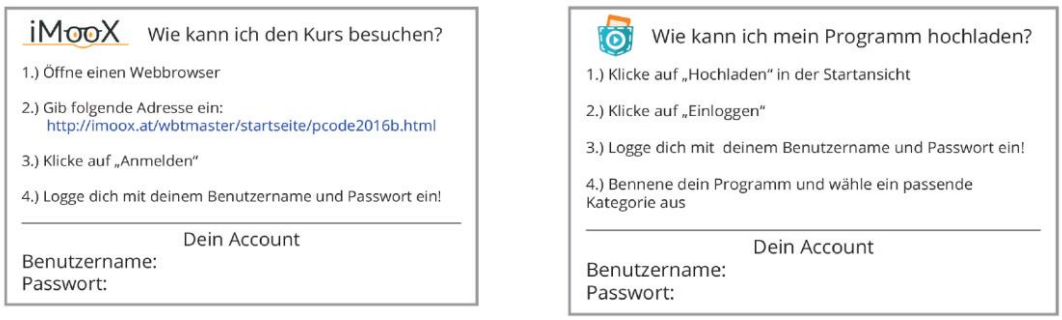


Abbildung 21: Vorder- und Rückseite des Kärtchens für die Benutzerdaten

Nach einer Instruktion über den Ablauf wurden den SchülerInnen ein Kärtchen und ein Tablet ausgehändigt. Anschließend konnten die SchülerInnen ihre mitgebrachten Kopfhörer am PC anschließen, sich auf iMooX mit ihren Daten anmelden und mit dem Kurs beginnen. Abbildung 21 verdeutlicht dieses Lernsetting.

[12] Kurs für die Schule: http://imoox.at/wbtmaster/courseMooc.htm?pcode2016**b** (letzter Abruf am 07.12.2016)

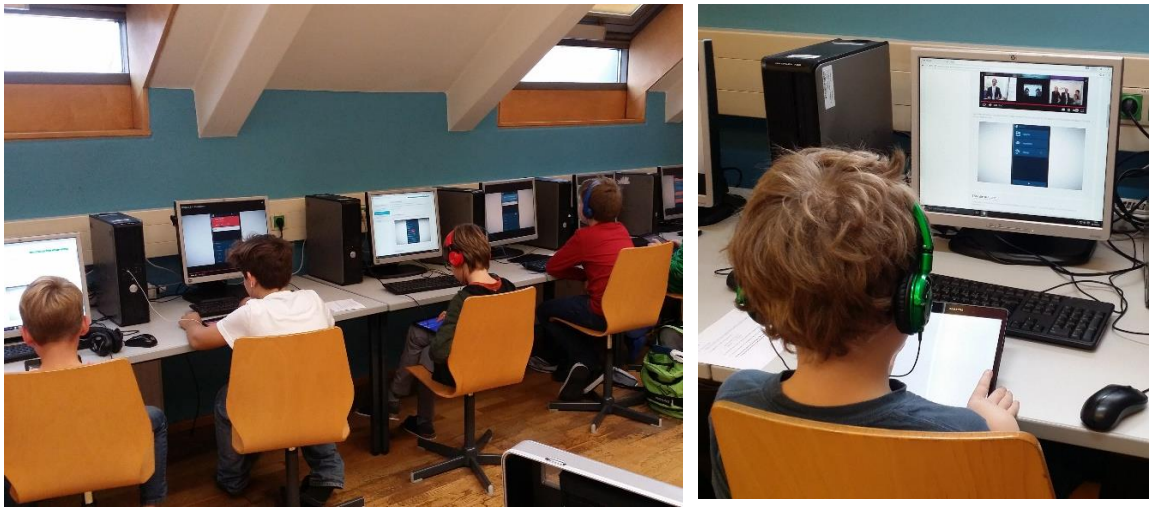


Abbildung 22: Lernsetting in der Schule

Bei Fragen zeigten die SchülerInnen auf und der Trainer kam, um weiterzuhelfen. Wenn alle Videos angeschaut waren, gab es die Aufforderung, das Quiz zu beantworten und die Aufgaben zu bearbeiten. Da einige SchülerInnen schneller waren als andere, wurden ihnen zusätzliche Aufgaben gegeben. Am Ende der Doppelstunde sollten die SchülerInnen ihre Programme auf die Community-Seite von Pocket Code laden. Dieser Ablauf wurde für die restlichen Einheiten beibehalten. Da zwischen der vierten und der letzten Einheit wegen eines Feiertages 2 Wochen lagen, wurde den SchülerInnen am Ende der vierten Doppelstunde ein Ausblick auf die letzte Einheit gegeben. Es wurde erwähnt, dass als Abschlussprojekt ein eigenes Programm entwickelt werden sollte. Den SchülerInnen wurde damit die Möglichkeit gegeben, schon Ideen für ihr Programm zu sammeln. Natürlich durfte auch zuhause auf dem eigenen Gerät programmiert werden. Am Ende der letzten Einheit gab es schließlich noch eine Präsentation der eigenen entwickelten Programme (Abbildung 22).

Abbildung 23: Präsentation der eigenen Programme

4.2 Evaluationspläne

Die Evaluationspläne dienten als Grundlage für Analyse und Auswertung des MOOCs. In der ersten Spalte findet sich die Zielsetzung des Kursleiters. In der nächsten Spalte finden sich Fragen, mit denen das Erreichen der Zielsetzung überprüft werden kann. Die restlichen Spalten beinhalten, wie Daten gesammelt wurden und den Zeitraum.

Tabelle 2: Evaluationsplan für das Online-Setting

Zielsetzung	**Fragestellung der Evaluation**	**Evaluationsquellen und Auswertung**	**Zeitraum**
Es sollen Materialien zum Thema „Programmieren mit Pocket Code“ geschaffen werden, die auch genutzt werden	Welche und wie oft bzw. wie viel wurden bestimmte Materialien benutzt (Videos, Pocket Karten, schriftliche Anleitungen)?	Learning Analytics-Auswertung von iMooX, Daten von Youtube Analytics	Ende des Kurses

Die TeilnehmerInnen sollen in der Lage sein, die Konzepte und das Gelernte selbst einzusetzen	Wie wurden die Aufgaben erfüllt? Wie wurde in den Quiz-Bearbeitungen abgeschnitten?	Auswertung der abgegeben Programme und der Quiz-Bearbeitungen	Während und Ende des Kurses
Allgemeine Kursanalyse – Kann man informatische Bildung mit einem MOOC vermitteln?	Wie veränderte sich das Lernverhalten während des Kurses?	Analyse der abgeschlossen Quiz-Bearbeitungen, Video-Verhalten über die Dauer des Kurses, Forumsbeteiligung, abgegebene Programme	Ende und Beobachtung während des Kurses
	Wie viele Personen haben den Kurs abgeschlossen, Wann gab es die meisten Dropouts?	Beobachtung, wann weniger Videos angeschaut, Programme abgegeben und Quizfragen beantwortet wurden	Während und am Ende des Kurses
	Welche Personen haben den MOOC gemacht?	Auswertung des Feedbackformulars	Ende des Kurses
	Wie ist der Kurs bei den TeilnehmerInnen angekommen?	Auswertung des Feedbackformulars Beiträge im Forum	Ende des Kurses

Tabelle 3 Evaluationsplan für das Schul-Setting

Zielsetzung	**Fragestellung der Evaluation**	**Evaluationsquellen und Auswertung**	**Zeitraum**
Es sollen Materialien zum Thema „Programmieren mit Pocket Code“ geschaffen werden, die im Unterricht verwendet werden.	Welche und wie oft bzw. wie viel wurden bestimmte Materialen benutzt (Videos, Pocket-Karten, schriftliche Anleitungen)?	Learning Analytics-Auswertung von iMooX, Daten von Youtube Analytics	Ende des Kurses

Die Schüler sollen in der Lage sein, die Konzepte und das Gelernte selber einzusetzen	Wie wurden die Aufgaben erfüllt? Wie komplex war das Abschluss-Programm? Können Konzepte der Programmierung angewendet werden?	Auswertung der abgegeben Programme und des Abschlusstests	Während und Ende des Kurses
Allgemeine Kursanalyse – kann man einen MOOC in der Schule einsetzen, um informatische Bildung zu vermitteln?	Wie veränderte sich das Lernverhalten während des Kurses?	Analyse des Video-Verhaltens und der abgegebenen Programme über die Dauer des Kurses	Ende und während des Kurses beobachten
	Wie war die Motivation der Schüler über die Dauer des Kurses?	Beobachtung, wann weniger Programme abgegeben wurden, subjektive Beobachtung des Verhaltens der Kinder	Während und am Ende des Kurses
	Wie haben den SchülerInnen das Thema und das Lernen via MOOC gefallen?	Auswertung des Feedbackformulars.	Ende des Kurses

5. Evaluation

In diesem Kapitel werden die gesammelten Daten beschrieben. Da es zwei verschiedene Settings gegeben hat, werden die Daten für diese zwei Settings separat gelistet. Bei der Durchführung als reiner Online-Kurs waren am Ende 571 (Stand 01.12.2016) TeilnehmerInnen angemeldet, die Schulklasse bestand aus 14 SchülerInnen.

5.1 Videos

Aufgrund eines Flash-Problems auf iMooX, das während der Durchführung des Kurses bestand, wurden alle Videos mittels Iframe eingebunden, um die Analyse Tools von Youtube für die Auswertung der Videos verwenden zu können. Youtube Analytics zeichnet neben der Anzahl der Videoaufrufe (Views) auch andere Parameter auf, wie zum Beispiel wie lange das jeweilige Video angeschaut wurde. Daraus kann die durchschnittliche Wiedergabedauer eines Videos (Average view duration), sowie der durchschnittliche Prozentsatz der Wiedergabe des Videos (Average percentage viewed) für einen bestimmten Zeitraum erhoben werden. Hier gilt es zu beachten, dass es Werte über 100% geben kann, die durch Zurückspulen und erneutes Ansehen zustande kommen. Sind NutzerInnen, während sie ein Video anschauen, mit ihrem Youtube-Konto angemeldet, können zusätzlich noch Daten über Alter, Geschlecht und Herkunft erhoben werden. Die Daten, die für die Auswertung der Videos verwendet wurden, waren die Aufrufe, die durchschnittliche Wiedergabedauer und der durchschnittliche Prozentsatz der Wiedergabe. Die folgenden Tabellen geben eine Übersicht über das Videoverhalten der TeilnehmerInnen und SchülerInnen in den jeweiligen Kapiteln wieder.

5.1.1 Videos Online-Setting

Hier wurden die Videos ausgewertet, die die TeilnehmerInnen Im Rahmen des Online-Kurses benutzten. Als Zeitraum der Daten wurde der Start des jeweiligen Kapitels der Videos bis zum 01.01.2017 gewählt.

Video Name	Duration	Views	Average view duration (min)	Average percentage viewed (%)
1.0 Herzlich willkommen!	02:36	419	01:45	67%
1.1 Installation	01:21	469	00:48	60%
1.2 Neues Programm erstellen	01:00	401	00:45	79%
1.3 Aufbau von Pocket Code	01:16	369	00:59	78%
1.4 Neues Objekt programmieren	03:43	363	02:51	77%
1.5 Aussehen verändern	02:01	331	01:36	80%
1.6 Ereignis und Klang	02:47	318	02:10	78%
1.7 Position verändern	04:47	283	03:01	63%
1.8 Hintergrund, Löschen, Verschieben	01:50	230	01:32	84%
1.9 Aufgabe: Programmiere einen Zoo	01:34	274	01:05	70%
1.10 Hochladen von einem Programm	03:28	142	02:10	63%
Mittelwerte Kapitel 1	**2:23**	**327**	**1:42**	**73%**

Tabelle 4: Videoverhalten Kapitel 1, Online-Setting

Die Videos des ersten Kapitels hatten durchschnittlich 327 Aufrufe bei einer durchschnittlichen Wiedergabedauer von 1:42 Minuten. Hochgerechnet aus den Daten der mit YouTube verbundenen TeilnehmerInnen waren die User dabei zu 64% männlich. Die TeilnehmerInnen kamen überwiegend aus Österreich (69%) und Deutschland (26%) gefolgt von Weißrussland (2,4%), Italien (1,2%), der Schweiz (1,5%) und vereinzelten Usern aus Serbien (0,7%), Ungarn (0,2%), Indien (0,5%) und Frankreich (0,2%).

Video Name	Duration	Views	Average view duration (min)	Average percentage viewed (%)
2.0 Objekte zeichnen	02:48	170	02:20	84%
2.1 Schleifen	02:42	475	02:17	85%
2.2 Aufgabe: Wir programmieren eine Animation	00:58	171	00:51	89%
2.3 Kommunikation von Objekten	02:49	181	02:30	89%
2.4 Aufgabe: Eine kleine Button-Steuerung	00:17	141	00:19	114%
2.5 Sensoren einbauen	03:52	164	03:16	85%
2.6 Aufgabe: Verfolgungsjagd	01:24	124	01:22	98%
Mittelwerte Kapitel 2	**02:07**	**204**	**1:50**	**92%**

Tabelle 5: Videoverhalten Kapitel 2, Online-Setting

Im Kapitel 2 des Kurses wurden die Videos im Durchschnitt 204-mal aufgerufen mit einem durchschnittlichen Prozentsatz der Wiedergabe von 92. Auffallend ist hier das Video 2.1 Schleifen, das mit 475 Aufrufen deutlich mehr hatte als die anderen Videos des gleichen Kapitels. Die User waren dabei zu 69% männlich und kamen aus Österreich, Deutschland, Weißrussland, Ungarn, Italien und Serbien.

Video Name	Duration	Views	Average view duration (min)	Average percentage viewed (%)
3.1 Der Zufall	03:28	141	02:36	75%
3.2 Aufgabe: Reaktionsspiel	00:37	130	00:35	95%
3.3 Wenn Anweisungen (if)	04:38	107	03:35	78%
3.4 Aufgabe: Ich verstecke mich	00:24	80	00:25	105%
3.5 Physikalische Objekte	04:36	125	03:24	81%
3.6 Kollisionen	02:33	103	02:12	86%
3.7 Aufgabe: Halte den Alien auf	01:19	88	01:19	101%
Mittelwerte Kapitel 3	**02:30**	**110**	**02:00**	**89%**

Tabelle 6: Videoverhalten Kapitel 3, Online-Setting

Die Videos von Kapitel 3 wurden im Mittel 110-mal aufgerufen und dabei zu 89% der Wiedergabezeit angeschaut. Die User waren zu 74% männlich und kamen hauptsächlich aus Österreich (70%). Andere Länder waren Deutschland, Italien, Serbien, Ungarn und Weißrussland.

Video Name	Duration	Views	Average view duration (min)	Average percentage viewed (%)
4.1 Punktestand	03:48	112	02:43	72%
4.2 It's the final countdown	04:43	75	04:13	90%
4.3 Aufgabe: Baue einen Punktestand ein	01:21	49	01:10	88%
4.4 Objekt Eigenschaften	05:00	56	03:46	76%
4.5 Aufgabe: Ausweichen Spiel	02:07	64	01:27	69%
Mittelwerte Kapitel 4	**03:23**	**71**	**02:39**	**79%**

Tabelle 7: Videoverhalten Kapitel 4, Online-Setting

Im Kapitel 4 hatte nur das erste Video (4.1 Punktestand) über 100 Aufrufe, die restlichen Videos hatten weniger als 80 Aufrufe; im Durchschnitt hatten alle Videos 71 Aufrufe bei einer 79%igen Wiedergabe der Länge der Videos. 75 % waren männlich, die Länder, aus denen die TeilnehmerInnen kamen, waren

diesselben wie bei den vorherigen Kapiteln (Österreich, Deutschland, Serbien, Weißrussland, Italien)

Video Name	Duration	Views	Average view duration (min)	Average percentage viewed (%)
5.1 Klonen von Objekten	02:10	76	01:35	74%
5.2 Touchscreen Steuerung	02:57	54	02:18	78%
5.3 Malstift	01:26	47	01:06	77%
5.4 Fragen stellen	02:21	44	02:46	90%
5.5 Aufgabe: Star Wars Spiel	01:37	43	01:36	100%
5.6 Dein eigenes Programm	00:47	42	00:32	70%
Mittelwerte Kapitel 5	**01:53**	**51**	**01:38**	**82%**

Tabelle 8: Videoverhalten Kapitel 5, Online-Setting

Im letzten Kapitel wurden die Videos im Durchschnitt 51-mal aufgerufen bei einer durchschnittlichen Wiedergabedauer von 01:38 Minuten (entspricht 82% des durchschnittlichen Prozentsatzes der Wiedergabe). 54% der Nutzer waren männlich, die Länder der UserInnen waren Österreich und Deutschland.

5.1.2 Videos Schule-Setting

Auch für das Video-Verhalten der Schülerinnen im Rahmen der Durchführung des Kurses in der Schulklasse wurden die Parameter durchschnittliche Wiedergabedauer eines Videos (Average view duration) sowie der durchschnittliche Prozentsatz der Wiedergabe des Videos (Average percentage viewed) analysiert. Die Schulklasse bestand aus 14 SchülerInnnen, wobei 2 davon weiblich waren. Als Startzeitpunkt der Beobachtung wurde das Datum gewählt, an dem die Videos für die SchülerInnen freigeschaltet worden waren; als Endzeitpunkt das Datum, bis zu dem die Videos für die TeilnehmerInnen des reinen Online-Kurses zugänglich waren. Da der Kurs im Schul-Setting früher ablief als der Online-Kurs, konnte eine Überschneidung der Daten verhindert werden.

Video Name	Duration	Views	Average view duration (min)	Average percentage viewed (%)
1.0 Herzlich willkommen!	02:36	12	01:22	53%
1.1 Installation	01:21	16	00:45	56%
1.2 Neues Programm erstellen	01:00	25	00:48	80%

1.3 Aufbau von Pocket Code	01:16	24	01:06	88%
1.4 Neues Objekt programmieren	03:43	22	03:03	82%
1.5 Aussehen verändern	02:01	16	01:47	89%
1.6 Ereignis und Klang	02:47	16	02:42	97%
1.7 Position verändern	04:47	14	04:29	94%
1.8 Hintergrund, Löschen, Verschieben	01:50	14	01:42	93%
1.9 Aufgabe: Programmiere einen Zoo	01:34	15	01:09	74%
Mittelwerte Kapitel 1	**2:23**	**17**	**1:53**	**81%**

Tabelle 9: Videoverhalten Kapitel 1, Schul-Setting

Die Videos des ersten Kapitels wurden im Durchschnitt 17-mal aufgerufen. Die ersten beiden Videos weisen einen durchschnittlichen Prozentsatz der Wiedergabe von weniger als 60 auf, der Rest der Videos wurde zu mehr als 70% ihrer Länge angeschaut.

Video Name	Duration	Views	Average view duration (min)	Average percentage viewed (%)
2.1 Schleifen	02:42	22	02:27	91%
2.2 Aufgabe: Wir programmieren eine Animation	00:58	16	00:43	75%
2.3 Kommunikation von Objekten	02:49	17	02:42	96%
2.4 Aufgabe: Eine kleine Button Steuerung	00:17	16	00:17	104%
2.5 Sensoren einbauen	03:52	18	04:22	113%
2.6 Aufgabe: Verfolgungsjagd	01:24	13	01:39	118%
Mittelwerte Kapitel 2	**02:07**	**17**	**02:01**	**100%**

Tabelle 10: Videoverhalten Kapitel 2, Schul-Setting

Für die Videos des 2. Kapitels gab es 17 Aufrufe, wobei die Videos im Durchschnitt in ihrer vollen Länge angesehen wurden. Die letzten 3 Videos weisen einen Prozentsatz über 100 auf, was auf Zurückspulen und erneutes Ansehen hindeutet.

Video Name	Duration	Views	Average view duration (min)	Average percentage viewed (%)
3.1 Der Zufall	03:28	20	02:23	69%
3.2 Aufgabe: Reaktionsspiel	00:37	18	00:42	114%
3.3 Wenn Anweisungen (if)	04:38	18	03:34	77%
3.4 Aufgabe: Ich verstecke mich	00:24	15	00:24	104%
3.5 Physikalische Objekte	04:36	18	04:24	96%
3.6 Kollisionen	02:33	18	02:09	85%
3.7 Aufgabe: Halte den Alien auf	01:19	16	01:40	127%
Mittelwerte Kapitel 3	**02:30**	**18**	**02:10**	**96%**

Tabelle 11: Videoverhalten Kapitel 3, Schul-Setting

Im 3. Kapitel gab es im Mittel 18 Aufrufe, die durchschnittliche Wiedergabedauer war 2:10 min.

Video Name	Duration	Views	Average view duration (min)	Average percentage viewed (%)
4.1 Punktestand	03:48	18	03:41	97%
4.2 It's the final countdown	04:43	17	03:17	70%
4.3 Aufgabe: Baue einen Punktestand ein	01:21	15	01:10	86%
4.4 Objekt Eigenschaften	05:00	10	04:05	82%
4.5 Aufgabe: Ausweichen Spiel	02:07	12	02:29	118%
Mittelwerte Kapitel 4	**03:23**	**14**	**02:56**	**91%**

Tabelle 12: Videoverhalten Kapitel 4, Schul-Setting

Die Videos des 4. Kapitels wurden 14-mal aufgerufen, bei einem durchschnittlichen Prozentsatz der Wiedergabelänge von 91%.
Für das 5. Kapitel gab es in diesem Setting keine Videos, da die Zeit genutzt werden sollte, um ein eigenes Programm zu entwickeln.

5.1 Quiz

In diesem Abschnitt werden die Daten der Quiz-Bearbeitungen gelistet. Es wurde untersucht, in welchem Ausmaß die Quiz-Aufgaben von den TeilnehmerInnen im Online- und im Schul-Setting absolviert wurden. Für die Absolvierung eines Quiz mussten 75% Prozent der Fragen richtig beantwortet werden.

5.1.1 Quiz Online-Setting

Von 571 TeilnehmerInnen absolvierten 160 mindestens 1 Quiz positiv. 45 von diesen 160 TeilnehmerInnen schlossen genau ein Quiz ab, alle 5 Quiz wurden von 70 TeilnehmerInnen bearbeitet (Abbildung 23). 64 davon bekamen, nachdem sie noch den Online-Feedback-Bogen ausgefüllt hatten, eine Teilnahmebestätigung.

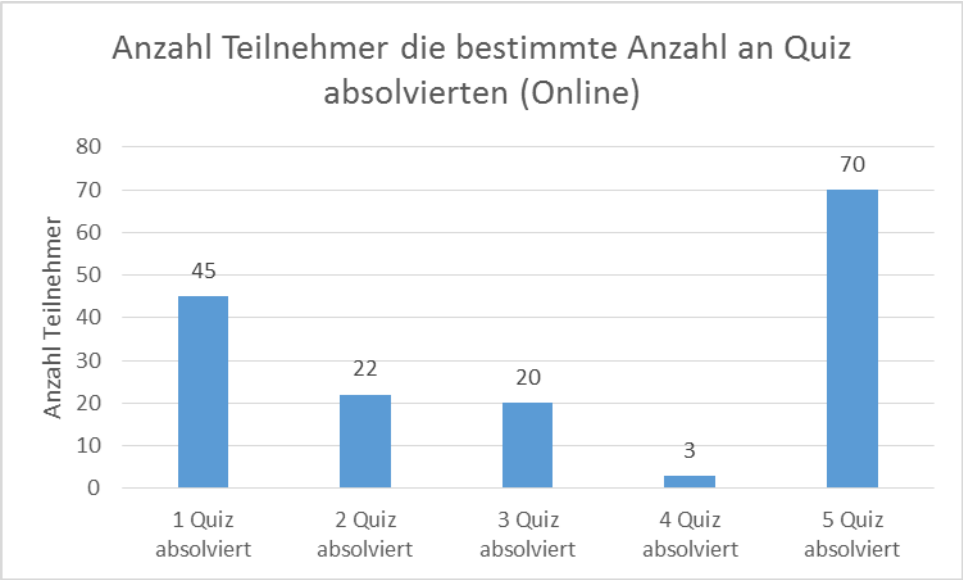


Abbildung 24: Anzahl von Teilnehmerinnen, die genau eine bestimmte Anzahl an Quiz positiv absolvierten im Online-Setting.

Abbildung 24 zeigt, wie oft ein Quiz eines Kapitels absolviert wurde. Das Quiz des ersten Kapitels (Quiz 1) wurde 151-mal positiv abgeschlossen; in den darauffolgenden Kapiteln zeigte sich eine stetige Abnahme. Das Quiz von Kapitel 5 wurde 71-mal beantwortet.

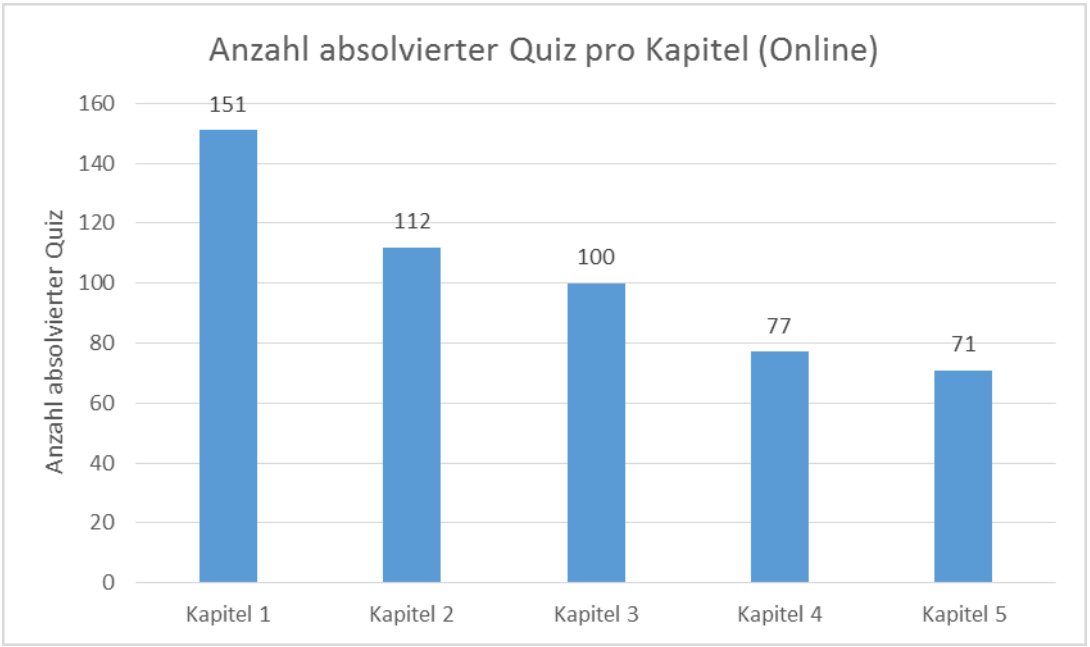


Abbildung 25: Anzahl absolvierter Quiz pro Kapitel im Online-Setting

5.1.2 Quiz Schul-Setting

Auch in der Schule gab es am Ende eines Kapitels ein Quiz. Ausnahme bildete Kapitel 5, in dem es kein Quiz gab, da hier ein verpflichtender Test für alle SchülerInnen zum Einsatz kam. Insgesamt absolvierten 9 SchülerInnen 1 oder mehrere Quiz Aufgaben. Alle 4 Quiz wurden von 2 SchülerInnen abgeschlossen (Abbildung 25)

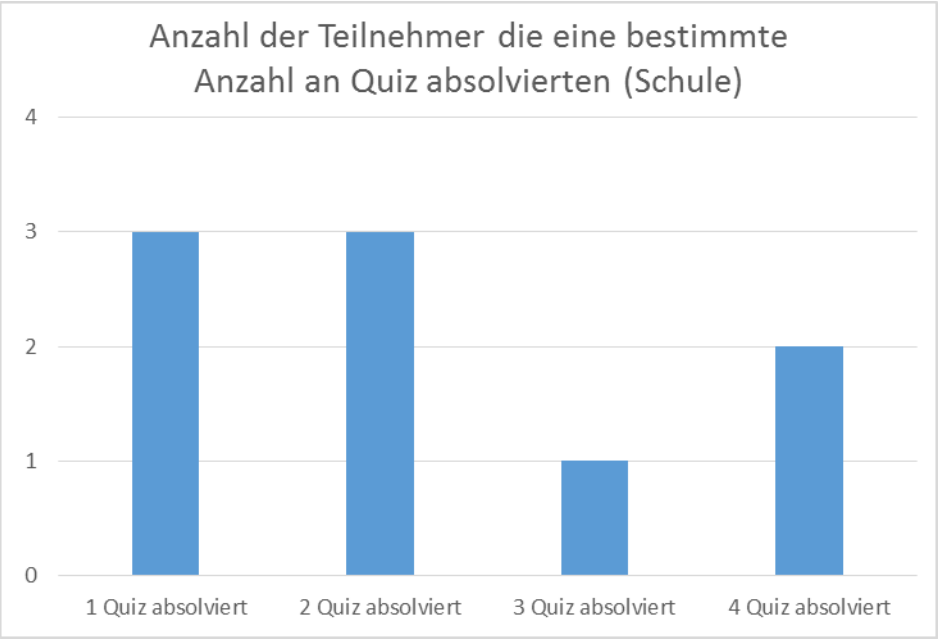


Abbildung 26: Anzahl von Teilnehmerinnen, die eine bestimmte Anzahl an Quiz-Aufgaben positiv absolvierten im Schul-Setting

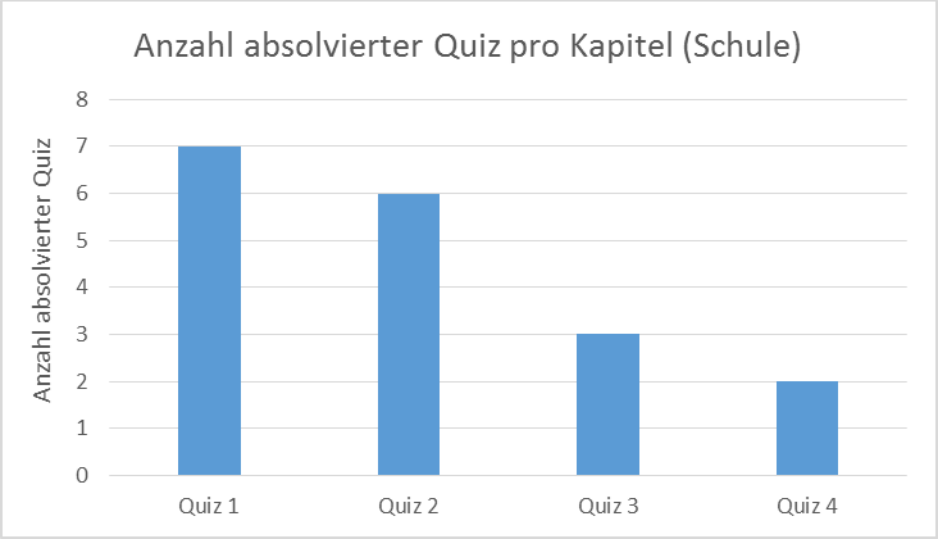


Abbildung 27: Anzahl absolvierter Quiz pro Kapitel im Online-Setting

Der Test, der am Ende des Kurses zum Einsatz kam, wurde den SchülerInnen auf Papierform ausgehändigt, um sicherzugehen, dass er von allen bearbeitet wird (siehe Anhang). Im Gegensatz zu den Quiz, die während des Kurses zum Einsatz kamen, beschäftigte sich dieser Test mit dem gesamten Inhalt des Kurses. Um an die Gegebenheiten des Kurses anzuschließen, wurde dieser Test als Quiz bezeichnet. Insgesamt wurden 5 Fragen gestellt, die als Rätsel bezeichnet wurden, jedes dieser Rätsel beschäftigte sich dabei mit einem bestimmten Konzept. Das Konzept des ersten Rätsels befasste sich mit Ausführung von Programmcode. Beim 2. Rätsel war das Koordinatensystem im Zentrum, das 3. und 4. Rätsel beschäftigte sich mit Variablen und Schleifen. Bei Rätsel 3 gab es eine Schleife mit einer Variablen, bei Rätsel 5 gab es 2 Variablen sowie eine Schleife mit einer Bedingung. Das letzte Rätsel beschäftigte sich mit einer einfachen Bedingung. Um die SchülerInnen nicht zu überfordern, wurden die Fragen alle in Pseudo-Code beschrieben, der sich an Pocket orientierte. Abbildung 27 verdeutlicht die Ergebnisse dieses Tests.

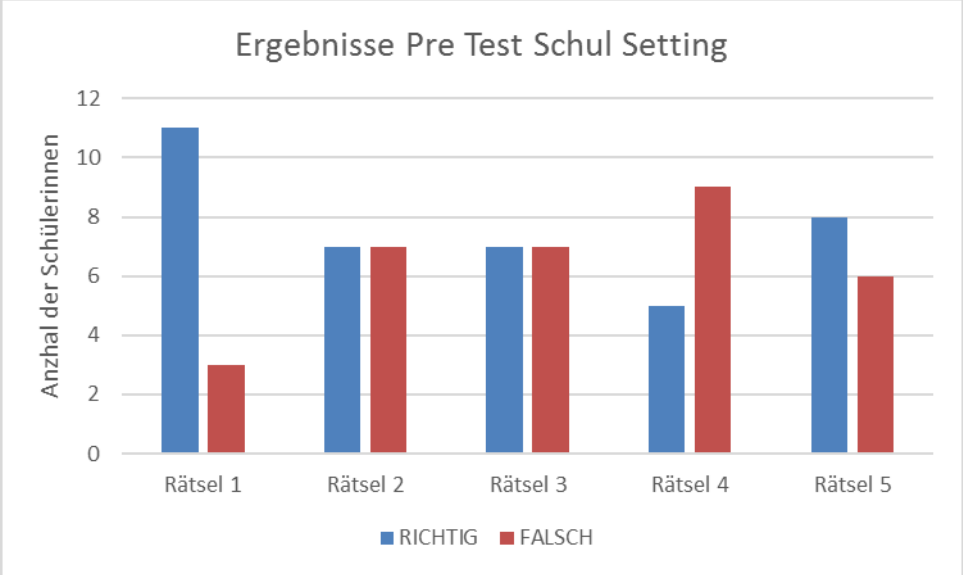


Abbildung 28: Übersicht der richtigen und falschen Antworten der SchülerInnen im Pre-Test

5.2 Forum

Im Schul-Setting wurde das Forum aufgrund des direkten Austausches mit MitschülerInnen und Trainer nicht genutzt. Die Benutzung des Forums wurde somit nur im Online-Setting ausgewertet. Es wurde analysiert, wie viele Beiträge ge-

postet und wie oft Beiträge angesehen bzw. gelesen wurden. Neben allgemeinen Themen wie „Fragen, technische Probleme, …“, „Spielwiese“, „sonstiges“ und „Vorstellung“ gab es für jedes Kapitel einen eigenen Forumsthread, in dem die Namen von hochgeladenen Programmen gepostet werden konnten. Während der Dauer des Kurses wurden im gesamten Forum 118 Beiträge verfasst, nimmt man die Beiträge des Kursleiters hinzu, erhöht sich die Zahl auf 224. Diese Beiträge im Forum wurden 4296-mal angeklickt, mit Kursleiter 5044-mal (Tabelle 11).

	Ohne Kursleiter	*Mit Kursleiter*
Gepostete Forumsbeiträge	118	224
Forumsbeiträge gelesen	4296	5044

Tabelle 13: Übersicht Forumsbeiträge gesamter Kurs

Abbildung 28 und 29 verdeutlichen die Anzahl der geposteten und angeklickten Forumsbeiträge über die gesamte Kursdauer ohne die Beiträge des Kursleiters. Das Kursende beinhaltet den Zeitraum nach Ende des Kurses (21.11.2016) bis zum 31.01.2017

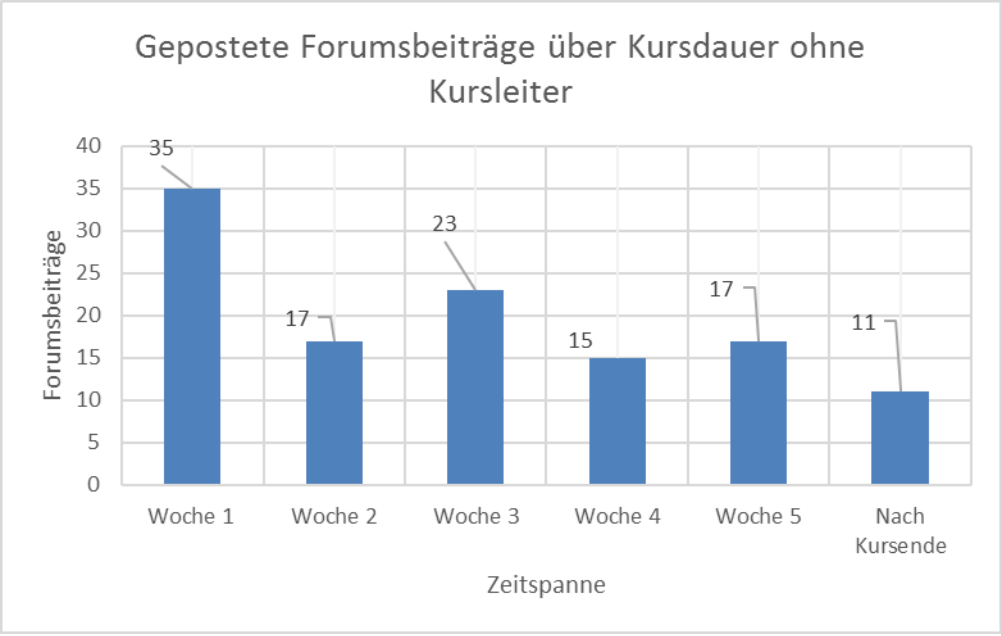


Abbildung 29: Gepostete Forumsbeiträge über Kursdauer, Erhebungszeitraum Kursende: 21.11.2016 – 31.01.2017

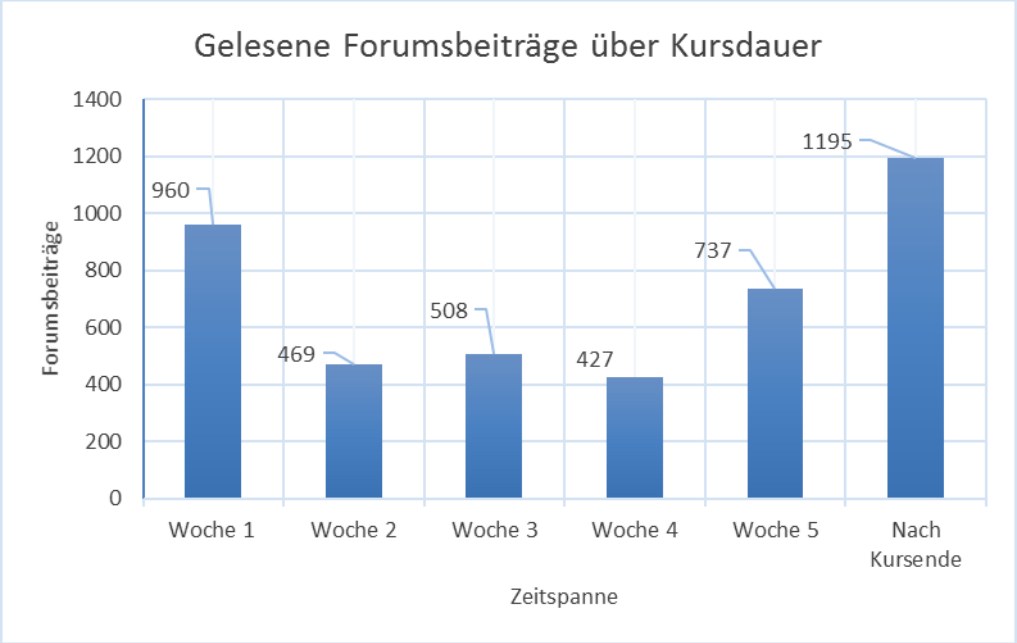


Abbildung 30: Angeklickte Forumsbeiträge über Kursdauer, Erhebungszeitraum Kursende: 21.11.2016 – 31.01.2017

Zusätzlich zu der Analyse der Forumsbeiträge wurde noch erhoben, wann die TeilnehmerInnen im Forum aktiv waren. Abbildung 30 und 31 veranschaulichen diese Daten.

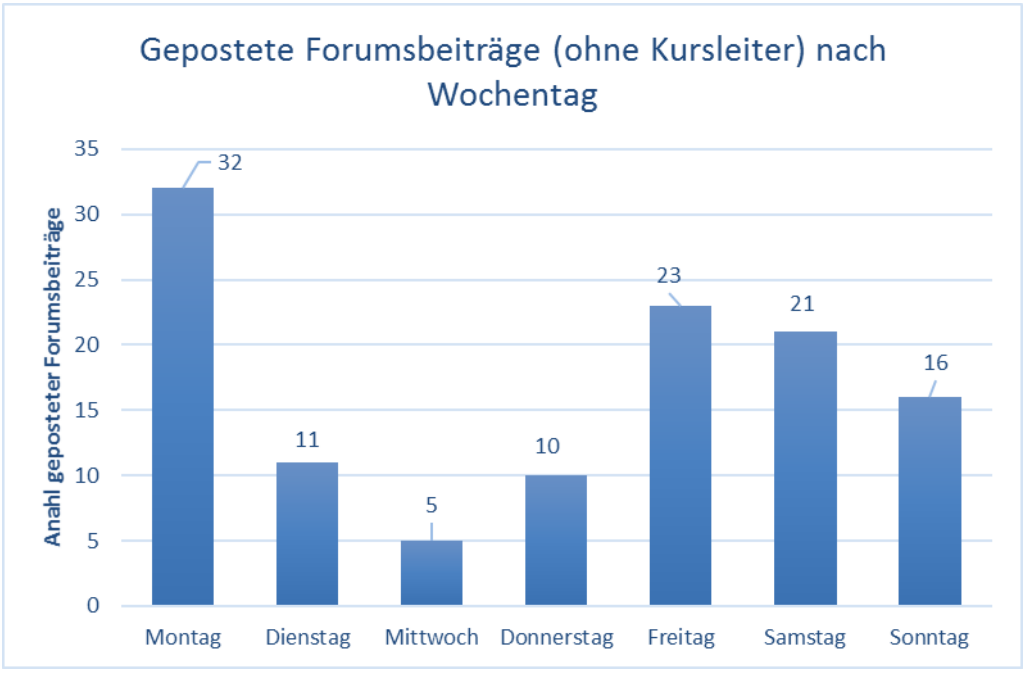


Abbildung 31: Gepostete Forumsbeiträge (ohne Kursleiter) eingeteilt nach Wochentag

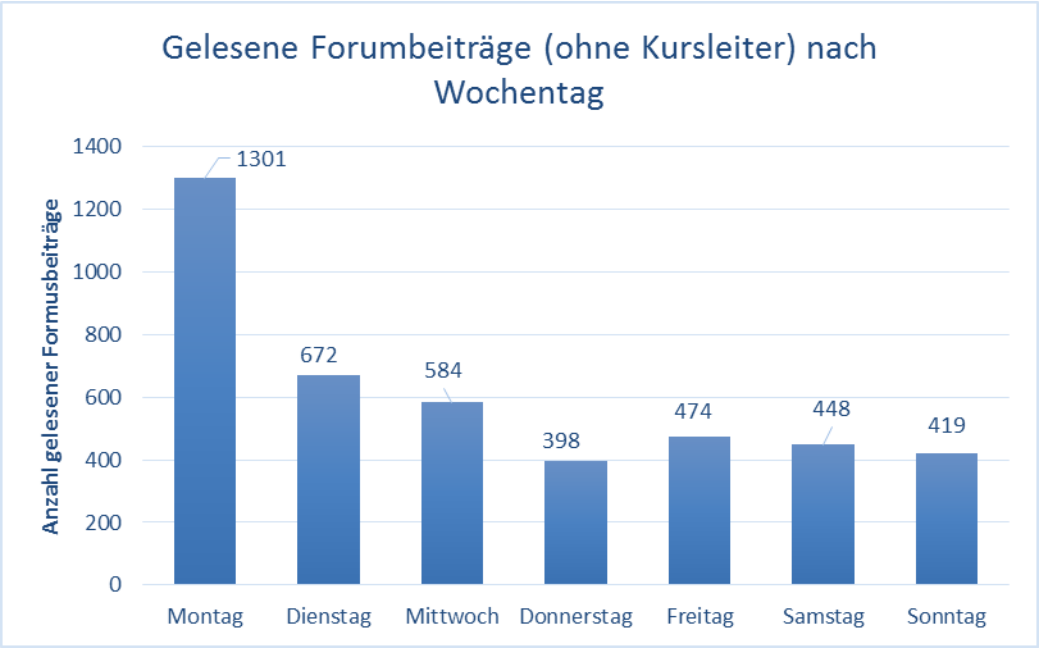


Abbildung 32: Gelesene Forumsbeiträge (ohne Kursleiter) eingeteilt nach Wochentage

5.3 Begleitmaterial

Zum Begleitmaterial wird all das Material gezählt, das neben den Videos angeboten wurde. Dazu gehören die schriftlichen Anleitungen, Tipps und Lösungsvorschläge zu den einzelnen Aufgaben sowie die Pocket-Karten. In diesem Kapitel soll analysiert werden, welche Art des Begleitmaterials im Online-Setting wie oft aufgerufen worden ist.

Das Begleitmaterial wurde während des gesamten Kurses insgesamt 2769-mal aufgerufen. Immer wenn ein PDF geöffnet wurde, wurde dies als Aufruf gewertet. Die Dauer eines solchen Aufrufes konnte nicht erfasst werden. Das Begleitmaterial kann in verschiedene Kategorien eingeteilt werden. Zu jeder Aufgabe gab es neben einem Video eine schriftliche Anleitung, einen Tipp sowie eine mögliche Lösung. Die sogenannten Pocket-Karten gaben einen Überblick über die Funktion von bestimmten Bausteinen. In der ersten Woche gab es zusätzlich zu den Videos noch schriftliche Kurzanleitungen, die einen schnellen Überblick über die gezeigten Inhalte geben sollten. Abbildung 32 verdeutlicht, welche Art von Begleitmaterialien wie oft aufgerufen wurde. Am meisten wurden die Kurzanleitungen von Woche 1 aufgerufen, gefolgt von den Pocket-Karten.

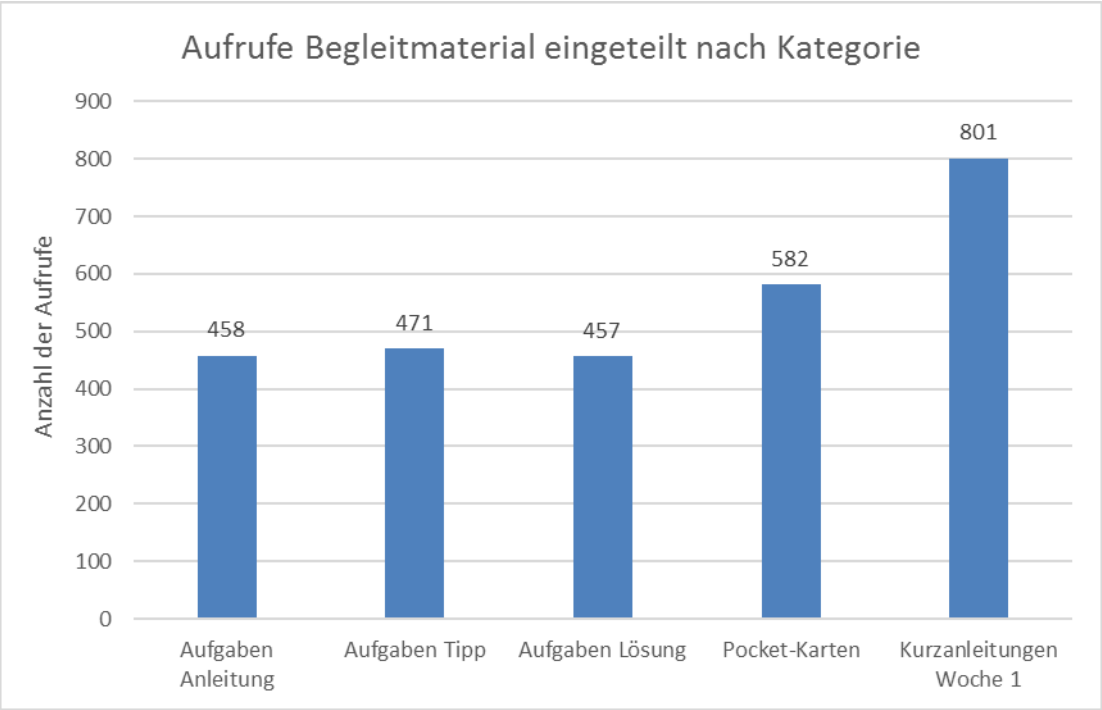


Abbildung 33: Anzahl der Aufrufe der verschiedenen angebotenen Begleitmaterialien

Der Einteilung der Aufrufe des Begleitmaterials nach Kurswochen lässt erkennen, dass es in der ersten Woche die meisten Aufrufe gab. Bis zur 5. Woche sinken die Aufrufe; in der letzten Woche gibt es wieder einen Anstieg (Abbildung 33)

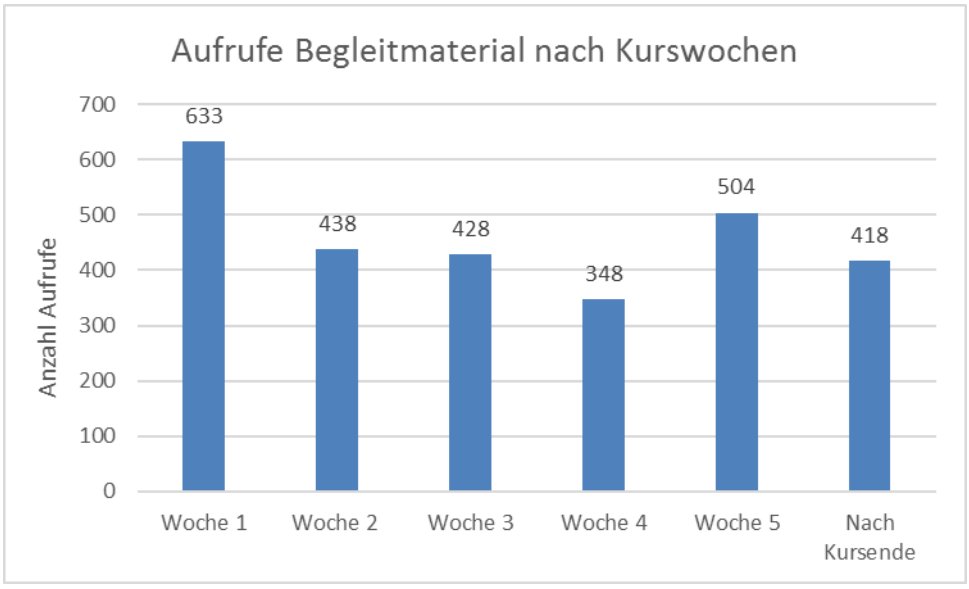


Abbildung 34: Anzahl der Aufrufe der Begleitmaterialien eingeteilt nach Kurswochen

5.4 Abgegebene Programme

In diesem Kapitel wird ein Blick auf die abgegebenen Programme der TeilnehmerInnen des Online-Kurse und der SchülerInnen geworfen.

5.4.1 Abgegebene Programme Online-Setting

Im reinen Online-Kurs gab es jede Woche einen Thread im Forum, der „Kapitel X-Programme“ hieß (X bezeichnet die jeweilige Nummer des Kapitels). In diesem Thread konnten die TeilnehmerInnen die Namen ihrer hochgeladenen Programme posten. Diese wurden in weiterer Folge vom Kursleiter angesehen, um ein entsprechendes Feedback zu geben. Die Programme waren zum Großteil Programme zu den im jeweiligen Kapitel gestellten Aufgaben; vereinzelt waren aber auch Eigenkreationen zu finden. Gaben im ersten Kapitel 7 Personen ihre Programme ab, sank diese Zahl im 5. Kapitel auf 3 Personen (Abbildung 34).

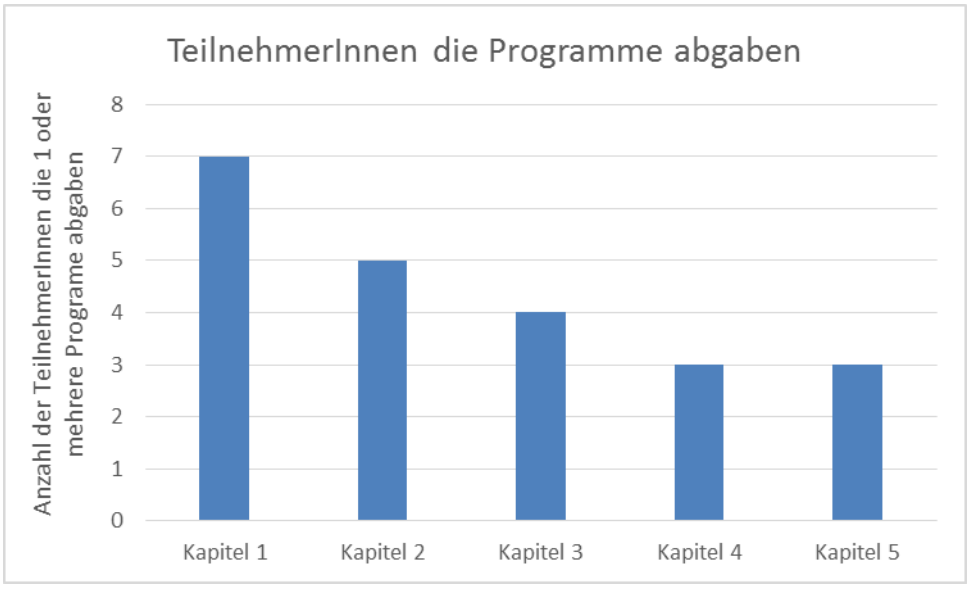


Abbildung 35: Übersicht über die Anzahl der TeilnehmerInnen, die ein oder mehrere Programme posteten, eingeteilt nach Kapiteln

Im letzten Kapitel wurde die Aufgabe gestellt, ein eigenes Programm von Grund auf zu entwickeln. Es gab keine Vorgaben wie bei den vorherigen Kapiteln, die gelernten Konzepte sollten kreativ eingesetzt werden. Ein paar dieser Abschluss-Programme werden hier kurz vorgestellt:

Mozartkugel (https://share.catrob.at/pocketcode/program/18812, letzter Abruf am 10.04.2017): In diesem Spiel gilt es, die vorbeifliegenden Ufos abzuschießen. Durch Tippen auf den Alien werden Mozartkugeln abgefeuert. Immer wenn ein Ufo getroffen wird, erhöht sich der Punktstand um eins und das Ufo ändert die Farbe (Abbildung 35).

Hunger! (https://share.catrob.at/pocketcode/programm/21233, letzter Abruf am 10.04.2017)**:** Ziel des Spieles ist es so viele Punkte wie möglich zu sammeln. Um das Ufo zu bewegen muss das Smartphone hin- und hergeneigt werden. Wenn der Bildschirm berührt wird, schickt das Ufo einen Alien nach oben, wo sich die schwarzen Punkte befinden. Berührt der Alien ein oder mehrere Punkte, werden diese von ihm eingesammelt. Danach fällt er wieder hinunter und muss vom Ufo aufgefangen werden. Wird er nicht aufgefangen oder kollidiert er mit einem herumschwebenden Hindernis, ist das Spiel vorbei (Abbildung 36).

Alien retten (https://share.catrob.at/pocketcode/program/18670, letzter Abruf am 10.04.2017): In diesem Programm fallen zwei verschiedene Aliens herunter. Ziel ist es, den hellgrünen Alien mit dem Ufo, das mit der Neigung gesteuert wird, zu fangen und dem dunkelgrünen Alien auszuweichen oder ihn abzuschießen. Alle 10 Sekunden wechselt der Hintergrund sein Aussehen, kollidiert der dunkelgrüne Alien mit dem Ufo, ist das Spiel vorbei (Abbildung 37).

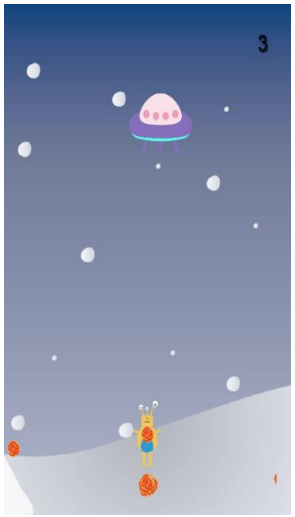


Abbildung 36: Abschluss Programm Mozart-kugel

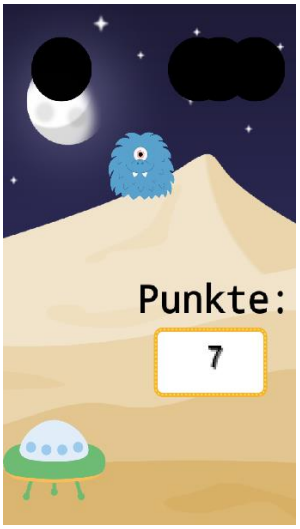


Abbildung 37: Abschluss Programm Hunger!

Abbildung 38: Abschluss Programm Alien retten

5.4.2 Abgegebene Programme Schul-Setting

Im Schul-Setting wurden die SchülerInnen aufgefordert nach Ende jeder Einheit ihre Programme auf die Community-Seite von Pocket hochzuladen. Da der Benutzername jedes einzelnen Schülers und Schülerin bekannt war, konnte so im Nachhinein festgestellt werden, welcher SchülerIn welche und wie viele Programme erstellte. Den SchülerInnen war es möglich, Fragen zu stellen, und der Kursleiter konnte laufend Feedback zu den Programmen geben. Zusätzlich gab

es einen Austausch der Kinder untereinander. Obwohl die SchülerInnen am Ende der Stunde daran erinnert worden waren, ihre Programme hochzuladen, wurde nur in der ersten Woche von jedem Schüler und jeder Schülerin mindestens ein Programm hochgeladen. In den darauffolgenden Einhiten war diese Zahl geringer (Abbildung 38). Hier muss allerdings erwähnt werden, dass einige SchülerInnen zusammen an Programmen arbeiteten sowie die Tatsache, dass manche SchülerInnen ihre Programme unter einem falschen Account hochluden.

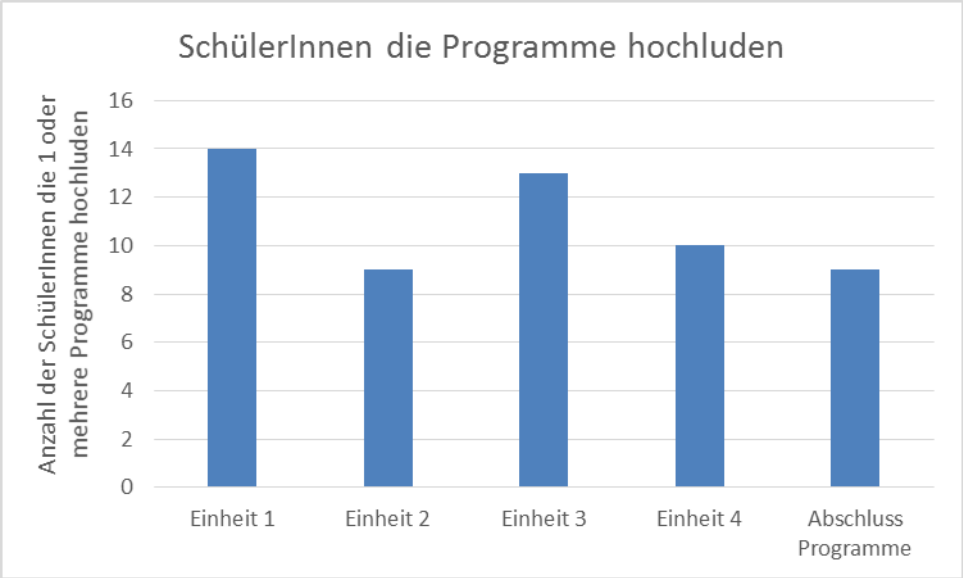


Abbildung 39: Übersicht über die Anzahl der SchülerInnen, die ein oder mehrere Programme hochluden, eingeteilt nach Einheit.

Hauptaugenmerk wurde auf die Abschlussprogramme gelegt. Für diese Programme gab es keine Vorgaben, die SchülerInnen konnten ihr Wissen anwenden und ihrer Kreativität freien Lauf lassen. Am Ende der letzten Stunde präsentierte jedes Kind sein Programm, allerdings stellten hier nur 8 Kinder ihre Programme auch auf die Pocket-Code-Community-Seite. Hier darf allerdings der angesproche Kollaborations-Aspekt nicht vergessen werden. Im Folgenden werden drei dieser Abschlussprogramme kurz vorgestellt.

Abbildung 40: Abschluss Programm Früchte Fangen

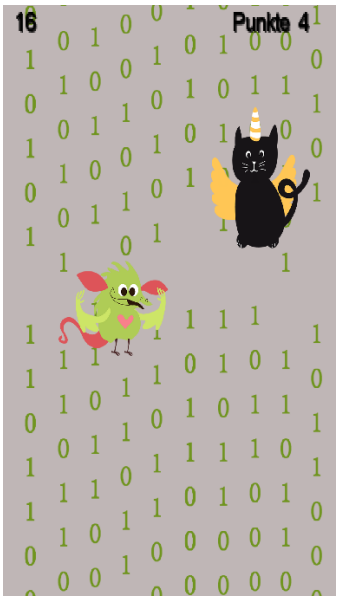


Abbildung 41: Abschluss Programm Monster Antipp Spiel

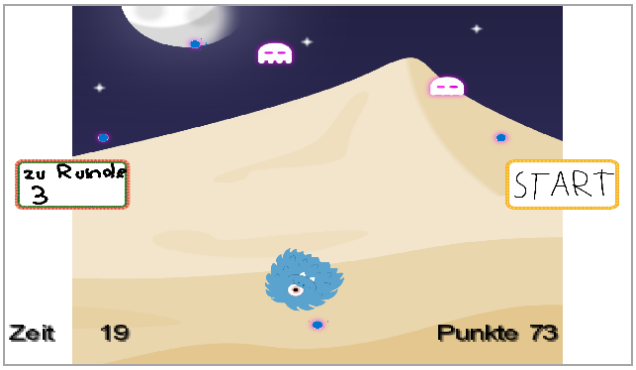


Abbildung 42: Abschluss Programm Punkte einsammeln

Früchte fangen (https://share.catrob.at/pocketcode/programm/17888, letzter Abruf am 10.04.2017): Dieses Spiel wurde von 3 SchülerInnen entwickelt. Ziel des Spieles ist es, möglichst viele Früchte mit dem Obstkorb zu fangen. Der Obstkorb wird mit der Neigung des Smartphones gesteuert. Immer wenn Obst zu Boden fällt oder der Planet gefangen wird, werden Punkte abgezogen. Erreicht der Punktestand 0 Punkte, ist das Spiel vorbei (Game Over); ab einem gewissen Punktestand verändert sich der Hintergrund (Levels) (Abbildung 39).

Monster-Antipp-Spiel: (https://share.catrob.at/pocketcode/programm/17890, letzter Abruf am 10.04.2017): In diesem Spiel, das von einem Schüler pro-

grammiert wurde, tauchen zwei Figuren am Bildschirm auf, eine schwarze Katze und ein grünes Monster. Ziel des Spiels ist es, innerhalb eines Zeitlimits das grüne Monster so oft wie möglich anzutippen. Beide Figuren tauchen allerdings nur sehr kurz und immer an einer anderen Stelle am Bildschirm auf. Tippt man auf die Katze, wird ein Punkt abgezogen (Abbildung 40).

Punkte einsammeln (https://share.catrob.at/pocketcode/programm/16911, letzter Abruf am 10.04.2017)**:** Dieses Spiel wurde von einer Schülerin in Zusammenarbeit mit ihrer Mutter entwickelt. Das Spiel besteht aus drei Runden, in jeder Runde gilt es möglichst viele Punkte zu sammeln, ohne von den herumschwebenden Geistern erwischt zu werden. Die Figur wird dabei mit der Neigung des Smartphones gesteuert, in jeder Runde kommen mehr Geister hinzu und die Spielfigur wird kleiner. Wird man von einem Geist erwischt oder verstreicht das Zeitlimit von 25 Sekunden, dann ist das Spiel vorbei (Abbildung 41).

An diesen Beispielen sieht man, dass vor allem Spiele als Thema des Abschlussprogramms gewählt worden sind. Lediglich 2 der 14 SchülerInnen programmierten eine Animation.

5.5 Feedback

Diese Ergebnisse befassen sich mit den erhobenen Daten aus dem Online-Feedback-Bogen, den es am Ende des Online-Kurses gab, und dem Feedback-Bogen, der am Ende des Kurses in der Schule ausgeteilt wurde.

5.5.1 Feedback Online-Setting

Zur erfolgreichen Absolvierung des Kurses musste neben der Beantwortung der Quiz-Fragen ein Online-Feedback-Bogen ausgeteilt werden. Dieser Feedback-Bogen beinhaltete Fragen zum Kurs sowie Fragen über demografische Daten. Der Bogen wurde von 69 TeilnehmerInnen ausgefüllt, von denen 46% weiblich und 54% männlich waren (Abbildung 42).

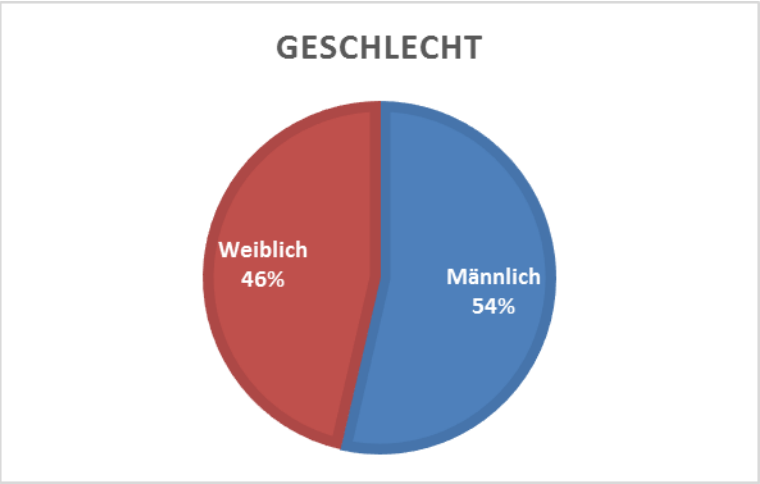


Abbildung 43: Geschlecht der Kursteilnehmerinnen

Unter den 69 TeilnehmerInnen, die den Fragebogen ausgefüllt haben, waren 30 TeilnehmerInnen im Alter von 11 -15 Jahren; der Rest verteilte sich auf die anderen Altersgruppen (Abbildung 43).

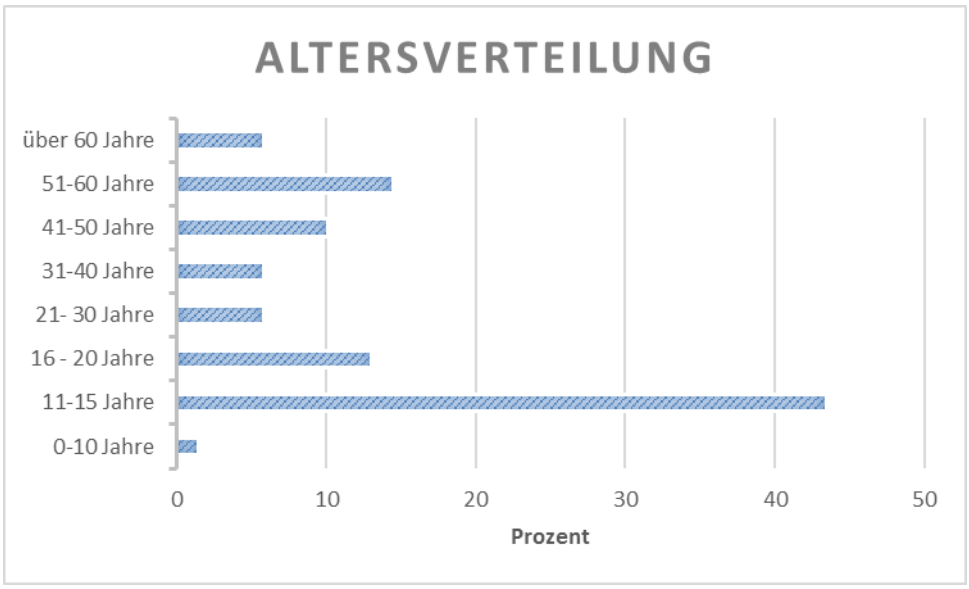


Abbildung 44: Altersverteilung der KursteilnehmerInnen in Prozent

Die meisten KursteilnehmerInnen kamen aus der Steiermark und deutschsprachigen EU (Abbildung 44). Zum Zeitpunkt der Absolvierung des Kurses waren 35 TeilnehmerInnen SchülerInnen, gefolgt von 18 in Vollzeit Berufstätigen (Abbildung 45).

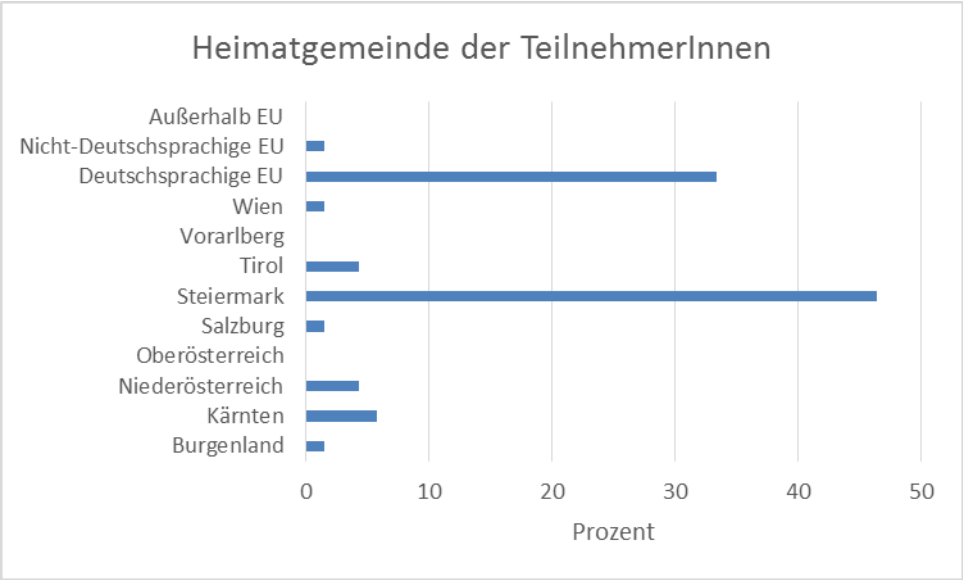


Abbildung 45: Heimatgemeinde der TeilnehmerInnen in Prozent

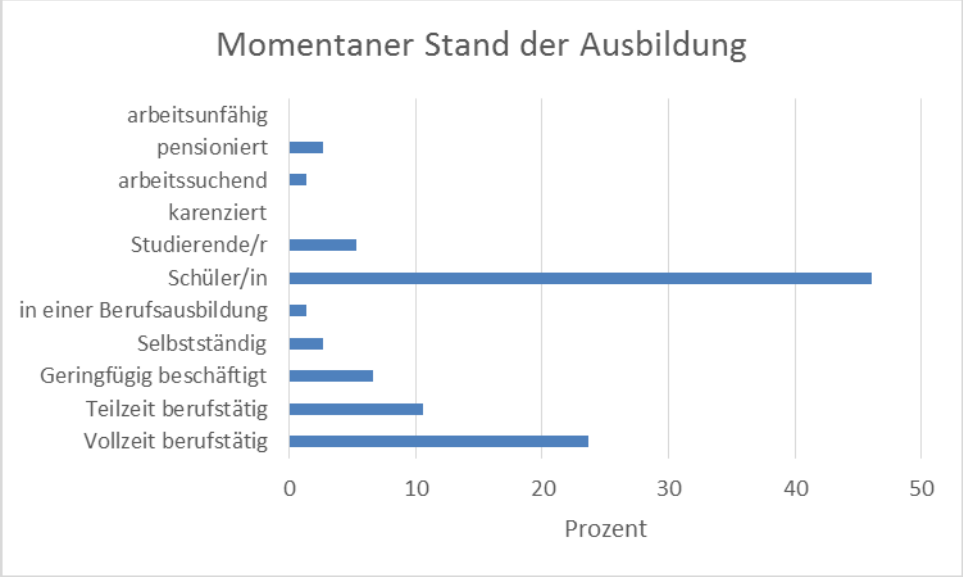


Abbildung 46: Beschäftigung der KursteilnehmerInnen zum Zeitpunkt der Absolvierung

Die Fragen zum Kurs gliederten sich in verschiedene Kategorien (Gründe für die Teilnahme, Bewertung des Kurses, Zufriedenheit, etc…). Die meisten Fragen waren dabei zum Ankreuzen, wobei es auch offene Fragen gab. In den folgenden Tabellen soll eine Übersicht über die Fragen sowie deren Bewertung gegeben werden. Die erste Frage beschäftigte sich mit den Gründen für eine Teilnahme an dem Kurs. In Tabelle 12 sind die Prozente der Antworten zusammengefasst. Neben diesen Gründen wurden von 2 TeilnehmerInnen Interesse an der Spieleprogrammierung genannt, von weiteren TeilnehmerInnen

Spaß an der Programmierung. Als zusätzlicher Grund wurde von einer Person die eigene Zeiteinteilung genannt, eine andere Person erwähnte, dass die ganze Klasse bei dem Kurs mitmachte

Warum haben sie an diesem Online-Kurs teilgenommen?	**1 (trifft voll und ganz zu); 2 (trifft eher zu), 3 (teils teils), 4 (trifft eher nicht zu); 5 (trifft gar nicht zu)**				
	1	**2**	**3**	**4**	**5**
Ich bin am Kursthema generell interessiert.	58	32	10	0	0
Mich interessiert die/der Vortragende.	25	17	25	19	14
Meine FreundInnen/KollegInnen nehmen ebenfalls an diesem Kurs teil.	51	7	7	13	22
Das Thema ergänzt meine Ausbildung.	29	20	23	13	14
Ich habe Interesse an einer Zusatzausbildung.	25	25	25	10	16
Das Thema ergänzt meine derzeitige berufliche Tätigkeit.	26	19	14	12	29
Ich möchte mich beruflich neu orientieren.	16	17	7	16	43
Ich benötige eine Teilnahmebestätigung, die die positive Absolvierung dieses Kurses bestätigt.	45	16	13	6	20
Ich möchte Erfahrungen mit Online-Ausbildungen sammeln.	35	29	30	4	1
Ich habe Interesse an der Gestaltung von Online-Kursen.	29	20	25	17	9
Aufgrund meiner Berufstätigkeit sind Online-Kurse für mich optimal.	29	19	29	13	10
Aufgrund eines körperlichen Handicaps sind Online-Kurse für mich optimal.	12	10	6	7	65
Aufgrund meines Wohnortes sind Online-Kurse für mich optimal.	23	12	17	23	25
Aufgrund meiner Betreuungspflichten sind Online-Kurse für mich optimal.	22	4	16	13	45

Tabelle 14. Antworten in Prozent für die Gründe der Teilnahme an dem Kurs

Die Fragen zur Bewertung des Kurses bezogen sich einerseits auf den Kurs selbst (Tabelle 13), andererseits sollte auch die MOOC-Plattform bewertet werden (Tabelle 14).

Bewertung des Kurses	**1 (sehr gut), 2 (gut), 3 (befriedigend), 4 (genügend), 5 (nicht genügend)**				
	1	2	3	4	5
Grafische Darstellung der Kursinhalte	58	32	9	1	0
Textuelle Darstellung der Kursinhalte	36	46	14	3	0
Navigation durch die Kurseinheiten	46	42	9	3	0
Aufbau und Gliederung der Kurseinheiten	57	35	7	1	0

Zeitlicher Umfang der Kurseinheiten	52	38	10	0	0
Zeitaufwand für die Bearbeitung/Beschäftigung mit den Kursinhalten	48	39	12	1	0
Abstände zwischen den Kurseinheiten	46	35	16	3	0
Auswahl der Lernziele	52	35	12	1	0
Aufbereitung der Lehrinhalte	54	36	10	0	0
Betreuung durch die Kursleitung	51	30	16	3	0
Möglichkeiten zum Austausch im Forum	49	25	26	0	0
Quiz am Ende der Kurseinheiten	49	36	12	3	0
Gesamtbeurteilung des Kurses	59	35	4	1	0

Tabelle 15: Prozentuale Antworten der Bewertung des Online-Kurses

Bewertung der Plattform	**1 (sehr gut), 2 (gut), 3 (befriedigend), 4 (genügend), 5 (nicht genügend)**				
	1	**2**	**3**	**4**	**5**
Aufbau und Gliederung der Plattform	59	38	3	0	0
Navigation in der Kursplattform	49	41	10	0	0
Grafische Darstellung der Plattform	51	41	7	1	0
Textuelle Darstellung der Plattform	46	39	9	6	0
Gesamtbeurteilung der Plattform	52	41	4	0	3

Tabelle 16: Prozentuelle Bewertung der MOOC-Plattform

Bei der Zufriedenheit wurde nach der Zufriedenheit mit dem Kurs gefragt, aber auch nach der eigenen Lernzufriedenheit (Tabelle 15). Hierfür wurde auch erhoben, wie oft die TeilnehmerInnen etwas im Forum posteten und wie viele Stunden sie sich pro Woche mit den Kursinhalten beschäftigten. Des Weiteren wurde gefragt, ob der Kurs Begeisterung für das Thema hervorrief und an wie vielen Online-Kursen bis jetzt teilgenommen wurde.

Zufriedenheit mit:	**1 (sehr zufrieden); 2 (eher zufrieden); 3 (mäßig zufrieden); 4 (eher nicht zufrieden); 5 (gar nicht zufrieden)**				
	1	**2**	**3**	**4**	**5**
...den erhaltenen Informationen zum Ablauf des Kurses	65	33	1	0	0
..dem zur Verfügung gestellten Unterrichtsmaterial	59	30	10	0	0
...der Gliederung der Lerninhalte	62	32	6	0	0

...den Möglichkeiten des Austausches im Diskussionsforum	41	42	16	1	0
...der Betreuung durch die Kursleitung	54	28	16	3	0
...dem persönlichen Lernfortschritt	65	33	1	0	0
… der persönlichen Mitarbeit und Aktivität im Kurs	59	30	10	0	0

Tabelle 17: Prozentuelle Zufriedenheit der KursteilnehmerInnen

Wie beurteilen Sie …	**1 (zu gering); 2 (eher gering); 3 (genau passend); 4 (eher hoch); 5 (zu hoch)**				
	1	**2**	**3**	**4**	**5**
... die Komplexität des Lernstoffes?	54	28	16	3	0
… den Schwierigkeitsgrad des Lernstoffes?	13	22	49	16	0
... den Umfang des Lernstoffes?	13	14	62	9	1

Tabelle 18: Prozentuelle Beurteilung des Lernstoffes

Häufigkeit an Forums-beteiligungen	**Öfter als 10-mal**	**5- bis 10-mal**	**Bis zu 5-mal**	**Nie**
	4	9	28	59
Wie viele Stunden pro Woche Beschäftigung mit Lernstoff?	**Bis zu 1h/W**	**1-3h/W**	**4-5h/W**	**Mehr als 5h/W**
	32	51	16	1
Wie viel würden Sie für ein MOOC-Angebot im selben Ausmaß zahlen?	0 Euro	Bis zu 50 Euro	Bis zu 100 Euro	Über 100 Euro
	62	30	7	0
Kurs weckte Begeiste-rung für Thema	**Trifft voll und ganz zu**	**Trifft zu**	**Teils teils**	**Triff eher nicht zu**	**Triff nicht zu**
	39	42	17	0	1
Teilnahme an wie vielen Online-Kursen bis jetzt?	**Keine Online-Kurse**	**1-3 Online-Kurse**	**Mehr als 3 Online-Kurse**		
	32	52	16		

Tabelle 19: Prozentuelle Werte für Antworten an der Beteiligung

Neben den Fragen mit vorgegebenen Antworten gab es auch offene Fragen über den Kurs. Diese wurden nur teilweise beantwortet und werden nachfol-

gend aufgelistet. Gleiche Antworten werden mit der entsprechenden Anzahl vermerkt.

Welche Themen würden Sie für Ihre nächste MOOC-Teilnahme besonders interessieren?	
Programmieren (3)	Weitere Programmierkurse
Unbedingt ein Fortsetzungskurs, auch Roboter programmieren, weitere praxisbezogene IT-Kurse	Arduino-Programmierung, Raspberry-Programmierung, Java-Programmierkurs
Programmieren; Fotobearbeitung; Sicherheit im Internet	Weitere Möglichkeiten, ein Smartphone oder Tablet kreativ zu nutzen.
Habe alles schon gelernt	Eine Fortsetzung des Kurses!
Einführung in eine Programmiersprache; Webdesign: Tipps und Tricks für Webdesigns	Spieleprogrammierung
Programmierung von Spielen	Vertiefung Pocket Code
Computer-Tricks	Am PC programmieren (2)
E-Learning	c++ oder java
Gamification	Weitere Pocket-Code-Kurse

Tabelle 20: Übersicht über offene Antworten bezüglich Themen für eine nächste MOOC-Teilnahme

Was wünschen Sie sich für ihren nächsten MOOC?	
Eine Hilfe für Technik und Bau von Websiten	So wie bisher...
Dass er genauso praxisorientiert ist.	Dass man sich Tiere usw auch ohne Internet runterladen kann
Dass der Kurs länger dauert.	Scratch (2)
Programmieren	C++ oder Java
Dass er genauso gut gemacht ist und Spaß macht	Nix (7)
Weiß ich nicht (2)	

Tabelle 21: Übersicht über offene Antworten bezüglich Wünsche für einen nächsten MOOC

Das hat mir am besten gefallen.	
Die Steuerung	Die Qualität der individuellen, uneingeschränkten Entfaltungsmöglichkeiten
Die Erklärungen der einzelnen Einheiten; die Stimme...	Dass immer wieder Aufgaben gestellt wurden; dass man Rückmeldungen von der Kursleitung bekam; die Schritt-für-Schritt-Anleitungen - es war einfach super!!!!!
Einfachheit	Die erste Woche
Offene Aufgabenstellung (entweder ein Grundprogramm zu erstellen oder selbst kreativ weiterzuarbeiten)	Die interessanten Aufgabenstellungen und die Möglichkeiten zur Entfaltung der eigenen Kreativität.
Dass man Schritt für Schritt mitmachen konnte	Plattform
Erster Tag	Die Aufgaben und die Quizbearbeitungen
Mir hat am besten gefallen, dass ich mein Wissen im Programmieren erweitert habe	Möglichkeit eines Feedbacks durch den Kursleiter
Beispiele	Das Thema
die leichten Quiz-Aufgaben	Mir hat gefallen, dass das Programmiersystem so leicht aufgebaut ist.

Aufbau	Das Thema
Der ganze Kurs	Die Quiz-Aufgaben
Die gute Kursstruktur	Die erste Woche
Das Lernen und eigenständige Arbeiten	

Tabelle 22:Übersicht über offene Antworten, was am besten gefallen hat.

Gar nicht gefallen hat mir	
Das mit dem Sprechen!	Die letzte Woche
Grammatikfehler	In der fünften Einheit wurde auf einmal die Version von Pocket Code im Kurs geändert.
Sensor funktionierte bei Tablet nicht.	Die lange Wartezeit zwischen den Kursen
Dass ich zu wenig Zeit für das Programmieren hatte.	Dass es keine IOS-Version gibt
Funktioniert nicht auf I-Phone. Konnte das nicht am eigenen Handy machen und testen. Hat sich alles in Gedanken abspielen müssen. Das Forum ist selten schlecht gelöst. Nicht die Betreuung, sondern die Technik dahinter. Funktioniert anders als die üblichen Foren im Internet. Wem ist das bloß eingefallen?	Nix (7)

Tabelle 23: Übersicht über offene Antworten, was gar nicht gefallen hat.

Die letzte Frage des Feedbackformulars beschäftigte sich mit der Frage, wie die TeilnehmerInnen auf den Kurs aufmerksam wurden. Hier konnten neben vorgegebenen Antworten auch andere Quellen genannt werden. Die meisten TeilnehmerInnen (22 Personen) wurden aufgrund persönlicher Empfehlungen durch Freunde/Bekannte/Verwandte/KollegInnen auf den Kurs aufmerksam. Ein Großteil (21 Personen) gab auch an, durch Schule oder LehrerInnen auf den Kurs gekommen zu sein (Abbildung 46):

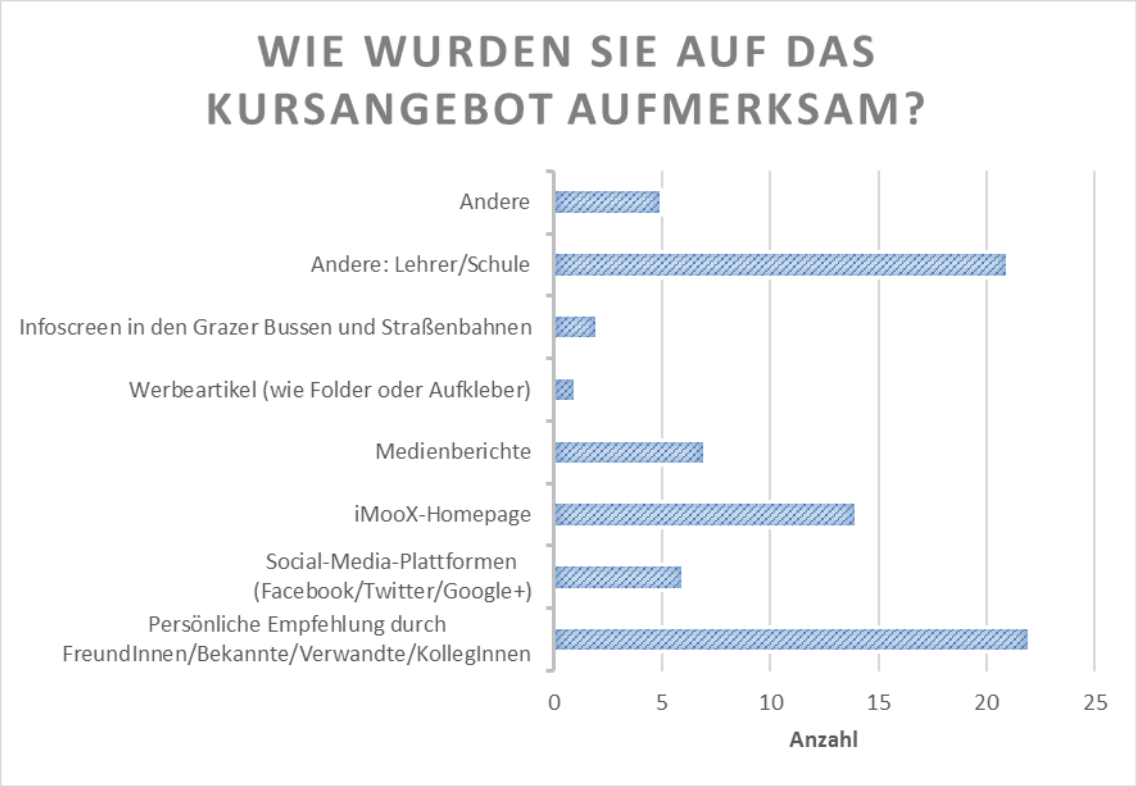


Abbildung 47: Übersicht über die verschiedenen Quellen, wie KursteilnehmerInnen auf den Kurs aufmerksam wurden

5.5.2 Feedback Schul-Setting

Auch im Schul-Setting gab es ein Feedback-Formular. Dieses wurde den SchülerInnen am Ende des Kurses ausgeteilt, nachdem der Pre-Test absolviert worden war. Der Name musste beim Formular nicht angegeben werden, die SchülerInnen konnten dieses Formular anonym ausfüllen, was auch vom Trainer noch betont wurde. Es gab 5 Fragen mit vorgegeben Antworten und drei Fragen mit offenen Antworten. Das Feedback wurde von 14 SchülerInnen ausgefüllt, wovon 2 weiblich waren. In den nachfolgenden Tabellen werden die Ergebnisse und Antworten des Formulars aufgelistet.

Wie hat dir diese Art des Lernens gefallen?			
Ganz schlecht	Geht so	Gut	Sehr gut
0	0	1	13
Wie hat dir diese Art des Programmierens gefallen?			
Ganz schlecht	Geht so	Gut	Sehr gut

0	0	3	11
Waren die Videos verständlich?			
Nein, nicht sehr	Geht so	Ganz ok	Ja, sehr
0	2	3	8
Hast du zuhause Videos angeschaut oder programmiert?			
Ja		Nein	
6		8	
Bist du weiterhin interessiert am Programmieren?			
Nein, nicht sehr	Geht so	Ein bisschen	Ja, sehr
0	0	0	14

Tabelle 24: Übersicht über die Bewertung des Kurses im Schul-Setting, Anzahl der TeilnehmerInnen = 14

Was hat dir am besten gefallen.	
Alles (4)	Das Programmieren (3)
Das man so viele Möglichkeiten hat	Das Spielen anderer Spiele
das Programmieren von eigenen Programmen	Mein Spiel
Dass man selbst Spiele machen kann und Animationen	Viele verschiedene Sachen

Tabelle 25: Übersicht über die Antworten auf die Frage „Was hat dir am besten gefallen?“

Was hast du gelernt?	
Wie man programmiert (6)	Wie man am Tablet programmiert
Wie man am Tablet programmiert	Wie man mit Pocket Code umgeht
was man alles mit Pocket Code machen kann	Wie man Bausteine setzt
Zu viel	Vor Scratch hatte ich keine Ahnung vom Programmieren; was die verschiedenen Bausteine bedeuten und wann man welche benutzt

Tabelle 26: Übersicht über die Antworten auf die Frage „Was hast du gelernt?“

Was hat dir nicht gefallen?	
Nichts (8)	Dass es nach 1h 45min ein bisschen langweilig wird
Nur dass es den Baustein „Stoppe alles“ nicht gegeben hat	Dass es nur so kurz war
Dass ich nach der vorletzten Einheit die Sachen zuhause gemacht habe und bei der vorletzten Einheit herausgefunden habe, dass doch noch die ganze Einheit ist	

Tabelle 27: Übersicht über die Antworten auf die Frage „Was hat dir nicht gefallen?“

6. Diskussion der Ergebnisse

6.1 Videoverhalten

Videos sind meistens die Kernstücke eines MOOCs. Auch in diesem Kurs wurden die Videos zur Inhaltsvermittlung verwendet. Als Hauptparameter zur Analyse der Verwendung der Videos wurden die Aufrufe sowie die durchschnittliche Wiedergabedauer und der durchschnittliche Prozentsatz der Wiedergabe hergenommen („Engagement time“). Hier ist aber zu beachten, dass vor allem im Online-Setting nicht überprüft werden konnte, ob ein Teilnehmer sich aktiv mit einem Video beschäftigte oder dieses nur im Hintergrund laufen ließ, während etwas anderes gemacht wurde (Guo, Kim, & Rubin, 2014). Im Schul-Setting konnte dies aufgrund der Anwesenheit des Kursleiters besser, aber auch nicht vollständig überprüft werden.

Die Analyse dieser „Engagement time“ über die Dauer des gesamten Kurses zeigte bei beiden Settings hohe Werte. Im Online-Setting lag der durchschnittliche Prozentsatz der Wiedergabe der Videos bei allen Kapiteln über 70%. Das bedeutet somit, dass, wenn die Videos angeschaut wurden, diese im Durchschnitt eine lange Abspieldauer hatten (mehr als 2/3 des Videos wurde angeschaut). Auffallend ist, dass es die geringste durchschnittliche Wiedergabedauer im ersten Kapitel gab (73%). Dies kann dadurch erklärt werden, dass es gerade im ersten Kapitel Personen gab, die nur einen Blick in den Kurs warfen und deswegen nicht die gesamte Länge eines Videos abspielten. Zusätzlich gab es nur in diesem Kapitel zu den Videos die schriftlichen Kurzanleitungen, die anstatt der Videos genutzt wurden. In den anderen Kapiteln lag dieser Wert um 80%, im zweiten Kapitel sogar bei 92%.

Im Schul-Setting waren diese Werte noch höher. In Kapitel 2 wurde im Durchschnitt sogar 100% erreicht, was bedeutet, dass die Videos in voller Länge angeschaut wurden. Da bei einzelnen Videos die SchülerInnen das Video stoppten und Teile noch einmal anschauten, lag dieser Wert bei manchen Videos sogar über 100%. Den geringsten Wert gab es, wie auch im Online-Setting, im 1. Kapitel. Dies lässt sich dadurch erklären, dass es auch hier neben den

Videos die schriftlichen Kurzanleitungen gab, die anstatt der Videos verwendet werden konnten.

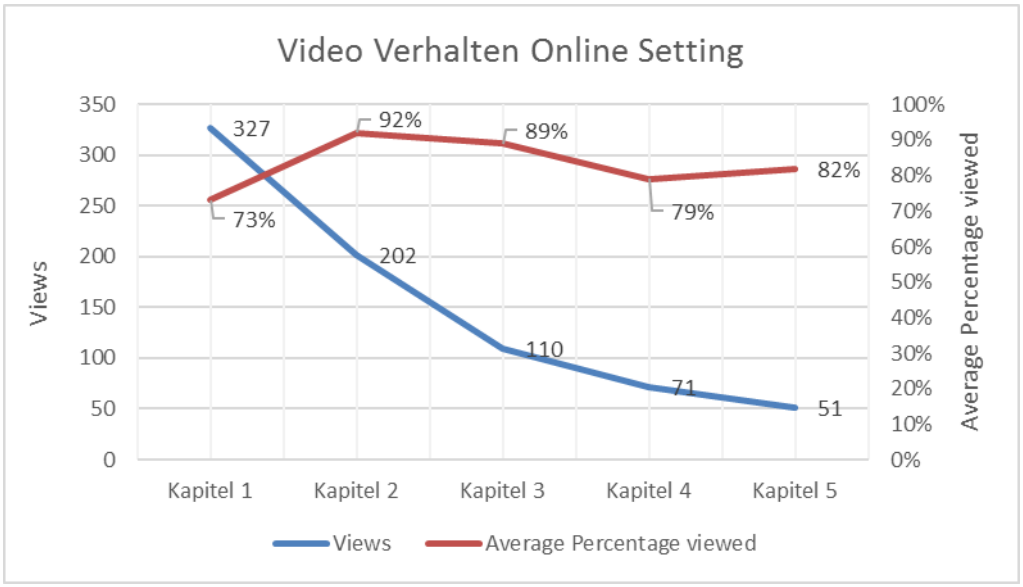


Abbildung 48: Video-Verhalten (Aufrufe und durchschnittliche Prozentsatz der Wiedergabe) der TeilnehmerInnen im Online-Setting

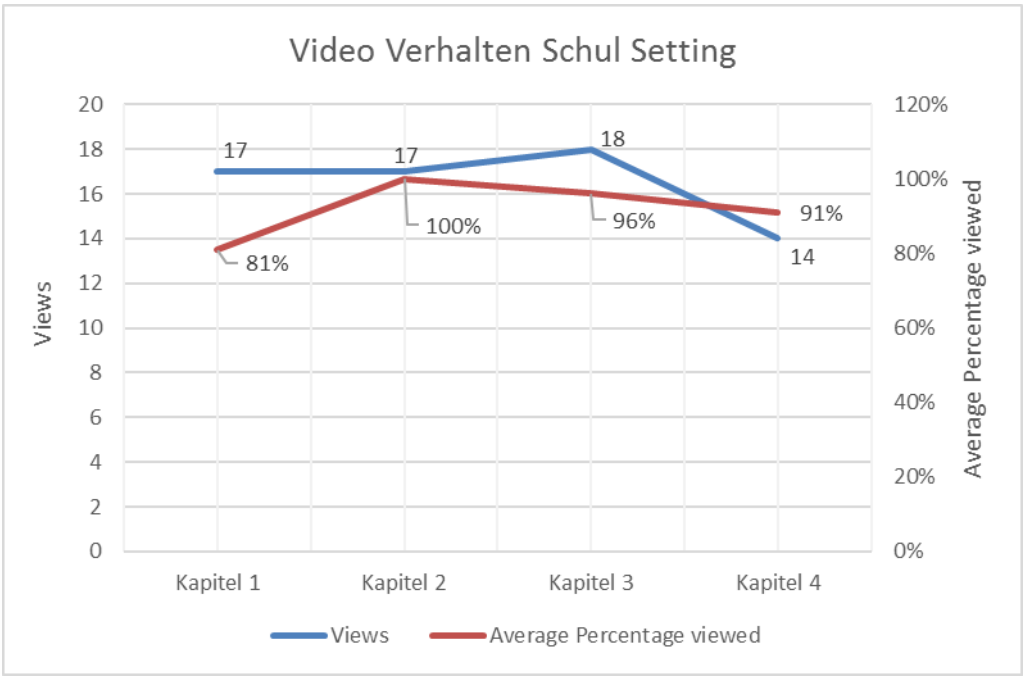


Abbildung 49: Video-Verhalten (Aufrufe und durchschnittliche Prozentsatz der Wiedergabe) der Teilnehmerinnen im Schul-Setting

Einen sehr großen Unterschied zwischen den beiden Settings gab es bei der Analyse der Video-Aufrufe. Beim Online-Setting zeigte sich über die Dauer des

Kurses ein stetiger und deutlicher Abfall der Aufrufe, während im Schul-Setting diese Zahl, mit Ausnahme des letzten Kapitels, immer gleich blieb. Im ersten Kapitel gab es im Durchschnitt 327 Aufrufe pro Video, im letzten Kapitel fiel diese Zahl auf 51 Aufrufe (Abbildung 47). Allein durch diese Analyse der Videoaufrufe lässt sich eine starke Dropout-Rate (84%) über die Dauer des Kurses erkennen. Diese Entwicklung deckt sich mit Ergebnissen von Studien, die generell eine hohe Dropout-Rate von MOOCs feststellten (Khalil & Ebner, 2014) (Jordan, 2015).

Die Gründe für eine solche Dropout-Rate können laut Khalil und Ebner (2014) sehr verschieden sein. Hier werden Gründe, wie zum Beispiel Zeitmangel, Motivation, Gefühl der Isolation, mangelndes Hintergrundwissen oder auch versteckte Kosten, genannt.

Bei der Durchführung des Kurses in der Schule als sogenanntes Inverse-Blended-Learning-Konzept sollten einige dieser Gründe wie zum Bespiel Gefühl der Isolation, Zeitmangel oder auch Motivationsmangel verhindert werden. Da der Kurs als Unterrichtseinheit durchgeführt wurde, gab es hier keine Dropout-Rate. Keine Dropout-Rate muss in diesem Setting aber nicht unbedingt bedeuten, dass die SchülerInnen sich auch tatsächlich mit dem Inhalt beschäftigten. Obwohl der Kursleiter vor Ort war, konnte nicht immer kontrolliert werden, ob sich die SchülerInnen tatsächlich mit dem Stoff beschäftigten und wie intensiv diese Beschäftigung war. Ein Blick auf die Aufrufe lässt erkennen, wie oft die Videos angeschaut worden sind. Da die Zahl der Aufrufe in jedem Kapitel mindestens gleich hoch war wie die Anzahl der SchülerInnen, lässt sich die Vermutung aufstellen, dass jede/r der SchülerInnen die Videos anschaute. In den Kapiteln 1 bis 3 gab es mehr als 14 Aufrufe, was dadurch zustande kommen kann, dass sich manche SchülerInnen ein Video noch einmal anschauten. Zusätzlich ist der, wie schon angesprochen, sehr hohe durchschnittliche Prozentsatz der Wiedergabe der Videos ein Indiz für eine wirkliche Beschäftigung mit den Videos. Zusammengefasst lässt sich feststellen, dass die Videos sowohl im Online-Setting als auch im Schul-Setting eine lange Wiedergabedauer hatten. Im Online-Setting war auffallend, dass die Videos von Kapitel zu Kapitel weniger angeschaut wurden, die Länge der Aufrufe aber ziemlich gleich blieb. Bei der Durchführung in der Schule zeigte sich ein wesentlicher Vorteil, der für MOOCs als Inhaltsvermittlung sprechen könnte. Die SchülerInnen konnten in ihrem ei-

genen Tempo lernen. Manche SchülerInnen schauten sich ein Video noch einmal an und Videos wurden gestoppt, um Teile davon zu wiederholen.

6.2 Bearbeitung der Quiz-Aufgaben

Neben den Videos zeigte auch die Analyse der Quiz-Aufgaben, welche Beteiligung es an dem Kurs gab. Im Online-Setting absolvierten 160 von 571 Teilnehmerinnen zumindest ein Quiz. Auch hier gab es über die Dauer des Kurses einen stetigen Rückgang an Bearbeitungen. Absolvierten das Quiz in Kapitel 1 noch 151 TeilnehmerInnen, wurde das Quiz vom letzten Kapitel nur noch 71-mal beantwortet. Die Dropout-Rate, die schon bei den Videos beobachtet werden konnte, zeigt sich somit auch hier, wenn auch nicht so ausgeprägt. Wird allerdings das Kapitel, in dem ein Quiz absolviert wurde, weglassen und nur ein Blick darauf geworfen, wie viele Personen wie viele Quiz-Aufgaben abgeschlossen haben, zeigt sich ein anderes Bild. Von den 160 Personen, die zumindest 1 Quiz absolvierten, waren es 45 TeilnehmerInnen, die genau ein Quiz beantworteten, hingegen 70, die alle fünf Quizze abschlossen. Hier könnte sich ein Motivationsaspekt herauslesen lassen. Waren die TeilnehmerInnen bereit ein Quiz zu absolvieren, dann war in gewisser Weise schon die Motivation vorhanden, den Kurs abzuschließen. Nimmt man nur die 160 Personen her, die sich überhaupt mit den Quiz-Aufgaben beschäftigten, und vergleicht man diese mit der Anzahl, die den Kurs wirklich beendeten (64), dann ergibt sich eine Dropout-Rate von 60%. Diese Dropout-Rate ist wesentlich geringer als die tatsächliche, die sich aus der Anzahl der Anmeldungen zum Kurs (571) und der Abschlüsse ergibt (64) und 89% betrug.

Im Schul-Setting zeigte sich ein anderes Bild. Da die Unterrichtseinheiten zeitlich eingeschränkt waren, wurden die Quiz-Aufgaben nicht priorisiert Sie wurden vom Kursleiter erwähnt, aber nachdem die Videos angeschaut worden waren, lag die Motivation der SchülerInnen eher beim Selber-Ausprobieren und Üben. Das spiegelt sich auch in der Beteiligung der Quiz wider. In der ersten Einheit wurde das Quiz noch 7-mal absolviert, in der vierten Einheit 2-mal. Es gab insgesamt nur 2 SchülerInnen, die alle Quiz-Aufgaben absolviert haben. In diesem Setting spiegelt die Beteiligung an den Quiz-Aufgaben mehr eine Umkehrung der Motivation wider. Je mehr die SchülerInnen lernten (je mehr der

Kurs vorangeschritten war), desto weniger Quiz-Aufgaben wurden absolviert. Es wurde vom Kursleiter wahrgenommen, dass aufgrund der zeitlichen Einschränkung die SchülerInnen die Zeit nutzen wollten, um die Aufgaben zu erfüllen oder auch eigene Programme zu entwickeln. Ganz am Ende des Kurses gab es jedoch ein verpflichtendes „Quiz“. Dieses unterschied sich allerdings von den anderen Quiz Aufgaben, da es in schriftlicher Form abgehalten wurde, den gesamten Stoff des Kurses umfasste und in Pseudo-Code geschrieben war. Es sollte vor allem als Rückmeldung für den Kursleiter dienen, um zu sehen, ob die Konzepte verstanden wurden und um herauszufinden, ob informatische Bildung stattgefunden hatte. Die Aufgaben können als schwieriger als die Quiz Aufgaben beschrieben werden, da anders als bei den Fragestellungen im Kurs, Probleme in einer neuen Art und Weise dargestellt worden sind (Pseudocode). Die Auswertung zeigte eine richtige Beantwortung aller Fragen von mindestens der Hälfte der SchülerInnen mit Ausnahme von Frage 4. Frage 4 beinhaltete die Kombination von mehreren Konzepten gleichzeitig und konnte nur von 5 SchülerInnen richtig beantwortet werden. Dieses schlechtere Abschneiden gegenüber den anderen Fragen ist dadurch erklärbar, dass diese Kombination von Konzepten (Schleife mit 2 Variablen und einer Bedingung) beim Programmieren nicht gebraucht wurde und deswegen gänzlich neu war für die SchülerInnen. Es gab insgesamt 3 SchülerInnen, die alle Fragen richtig beantworten konnten. Interessanterweise waren 2 von diesen 3 SchülerInnen Schüler, die auch alle 4 Quiz-Vorlagen des Kurses positiv absolviert hatten.

Zusammengefasst lässt sich für das Online-Setting sagen, dass die Quiz-Bearbeitungen ein Indikator dafür sein könnten, wie intensiv man sich mit dem Kurs beschäftigt. So ergibt sich unter den TeilnehmerInnen, die sich mit mindestens einem Quiz beschäftigt hatten, eine wesentlich niedrigere Dropout-Rate als unter allen angemeldeten TeilnehmerInnen. Im Schul-Setting stellt sich die Frage, ob es unter dem zeitlichen Aspekt Sinn macht, die Quiz-Aufgaben anzubieten. Die Teilnahme an den Quiz wurden zwar von Einheit zu Einheit geringer, aber am Pre-Test am Ende zeigte sich, dass die SchülerInnen, die sich mit allen Quiz-Aufgaben auseinandergesetzt hatten, sehr gut abschnitten. Der Pre-TestDie Quiz sollten somit beibehalten werden als Möglichkeit, sich mit dem Inhalt auf einer anderen Ebene als dem eigenen Üben auseinanderzusetzen. Lerntypen und Motivation sind unterschiedlich, deswegen macht es Sinn, ver-

schiedene Möglichkeiten zu Wissensüberprüfung und -anwendung anzubieten. Zusätzlich zeigte der Pre-Test, dass die SchülerInnen durchaus in der Lage waren, die Konzepte an Pseudocode anzuwenden. Letztendlich taten sich natürlich manche SchülerInnen leichter als andere; im Allgemeinen konnte aber informatische Bildung auch in der Hinsicht vermittelt werden, dass den SchülerInnen zumindest Wörter wie Schleifen, Koordinatensystem, Variablen, Objekte oder Bedingung ein Begriff wurden.

6.3 Beteiligung im Forum

Ein wesentlicher Nachteil von MOOCs, der auch oftmals ein Grund für Dropout sein kann, ist die fehlende Interaktion zwischen den TeilnehmerInnen und den KursleiterInnen. Das Forum im Online-Setting hatte den Zweck, eine Anlaufstelle bei Fragen zu sein sowie den TeilnehmerInnen einen Austausch untereinander zu ermöglichen. Im Schul-Setting wurde das Forum nicht genutzt und gebraucht, die Interaktion fand hier direkt in der Klasse statt.

Die meisten Forumsbeiträge wurden in der ersten Woche gepostet. Dies ist nicht verwunderlich, da es in Woche 1 einen Vorstellungsthread gab sowie vermehrt Fragen zum Ablauf bzw. zu technischen Problemen auftraten. In den darauffolgenden Wochen sank diese Zahl auf 15-17 Beiträge mit der Ausnahme von Woche 3, in der es einen Anstieg auf 23 Beiträge gab. Einen ähnlichen Verlauf gab es für die gelesenen Forumsbeiträge, in der ersten Woche wurden die Forumsbeiträge 960-mal angeklickt, in der Woche darauf nur mehr 469-mal. Auffallend war, dass es in der letzten Kurswoche einen deutlichen Anstieg gab und es zeigte sich, dass das Forum vor allem nach Ende des Kurses noch genutzt wurde, um Beiträge zu lesen.

In genereller Hinsicht lässt sich durch diese Analyse ein gewisses Verhalten der TeilnehmerInnen ableiten. Die meisten TeilnehmerInnen bevorzugten das Forum nicht aktiv, sondern passiv. Im Vergleich zu der TeilnehmerInnen-Zahl wurden sehr wenige Beiträge verfasst, die aber relativ oft gelesen bzw. angeklickt wurden. Die Mehrzahl der TeilnehmerInnen war interessiert daran, was ihre MitkollegInnen posteten, wollten sich aber nicht äußern. Daraus lässt sich ableiten, dass das Forum eine wirkliche Interaktion nicht ersetzen konnte.

6.4 Lernverhalten Begleitmaterial

Die Auswertung der Aufrufe des Begleitmaterials sollte als weitere Möglichkeit dienen, neben den Videos das Lernverhalten der TeilnehmerInnen im Online-Setting zu bestimmen. Anders als bei den Videoaufrufen zeigte sich hier kein stetiger Abfall. In den ersten Wochen gab es mit 633 die meisten Aufrufe, diese Zahl fiel in der Woche 2 auf 438, in Woche 3 auf 428, in Woche 4 auf 348, um letztendlich in Woche 5 wieder auf 504 Aufrufe zu steigen. Damit ergibt sich mit 20% eine wesentlich geringere Dropout-Rate als bei den Videos. Es scheint, dass die Begleitmaterialien nur von den TeilnehmerInnen aufgerufen wurden, die auch vorhatten, den Kurs abzuschließen. Die Art des Begleitmaterials spielte auch eine Rolle bei den Aufrufen. Die schriftlichen Kurzanleitungen in Woche 1 wurden insgesamt am öftesten aufgerufen, gefolgt von den Pocket-Karten. Die Materialien zu den Aufgaben wurden ähnlich oft aufgerufen, die schriftliche Anleitung und die Lösung sogar fast gleich oft (1 Aufruf Unterschied). Dieses häufige Aufrufen der Lösung der Aufgaben könnte erklären, warum sehr wenige Personen ihre Programme im Forum posteten (siehe Kapitel 6.5). Die Aufgabe konnte selbstständig kontrolliert werden und bedarf deswegen keines Feedbacks vom Kursleiter.
Generell zeigte sich, dass die Begleitmaterialien durchaus von den TeilnehmerInnen benutzt wurden und als wichtiger Bestandteil des Kurses gesehen werden können.

6.5 Eigene erstellte Programme

Die Überprüfung der eigenen erstellten Programme sollte dazu dienen zu sehen, wie die gestellten Aufgaben gelöst wurden. Gerade im Schul-Setting sollte dadurch Einsicht gewonnen werden, inwieweit die SchülerInnen das Gelernte umsetzen konnten. Auf der anderen Seite sollten die TeilnehmerInnen durch das Kreieren von eigenen Programmen die Möglichkeit erhalten zu üben. Üben und einfach ausprobieren spielt eine sehr wichtige Rolle im Lernprozess des Programmierens.
Das Angebot, eigene Programme abzugeben, um darauf Feedback zu bekommen, wurde im Online-Setting nicht sehr gut angenommen. Im ersten Kapitel

posteten nur 7 TeilnehmerInnen ihre Programme, im letzten Kapitel nur mehr 3. Des Weiteren waren die TeilnehmerInnen, die ihre Programme posteten, immer die gleichen User. Es stellt sich nun die Frage, warum dieses Angebot so schlecht angenommen wurde. Das könnte damit zusammenhängen, dass es Lösungen gab, mit denen die eigenen Programme überprüft werden konnten und somit das Feedback des Kursleiters nicht gebraucht wurde. Auf der anderen Seite könnte auch eine gewisse Hemmschwelle existieren, die Programme mit den anderen KursteilnehmerInnen zu teilen. Die Programme, die abgegeben wurden, erfüllten die Angaben und auch die Abschluss-Programme zeigten, dass die TeilnehmerInnen in der Lage waren, die Konzepte einzusetzen. Eigene Ideen konnten umgesetzt werden, was sich in der Variation dieser Programme zeigte.

Im Schul-Setting wurden weit mehr Programme abgegeben. Die SchülerInnen waren sehr motiviert, die Aufgaben zu lösen und mit Pocket Code zu experimentieren. Obwohl nicht in jeder Einheit von jeden ein Programm hochgeladen wurde, waren alle SchülerInnen am Programmieren und Üben. Meistens wurden die Aufgaben programmiert, wobei auch eigene Ideen ausprobiert wurden. Am Ende der letzten Einheit präsentierte jeder Schüler- und jede Schülerin den MitschülerInnen sein/ihr Abschluss-Programm; von 8 SchülerInnen wurde das Programm auch hochgeladen. Unter den Abschluss-Programmen zeigten sich unterschiedlichste Programme, die teilweise schon sehr komplex waren und viele verschiedene Konzepte beinhalteten. Ein Programm wurde von 3 Schülern im Team programmiert und eine Schülerin programmierte sogar zuhause zusammen mit ihrer Mutter.

Abbildung 50: Arbeit im Team

Fast alle dieser Programme waren Spiele, was sich mit der Lebenswelt der SchülerInnen deckt. Spiele mit dem Smartphone zu spielen ist gerade in der jetzigen Generation der Kinder und Jugendlichen ein sehr beliebter Zeitvertreib (Schippers, & Mak, 2015). Die Aussicht, ein eigenes Produkt (in Form eines Programms) in den Händen zu halten, erwies sich als weiterer Ansporn. Auch die Möglichkeit zu sehen, wie oft ein hochgeladenes Programm von anderen Personen heruntergeladen wurde, wirkte motivierend.
Im Großen und Ganzen zeigten die abgegebenen Programme im Schul-Setting, dass informatische Bildung vermittelt werden konnte.

6.6 Feedback der TeilnehmerInnen

Die Motivation spielt eine wesentliche Rolle, an einem MOOC teilzunehmen und diesen auch abzuschließen. Laut Belanger und Thornton (2013) kann die Motivation eines MOOC-Teilnehmers in vier Kategorien eingeteilt werden:

- Die Motivation, sich fortzubilden („Lebenslanges Lernen“) oder ein besseres Verständnis für die Materie zu bekommen, ohne dabei bestimmte Erwartungen an den erfolgreichen Abschluss zu haben
- Um Spaß, Unterhaltung, soziale Erfahrung und geistige Stimulation zu erleben.

- Aus Bequemlichkeit, oft in Verbindung mit Zugangsbeschränkungen zu traditionellen Bildungsoptionen
- Aus Neugier, um Einblicke in „Online Education“ zu bekommen

Die Analyse des Feedbacks soll über die Beweggründe, Motivation und Zufriedenheit der TeilnehmerInnen Aufschluss geben. Im Online-Setting gab es dazu ein Feedbackformular, das allerdings nur von den Personen ausgefüllt wurde, die den Kurs auch beendeten. Somit lassen sich die Beweggründe von den Personen, die den Kurs nicht zur Gänze absolvierten, nicht feststellen. Zusätzlich zum Feedback wurden auch demografische Daten erhoben. Hier stellte sich heraus, dass die meisten Teilnehmer SchülerInnen (46%) im Alter von 11 -15 Jahren (43%) waren. Von den gesamten TeilnehmerInnen war fast die Hälfte weiblich (46%), was als sehr erfreulich angesehen werden kann, da die sogenannten MINT(Mathematik, Informatik, Naturwissenschaft und Technik)-Fächer überwiegend männlich besetzt sind (Rigby, 2015). Geografisch gesehen kamen die meisten Teilnehmerinnen aus der Steiermark, gefolgt von der deutschsprachigen EU.

Die Fragen im Feedback-Formular reichten von der Zufriedenheit über die verschiedenen Kurskomponenten bis hin zu Fragen, warum der Kurs absolviert wurde. Es gab Fragen zum Ankreuzen und auch offene Fragen.

Bei der Frage nach dem „Warum haben Sie an diesem Online-Kurs teilgenommen?“ war das generelle Interesse am Kursthema der wichtigste Grund. Das soziale Umfeld spielte auch eine wichtige Rolle, für 51% war die Teilnahme von FreundInnen/KollegInnen ein wichtiger Grund, selbst teilzunehmen. Der drittstärkste Grund war die Benötigung einer Teilnahmebestätigung für die positive Absolvierung des Kurses. Wie wichtig das persönliche Umfeld ist, zeigte sich in der Frage, wie die TeilnehmerInnen überhaupt auf das Kursangebot aufmerksam geworden waren. 22 Personen (32%) wurden durch persönliche Empfehlung durch Freundinnen/Freunde/Bekannte/Verwandte/KollegInnen auf den Kurs aufmerksam und 21 Personen (30%) gaben an, durch LehrerInnen bzw. in der Schule von dem Kurs gehört zu haben. Die TeilnehmerInnen, die den Kurs zu Gänze absolvierten, bewerteten diesen auch in allen Punkten als sehr gut oder gut. Als Gesamtbeurteilung wurde zu 59% die Note „Sehr gut“ und zu 35% die Note „Gut“ vergeben. Auch bei Fragen zur Zufriedenheit mit dem Kurs wurden alle Punkte zu mehr als 80% mit „Sehr zufrieden“ oder „Eher zufrieden“

bewertet. Daraus lässt sich schlussfolgern, dass verständlicherweise die TeilnehmerInnen, die den Kurs abschlossen, auch zufrieden damit waren. Inwieweit Zufriedenheit allerdings in Zusammenhang mit der Absolvierung des Kurses steht, lässt sich nicht klar beantworten, da kein Feedback von Personen erhoben wurde, die den Kurs vorzeitig beendeten. Hier könnten auch andere bekannte Gründe wie Zeitmangel, Interesse oder auch ein Gefühl der Isolation eine Rolle spielen. 51% der TeilnehmerInnen beschäftigten sich 1-3 Stunden pro Woche mit dem Lernstoff und 32% weniger als 1 Stunde. Die Wichtigkeit eines kostenlosen Angebotes wurde von 62% der TeilnehmerInnen unterstrichen, die angaben, 0 Euro für ein MOOC im selben Ausmaß zu zahlen. 30 % würden bis zu 50 Euro ausgeben und nur 7% bis zu 100 Euro. Dass Programmieren durchaus interessant und spannend sein kein, zeigte die Frage, ob der Kurs die Begeisterung für das Thema weckte. Hier gaben 39% „Trifft voll und ganz zu“ und immerhin 42% „Trifft zu“ an. Auch bei der offenen Frage nach dem Interesse der Themen für eine nächste MOOC-Teilnahme wurde als Mehrzahl das Thema Programmieren genannt. Das Feedback der TeilnehmerInnen im Online-Setting zeigte sich insgesamt als sehr positiv.

Im Schul-Setting bestand der Fragebogen aus sehr einfach gestellten Fragen und auch hier war das Feedback sehr positiv. Im eigenen Tempo zu lernen wurde von den SchülerInnen sehr gut angenommen, lediglich einer Schülerin gefiel diese Art zu lernen nur als „Gut“, ansonsten wurde hier „Sehr gut“ angegeben. Die Art, mit Pocket Code zu programmieren, gefiel 11 SchülerInnen „Sehr gut“ und drei „Gut“. Das Baustein-System, das sich schon bei Scratch bewährt hat, zeigte auch auf mobilen Geräten großen Anklang. Die Videos waren für acht SchülerInnen sehr verständlich, für drei „Ganz ok“ und für zwei „Geht so“.

Wie interessiert die SchülerInnen am Programmieren waren, unterstrich die Frage, ob zuhause Videos angeschaut wurden oder ob weiterprogrammiert wurde. Obwohl es keine Hausübungen gab, gaben immerhin sechs SchülerInnen an, sich zuhause mit dem Kurs beschäftigt zu haben. Dass Programmieren durchaus auch für Kinder interessant ist und Spaß machen kann, bestätigte die letzte Frage, ob weiterhin Interesse am Programmieren bestehe. Sie wurde von allen SchülerInnen mit „Ja, sehr“ beantwortet. Sowohl was Interesse als auch

Motivation anbelangt, zeigte die Durchführung des MOOC in der Schule sehr gute Ergebnisse.

7. Zusammenfassung und Ausblick

Ist es möglich, mithilfe von MOOCs informatische Bildung zu vermitteln? Der Kurs „Learning to Code - Programmieren mit Pocket Code“, der auf der MOOC-Plattform iMooX durchgeführt wurde, bestätigte, dass es möglich ist, auf diese Art Programmierkompetenzen und Computational Thinking zu fördern.

In der Durchführung als reiner Online-Kurs zeigte sich in der Analyse der Videos, des Begleitmaterials und der Quiz-Bearbeitungen das für MOOCs bekannte Problem der hohen Dropout-Rate. Von den 571 angemeldeten Personen absolvierten nur 69 Personen den gesamten Kurs, was einer Dropout-Rate von 89% entspricht. Nichtsdestotrotz waren die TeilnehmerInnen, die den Kurs zur Gänze absolvierten, sehr zufrieden mit den Inhalten und dem Thema. Die abgegebenen Programme der KursteilnehmerInnen zeigten, dass Programmierkonzepte auf Problemstellungen angewendet werden konnten. Der Kurs weckte bei den meisten eine Begeisterung für das Thema. Ein wichtiger Faktor für die Teilnahme und Absolvierung zeigte sich im Umfeld der TeilnehmerInnen. Mehr als die Hälfte der TeilnehmerInnen, die den Kurs abschlossen, wurden aufgrund persönlicher Empfehlung durch Freundinnen/Freunde/Bekannte/Verwandte/KollegInnen oder LehrerInnen/Schule auf das Kursangebot aufmerksam. Hieraus könnte man eine Voraussetzung für MOOCs als Bildungsvermittlung ablesen. Gerade im Kinder- und Jugendalter scheint es ganz wenige zu geben, die einen MOOC von ganz allein anfangen. Die Schule bzw. LehrerInnen oder auch Eltern spielen hier wahrscheinlich eine große Rolle, in der Hinsicht, dass sich Kinder oder Jugendliche überhaupt mit einem MOOC beschäftigen.

Generell könnten MOOCs vor allem für den Schulbereich eingesetzt werden. So zeigte die Durchführung des MOOCs im Schul-Setting eine wahre Begeisterung der SchülerInnen für das Thema. Im Vergleich zur Durchführung als reiner MOOC konnten sich hier die SchülerInnen gegenseitig austauschen und auch zusammenarbeiten. Durch die Anwesenheit des Trainers war es den SchülerInnen zusätzlich möglich, ein Live-Feedback zu bekommen. Gerade bei MOOCs scheint sich hier das Inverse-Blended-Learning-Konzept als Lernmodell durchzusetzen (Käfmüller, 2016). Obwohl die Quiz-Aufgaben sehr wenig bearbeitet

wurden, gab es in anderen Bereichen eine große Beteiligung. Besonders in den eigenen entwickelten Programmen zeigte sich, dass die SchülerInnen in der Lage waren, die Kursinhalte auf eigene Problemstellungen anzuwenden. Das Baustein-Prinzip von Pocket Code erwies sich als ideal, um Kindern erste Programmiererfahrungen auf mobilen Geräten zu ermöglichen. Zusätzlich erlaubte der Aufbau als MOOC den SchülerInnen im eigenen Tempo zu lernen. So deckte die Analyse des Videoverhaltens auf, dass einige Schüler gesamte Videos oder Teile eines Videos öfters anschauten. Was auch für den Erfolg des Kurses nicht vernachlässigt werden sollte, ist das Konzept des sogenannten „Making“. Die SchülerInnen hatten zwar zuerst die Vorgabe, die Videos anzuschauen, waren dann aber frei, mit ihrem Tablet selbst zu programmieren. Das Augenmerk wurde auf das eigene Tun und Machen gerichtet. Ob das nun die Aufgaben oder eigene Programme waren, war ihnen selbst überlassen. Der Ausblick, am Ende ein eigenes Programm in den Händen zu halten, war für viele SchülerInnen ein weiterer Ansporn (Ebner & Schön, 2017).

Durch diese Arbeit ließ sich feststellen, dass MOOCs nicht nur geeignet sind, informatische Bildung zu vermitteln, sondern laut Feedback der TeilnehmerInnen auch in der Lage sind, Interesse für ein Thema zu wecken. Gerade Programmieren ist in den Lehrplänen der Sekundarstufe kaum verankert und deswegen für Kinder und Jugendliche oft nicht durchschaubar. Hier zeigte sich, dass ein entsprechender MOOC durchaus eingesetzt werden kann. Allerdings sollte der MOOC nicht als klassischer MOOC abgehalten werden, sondern mithilfe des didaktischen Konzepts „Inverse Blended Learning“. Die Inhalte müssen an die entsprechende Zielgruppe angepasst sein und sollten das sogenannte „Making“ miteinbeziehen. Für die ersten Schritte in der Programmierung eignete sich Pocket Code sehr gut, da aufgrund der einfachen visuellen Benutzeroberfläche typische Probleme wegfielen und sich die TeilnehmerInnen auf die Umsetzung ihrer Ideen konzentrieren konnten. Besonders wichtig sind auch der Austausch und die Interaktion der TeilnehmerInnen untereinander sowie die Anwesenheit eines Trainers. Im Schulbereich könnten die LehrerInnen diese Aufgabe übernehmen. Für die Zukunft wird es deswegen wichtig sein, MOOCs so anzubieten, dass sie von LehrerInnen in der Schule hergenommen werden können. Die Materialien müssen adaptierbar sein, sodass auch nur Teile des MOOCs im Unterricht verwendbar sind.

Inwieweit sich MOOCs im Bildungsbereich und hier besonders in der Vermittlung von informatischen Kompetenzen tatsächlich durchsetzen, bleibt abzuwarten. Fakt ist aber, dass MOOCs mit entsprechender Aufbereitung und Durchführung eine Abwechslung und Alternative sein können, um zusätzliche Lehrinhalte anzubieten. Diese Lerninhalte können einer breiten Masse zur Verfügung gestellt werden und so neue Chancen ermöglichen.

Literaturverzeichnis

Bauer, M., & Waba, S. (2016). eDucation - Digitale Bildung für alle. *Schule Aktiv, Sonderheft Oktober 2016*, S. 43-46.

Belanger, Y., & Thornton, J. (2013). Bioelectricity: A Quantitative Approach, Duke University's First MOOC. Abgerufen am 03. 06 2017 von http://dukespace.lib.duke.edu/dspace/handle/10161/6216

Boxser, M., & Agarwal, A. (2014). Why MOOCS Might Be Just Right for Schools. *Huffpost Education*. Abgerufen am 7. 1. 2016 von http://www.huffingtonpost.com/marc-boxser/why-moocs-might-be-just-r_b_5365503.html

Breslow, L., Pritchard, D. E., DeBoer, J., Stump, G. S., Ho, A. D., & Seaton, D. T. (2013). Studying learning in the worldwide classroom: Research into edX's first MOOC. *Research and Practice in Assesment 8 (Summer 2013)*. Abgerufen am 8. 1. 2017 von http://www.rpajournal.com/dev/wp-content/uploads/2013/05/SF2.pdf

Bundesministerium für Bildung. (2017b). *www.bmb.gv.at.* Abgerufen am 3. Januar 2017 von https://www.bmb.gv.at/schulen/unterricht/lp/lp_neu_ahs_14_11866.pdf?5i84ki

Bundesministerium für Bildung. (2017c). Abgerufen am 3. Januar 2017 von https://www.bmb.gv.at/schulen/unterricht/lp/lp_abs.html

Bundesministerum für Bildung. (3. Januar 2017a). *www.bmb.gv.at.* Abgerufen am 3. Januar 2017 von https://www.bmb.gv.at/schulen/unterricht/lp/lp_neu_ahs_21_11876.pdf?5i84lk

Dreisibner, S., Ebner, M., & Kopp, M. (2014). Kosten und Wert von MOOCs am Beispiel der Plattform iMooX. In T. Köhler, & N. Kahnwald, *Workshop GeNeMe'14, Gemeinschaft in Neuen Medien: Virtual Enterprises, Research Communities & Social Media Networks* (S. 191-204). Dresden: TUDpress.

Ebner, M., Schön, S., & Käfmuller, K. (2015). Inverse Blended Learning bei „Gratis Online Lernen" – über den Versuch, einen Online-Kurs für viele in die Lebenswelt von EinsteigerInnen zu integrieren. In N. Nistor , & S.

Schirlitz, *Digitale Medien und Interdisziplinarität* (Bd. 68, S. 197-206). Waxmann.

Ebner, M., & Schön, S. (2017). Die Maker-Bewegung macht Schule: Hintergründe, Beispiele sowie erste Erfahrungen. In J. Erpenbeck, & W. Sauter, *Handbuch Kompetenzentwicklung im Netz* (S. 257-270). Stuttgart: Schäffer-Poeschel Verlag.

Europäische Union. (2006). *EMPFEHLUNG DES EUROPÄISCHEN PARLAMENTS UND DES RATES vom 18. Dezember 2006 zu Schlüsselkompetenzen für lebensbegleitendes Lernen.* Brüssel: Amtsblatt der Europäischen Union. Abgerufen am 6. 1. 2017 von http://eur-lex.europa.eu/legal-content/DE/TXT/PDF/?uri=CELEX:32006H0962&from=DE

Feierabend, S., Plankenhorn, T., & Rathgeb, T. (2015). *Jugend, Information, (Multi-) Media. Basisstudie zum Medienumgang 12- bis 19- Jähriger in Deutschland.* Stuttgart: Medienpädagogischer Forschungsverbund Südwest. Abgerufen am 5. 1. 2017 von https://www.mpfs.de/fileadmin/files/Studien/JIM/2015/JIM_Studie_2015.pdf

Filzmoser, G. (2016). Wie wollen wir es nennen: Computerkompetenz, Medienkompetenz oder digitale Kompetenz. *Die österreichische Volkshochschule. Magazin für Erwachsenenbildung*(Heft 259/67).

Futschek, M. (2016). Computational Thinking im Unterricht. *Schule Aktiv, Sonderheft Oktober 2016*, S. 4-5.

Grimus, M., & Ebner, M. (2014). Learning with Mobile Devices Perceptions of Students and Teachers at Lower Secondary Schools in Austria. *Proceedings of World Conference on Educational Multimedia, Hypermedia and Telecommunications 2014* (S. 1600-1609). Chesapeake: VA: AACE.

Guo, P. J., Kim, J., & Rubin, R. (2014). How video production affects student engagement: an empirical study fo MOOC videos. *L@S '14 Proceedings of the first ACM conference on Learning @ scale conference* (S. 41-50). Atlanta: ACM .

Guzdial, M. (2004). *Programming environments for novices.* Abgerufen am 11. 1. 2017 von

https://pdfs.semanticscholar.org/d226/07680e128436ba0f4c53bb7e42fc90749089.pdf

Jackson, N. (2013). MOOCs go to K12: Higher ed trend expands to high schools. *District Administration*. Abgerufen am 08. 03 2017 von https://www.districtadministration.com/article/moocs-go-k12-higher-ed-trend-expands-high-schools

Janisch, S., Ebner, M., & Slany, W. (2016). Pocket Code - freier Online-Kurs für Kinder. *Schule Aktiv! Coding - Ein Baustein der informatischen Bildung*, S. 43-46.

Janisch, S., Slany, W., & Ebner, M. (2016). Programmieren für Kinder. *L.A. Multimedia 2016, 2*, S. 40-41.

Jordan, K. (2015). MOOC Completion Rates: The Data. Abgerufen am 02. 03 2017 von http://www.katyjordan.com/MOOCproject.html

Käfmüller, K. (2016). *Begleitstudie eines Online-Kurses im Inverse-Blended-Learning-Format.* Technische Universität Graz: Fakultät für Informatik und Biomedizinische Technik.

Kaplan, A. M., & Haenlein, M. (2016). Higher education and the digital revolution: About MOOCs, SPOCs, social media, and the Cookie Monster. *Business Horizons*(Volume 59), S. 441-450. doi:10.1016/2016.03.008

Khalil, M., & Ebner, M. (2014). MOOCs Completion Rates and Possible Methods to Imporve Retention - A Literature Review. *Proceedings of World Conference on Educational Multimedia, Hypermedia and Telecommunications 2014* (S. 1236-1244). Chesapeake: VA: AACE.

Khalil, M., & Ebner, M. (2015). A STEM MOOC for School Children - What does learning analytics tell us? *Proceedings fo 2015 International Conference onf Interactive Collaborative Learning (ICL).* Florence: IEEE.

Kizilcec , R. F., Piech, C., & Schneider , E. (2013). Deconstructing disengagement: analyzing learner subpopulations in massive open online courses. *Proceedings of the Third International Conference on Learning Analytics and Knowledge* (S. 170-179). New York: ACM. Abgerufen am 9. 1. 2017 von http://rene.kizilcec.com/wp-content/uploads/2013/09/Kizilcec-Piech-Schneider-2013-Deconstructing-

Disengagement-Analyzing-Learner-Subpopulations-in-Massive-Open-Online-Courses.pdf

Maloney, J., Peppler, K., Kafai, Y., Resnick, M., & Rusk, N. (2008). *Programming by Choice: Urban Youth Learning Programming with Sratch.* Abgerufen am 12. 1. 2017 von http://web.media.mit.edu/~mres/papers/sigcse-08.pdf

Maloney, J., Resnick, M., Rusk, N., Silverman, B., & Eastmond, E. (2010). The Scratch Programming Language and Environment. *ACM Transactions on Computing Education*. Abgerufen am 11. 1. 2017 von web.media.mit.edu/~jmaloney/papers/ScratchLangAndEnvironment.pdf

Nagler, W., Ebner, M., & Schön, M. (2016). R.I.P E-Mail * 1965 - 2015. *World Conference on Educational Media and Technology, Hypermedia and Telecommunications 2015* (S. 464-473). Chesapeake: VA: AACE.

Pappert, S. (1980). *Mindstorms. Children, Computers and Powerful Ideas.* New York: Basic Books.

Petri, A., Schindler, C., Slany, W., Spieler, B., & Smith, J. (2015). Pocket Game Jams - a Constructionist Approach at Schools. *MobileHCI*, S. 25-28.

Resnick, M., Maloney , J., Monroy-Hernandez, A., Rusk, A., Eastmond, N., Brennan, K., . . . Kafai, Y. (2009). Scratch: Programming for all. *Communications of the ACM*. Abgerufen am 11. 1. 2017 von http://web.media.mit.edu/~mres/papers/Scratch-CACM-final.pdf

Riegler, B. (2016). Kulturtechnik des 21. Jahrhunderts: Warum Kinder programmieren können sollen. *derStandard*. Abgerufen am 1. Januar 2017 von http://derstandard.at/2000045774124/Kulturtechnik-des-21-Jahrhunderts-Warum-Kinder-programmieren-koennen-sollen

Rigby, J. (2015). Is there any science behind the lack of women in science? *The Telegraph*. Abgerufen am 06. 03 2017 von http://www.telegraph.co.uk/women/womens-health/11401344/STEM-Is-there-any-science-behind-the-lack-of-women-in-science.html

Schippers,, S., & Mak, M. (2015). creating-outstanding-experiences-for-digital-natives. A survey of Digital Natives reveals a group of impatient users with fragmented attention spans who demand fast and intuitive products and services. *UXmag*. Abgerufen am 15. 03 2017 von

http://uxmag.com/articles/creating-outstanding-experiences-for-digital-natives.

Spieler, B., Petri, A., Schindler, C., Slany, W., Beltran, M., Boulton, H., . . . Smith, J. (2016). Pocket code: a mobile app for game jams to facilitate classroom learning through game creation. *The Irish Conference on Game-Based Learning* (S. 1-14). Dubling: iGBL.

Vuorikari, R., Punie, Y., Carretero, S., & Van den Brande, L. (2016). *DigComp 2.0: The Digital Competence Framework for Citizens.* Brüssel: European Commission. doi:10.2791/11517

Wedekind, J. (2013). MOOCs - eine Herausforderung für die Hochschulen? In G. Reinmann, M. Ebner, & S. Schön, *Hochschuldidaktik im Zeichen von Heterogenität und Vielfalt* (S. 45-62). Norderstedt: BoD.

Wing, J. (2006). Computational Thinking. *Communications of the ACM, 49(3)*, S. 33-35.

Abbildungsverzeichnis

Tabellenverzeichnis

Anhang

Anhang 1: Transkripte der Videos

Anhang 2: Post Test Schüler

Anhang 3: Feedback-Formular Schüler

Anhang 1: Video-Transkripte

Kapitel 1 – Mein erstes eigenes Programm:

Video 1.1 Installation

So lasst uns anfangen. Zuerst einmal müssen wir unser Smartphone rausholen. Bevor wir mit dem Programmieren beginnen können, musst du Pocket Code auf deinem Smartphone installieren. Gehe dazu zum Google Play Store und suche nach Pocket Code. Hast du die App gefunden, drücke auf Installieren. Um eigene Bilder zu zeichnen, brauchst du auch Pocket Paint. Perfekt. Jetzt hast du alles, was zum Programmieren benötigt wird.

Video 1.2 Neues Programm erstellen

Es wird Zeit unser erstes eigenes Programm zu erstellen. Wir werden einen Zoo mit verschiedenen Tieren programmieren. Wenn du Pocket Code öffnest, gelangst du zum Startbildschirm. Der Startbildschirm erlaubt dir verschiedene Optionen. Wir wollen ein neues Programm erstellen und drücken deshalb auf Neu. Nun müssen wir dem Programm einen Namen geben. Ich gebe dem Programm den Namen Zoo. Bevor du auf ok drückst, stelle sicher, dass die „erstelle leeres Programm“-Box ausgewählt ist. Nun müssen wir noch entscheiden, ob das Programm auf Hoch oder Querformat laufen soll. Hier ein kleiner Tipp. Die Ausrichtung hat eine Rolle auf die Anordnung deiner Objekte. Überlege dir deswegen gut, wie dein Programm ausschauen soll, damit du nachher keine Probleme hast. Für unser erstes Programm wählen wir am besten Hochformat. Der erste Schritt wäre geschafft, wir haben nun ein leeres Programm mit dem Namen Zoo erstellt.

Video 1.3 Aufbau von Pocket Code

Wir befinden uns nun in der Hauptansicht von unserem Programm. Jedes Programm besteht aus einem Hintergrund und aus einer beliebigen Anzahl von Objekten. Das kannst du dir so ähnlich wie bei einem Theaterstück vorstellen, der Hintergrund ist die Bühne und jedes Objekt ist ein Schauspieler. Oder als Fußballspiel, der Hintergrund sind das Spielfeld und die Spieler die Objekte.

Somit gibt es jedes Objekt nur einmal und es ist nicht möglich, zwei Objekte gleich zu benennen. Da wir ein leeres Programm erstellt haben, gibt es noch kein Objekt. Rechts unten findest du den Play Button, mit dem wir unser Programm jederzeit anschauen können. Was passiert, wenn wir jetzt auf Play drücken? – Genau wir sehen einen weißen Bildschirm, da wir noch nichts gemacht haben. Mit Tippen auf zurück und dann auf den Zurück-Button kommen wir wieder zur Hauptansicht. Soviel zum Aufbau von Pocket Code, Zeit, unserem Programm Leben einzuhauchen.

Video 1.4 Neues Objekt programmieren

Jedes Programm braucht Objekte. Um ein Objekt zu erzeugen, tippe auf das + Symbol links unten in der Hauptansicht. Jetzt gibt es 4 Möglichkeiten: Wir können unser Objekt selber zeichnen, ein Objekt aus der Medienbibliothek laden, ein Bild aus unserer Gallery nehmen oder ein eigenes Bild aufnehmen. In unserem Fall tippen wir auf Medienbibliothek. Jetzt siehst du eine Auswahl von verschiedenen Bildern. Wir suchen uns ein Tier für unseren Zoo aus, ein Panda wäre ganz nett. Um es auszuwählen, tippe auf das Objekt. Jetzt kannst du dem Objekt noch einen passenden Namen geben. Wenn du auf ok drückst, wird das Objekt zu unserem Programm hinzugefügt, wie du in der Hauptansicht erkennen kannst. Nun wollen wir dem Objekt ein bestimmtes Verhalten geben, es also programmieren. Tippe dazu auf das Objekt, um zur Objektansicht zu kommen. Jedes Objekt hat Skripte, Klänge und Aussehen. Wenn wir auf Skripte tippen, kommen wir zur Skripte-Ansicht. Hier können wir Bausteine hinzufügen, mit denen wir unser Objekt programmieren können. Tippe dazu auf das + Symbol. Wir sehen nun die verschiedenen Baustein-Kategorien. Wir gehen zuerst einmal auf Steuerung und tippen auf den Baustein „Wenn Programm gestartet“. Den nächsten Baustein holen wir wieder aus der Steuerung-Kategorie. Wir nehmen den Baustein „Warte 1 Sekunde“ und platzieren ihn unter „Wenn Programm gestartet“. Als nächsten Baustein nehmen wir den Baustein „Ändere Größe um“, den wir unter Aussehen finden. Bei manchen Bausteinen gibt es Zahlen, die leicht unterstrichen sind. Das bedeutet, dass du diese Zahlen verändern kannst. Tippe auf die Zahl, um den Formeleditor zu öffnen. Hier kannst du nun einen anderen Wert eingeben. Wir wählen den Wert 50. Tippe auf ok, um den neuen Wert zu speichern. Probiere nun folgende Bausteine selber zu

finden, zu verändern und zu platzieren. Hat du das gemacht, sollten wir uns überlegen, was unser Objekt nun eigentlich macht. Wenn das Programm startet, wartet unser Objekt eine Sekunde, dann soll es größer werden, dann wartet es wieder eine Sekunde und dann wird es wieder kleiner. Hier ist wichtig zu wissen, dass die Bausteine von oben nach unten abgearbeitet werden. Schauen wir, ob das wirklich der Fall ist. Um unser Programm zu starten, tippen wir auf den Play Button. Siehe da, es funktioniert. Tippe nun auf Zurück, um zur Steuerung zu kommen, und hier noch einmal auf zurück, um zur Skripten Ansicht zu kommen. Wenn du jetzt noch einmal auf zurück tippst, kommt du wieder zu Objekt Ansicht. Wir haben nun unser erstes Tier programmiert. Natürlich können wir noch viele weitere Bausteine hinzufügen oder auch andere Tiere erstellen und programmieren.

Video 1.5 Aussehen verändern

Zusätzlich zur Veränderung der Größe wollen wir nun unserem Panda ein weiteres Verhalten geben. Gehen wir dazu wieder zur Objektansicht von unserem Objekt. Neben Skripten kann jedes Objekt Aussehen und Klänge haben. So wie ein Schauspieler mehrere Kostüme hat oder ein Fußballspieler mehrere Trikots, so kann auch ein Objekt mehrere Aussehen haben. Wir fügen unserem Objekt nun ein neues Aussehen hinzu. Tippe unter der Objektansicht dazu auf Aussehen und schließlich auf das plus Symbol links unten. Wir holen unser Aussehen aus der Medienbibliothek, und zwar einen Panda, der die Augen verdeckt hat. Unser Objekt hat nun 2 verschiedene Aussehen. Diese können wir nun auch im Programm einsetzten. Tippe dazu auf zurück, um wieder zur Objekt Ansicht zu kommen. Unter Skripte müssen wir wieder nach passenden Bausteinen suchen, mit denen wir das Aussehen von unserem Objekt verändern können. Diese finden wir unter der Kategorie Aussehen. Wähle den Baustein nächstes Aussehen und platziere ihn unter „Verändere Größe um“. Wähle noch einmal den gleiche Baustein und platzieren ihn unter „verändere Größe um -50“. Schauen wir nun unser Programm wieder an. Siehe da, immer wenn unser Panda die Größe verändert, wechselt er auch sein Aussehen. Ziemlich cool, oder?

Video 1.6 Ereignis und Klang

Bis jetzt beginnt unser Objekt mit seinem programmierten Verhalten, wenn das Programm startet. Es ist allerdings auch möglich das Objekt so zu programmieren, dass erst wenn ein bestimmtes Ereignis eintritt, das programmierte Verhalten gestartet wird. Das werden wir jetzt ausprobieren. Dazu müssen wir wieder passende Bausteine finden. Gehe dazu unter Skripte in der Objektansicht. Schauen wir mal was es unter der Kategorie Steuerung für Bausteine gibt. Wir nehmen hier den „Wenn angetippt Baustein“. Wähle diesen Baustein aus und platziere ihn entweder unter dem Block mit den „Wenn Programm gestartet“ Bausteinen oder oberhalb“. Da dieser Baustein oben rund ist, ist es nicht möglich diesen an einen anderen Baustein anzudocken. Wenn unser Objekt also nun angetippt wird, soll etwas passieren. Was genau müssen wir natürlich wieder programmieren. Gehe dazu diesmal zur Klang Kategorie. Tippe auf den Baustein Klang starten und platziere ihn unter Wenn angetippt. Wenn du genau hinschaust siehst du das das Neu unter Klang starten fein unterstrichen ist. Das bedeutet wieder, dass wir diesen Wert verändern können. Tippe auf das Neu und du siehst verschiedene Optionen. Wir können einen Klang von unserem Smartphone nehmen, einen eigenen Klang aufnehmen oder wieder in die Medienbibliothek gehen. Wir tippen auf die Medienbibliothek um zu sehen welche Klänge es bereits gibt. Wenn du auf Play drückst ,kannst du dir den jeweiligen Klang anhören. Um ihn herunterzuladen, tippe auf den Titel des Klanges. Für unseren Panda nehmen wir am besten ein Tier Geräusch .Wir probieren unser Programm aus und schauen ob es funktioniert. Wenn wir das Programm starten, verändert der Panda nun seine Größe und Aussehen und jedes Mal wenn wir auf ihn tippen, spielt er seinen Klang ab. Nicht schlecht, wir sehen also dass wir unser Programm interaktiv machen können, erst wenn ein bestimmtes Ereignis auftritt, in diesem Fall der Panda angetippt wird, wird ein bestimmtes Verhalten gestartet

Video 1.7 Position verändern

Im letzten Video dieses Kapitels schauen wir uns an, wie wir ein zweites Objekt erzeugen können. Da ein Zoo mit einem Tier etwas langweilig ist, werden wir ein zweites Tier zu unserem Programm hinzufügen. Tippe dazu auf das Plus Symbol in der Hauptansicht. Wir gehen wieder zur Medienbibliothek und wählen den Luchs aus. In der Hauptansicht siehst du nun, dass wir zwei Objekt haben.

Schauen wir uns mal an, wie das in unserem Programm ausschaut, indem wir auf den Play Button drücken. Oje, wir haben zwar beide Objekte im Programm, doch leider sind sie übereinander. Das werden wir jetzt ändern. Geh zurück zur Hauptansicht und tippe auf den Luchs, um zur Objektansicht zu kommen. Damit nicht beide Objekte übereinander sind, wäre eine Möglichkeit, den Luchs an einer anderen Stelle des Bildschirms zu platzieren. Wenn das Programm startet, soll der Luchs an einer anderen Position sein als unser Panda. Wir brauchen somit wieder den Baustein, wenn Programm startet, den wir unter der Kategorie Steuerung finden. Um die Position unseres Objektes zu verändern, müssen wir unter Bewegung gehen. Dort gibt es unter anderem den Baustein „Setze an Position“. Tippe auf ihn und platziere ihn unterhalb von Wenn Programm gestartet. Unter der Schrift „Setze an Position“ siehst du X: und Y: mit dazugehörigen Werten. Da die Werte wieder unterstrichen sind, bedeutet das, dass wir sie verändern können. Doch was bedeuten eigentlich diese Werte? Dazu müssen wir einen kurzen Ausflug in die Mathematik zum sogenannten Koordinatensystem machen. Ein Koordinatensystem besteht aus zwei Achsen, der waagrechten X-Achse und der senkrechten Y-Achse. Mit Hilfe dieses Systems können wir Punkte mathematisch beschreiben. Sehen wir uns diese an einem Beispiel an. Wie gelangen wir zu dem aufgezeichneten Punkt? Wir starten immer in der Mitte des Koordinatensystems, auch Nullpunkt genannt. Zuerst einmal gehen wir 2 Schritte in X-Richtung. Nun müssen wir noch 3 Schritte nach oben also in Y-Richtung gehen. Unser Punkt hat somit die Koordinaten (2/3). Wie schaut es mit diesem Punkt aus. Hier müssen wir vom Nullpunkt zuerst noch Schritte nach links gehen und dann wieder 2 Schritte nach unten. Unser Punkt hat somit die Koordinaten (-3/-2). Das gleiche Prinzip gilt auch für Pocket Code. Auch hier gibt es ein Koordinatensystem und standardmäßig werden alle Objekte im Nullpunkt platziert. Wenn wir unseren Luchs nun anders platzieren möchten, müssen wir seine Koordinaten verändern. Wir wollen ihn direkt über unseren Panda setzen, welche Koordinaten müssen wir somit eingeben? Tippe zuerst einmal auf die Zahl, um zum Formeleditor zu kommen. Setzte X auf 0 und Y auf 600. Tippe auf ok, um die Auswahl zu bestätigen. Wenn wir jetzt unser Programm anschauen, sehen wir, dass der Luchs oberhalb des Pandas ist. Um das Koordinaten-System zu sehen, tippe auf zurück und unter der Steuerung auf Achsen ein. Für X hatten wir den Wert 0 und für Y

600. Das Objekt befindet somit auf der Y-Achsen auf der Position 600. Mit Koordinaten richtig umzugehen, ist sehr wichtig, da wir das sehr oft brauchen werden.

Video 1.8 Hintergrund, Löschen und Verschieben

Hintergrund:

Wenn wir unser Programm als Schauspiel betrachten, dann sind die Schauspieler die Objekte und der Hintergrund die Bühne. Deswegen gibt es in jedem Programm nur einen Hintergrund. Der Hintergrund kann aber - gleich wie Objekte - verschiedene Aussehen haben. Diese heißen allerdings nicht Aussehen, sondern Hintergründe. Die Hintergründe können sich während des Programms natürlich ändern, denn gleich wie Objekte ist es auch möglich, den Hintergrund zu programmieren.

Objekte und Bausteine löschen:

Um Objekte zu löschen, tippe auf die Menü-Taste. Bei meinem Smartphone befindet sie sich links unten. Im Pop-Menü findest nun verschiedene Optionen, unter anderem Löschen, Kopieren oder Umbenennen. Derselbe Trick funktioniert auch bei Bausteinen.

Objekte verschieben:

Um Objekte zu verschieben, tippe so lang auf ein Objekt, bis es die Farbe ändert. Nun kannst du es verschieben. Der gleiche Trick funktioniert auch wieder bei den Bausteinen.

Video 1.9 Aufgabe: Programmiere einen Zoo

Wir haben angefangen einen Zoo zu programmieren. Deine Aufgabe ist es jetzt, diesen Zoo fertig zu machen und ihn mit verschiedenen Tieren zu füllen. Probiere mindestens 3 verschiedene Tiere hinzuzufügen, die alle ein unterschiedliches Verhalten haben. Die Tiere können sich bewegen, Geräusche von sich geben oder auch ihr Aussehen verändern. Natürlich kannst du auch eigene Tiere zeichnen oder einen Hintergrund hinzufügen. Es muss auch nicht unbedingt ein Zoo sein, du kannst kreativ sein und dich austoben, denn du bist der Programmierer. Probiere dabei vor allem folgende Bausteine zu verwenden. Du kannst entweder bei unserem Programm weitermachen oder ganz von neu beginnen. Geh dazu zur Startansicht und drücke auf Neu. In der Startansicht wer-

den unter Programme alle deine Programme gespeichert und du kannst zwischen den Programmen jederzeit wechseln.

Video 1.10 Hochladen von einem Programm

Wenn du fertig bist mit deinem Programm, geh zur Hauptansicht. Tippe dort auf den linken Menü-Button und dann auf Hochladen. Nun musst du dich einloggen oder ein neues Konto anlegen, wenn du noch keines besitzt. Keine Angst, das geht sehr schnell, du brauchst dafür nur eine E-Mail-Adresse. Nun kannst du noch einen passenden Namen für das Programm wählen. Damit du das Programm nachher wieder findest, macht es Sinn, einen Namen zu finden, den du dir gut merkst. Gleichzeitig sollte der Name aber auch dein Programm beschreiben. Ich benenne deswegen das Programm – Mein cooler Zoo – Pauli. Wenn du willst kannst du auch noch eine Beschreibung hinzufügen. Geh auf weiter und wähle eine passende Kategorie, zu der dein Programm gehört. Wenn du fertig bist, geh auf Programm anzeigen, um dein Programm zu sehen. Hier gibt es nun 3 Möglichkeiten: Wenn du auf Programm herunterladen drückst, wird das Programm in Pocket Code geladen und du kannst es ansehen und verändern. Dies macht dann Sinn, wenn es ein Programm von jemand anderem ist und du es ansehen und/oder verändern willst. Unter Erkunden in der Startansicht kannst du Programme ansehen, die Personen auf der ganzen Welt gemacht haben. Wenn du auf App erstellen drückst, kannst du dein Programm direkt als App auf deinem Smartphone speichern. Wenn du darauf getippt hast, musst du ein bisschen warten, bis der Button App herunterladen auftaucht. Hier kannst du nun das Programm als App installieren und schließlich allen deinen Freunden zeigen. Die letzte Möglichkeit ist die schon erwähnte Teilnahme am Galaxie Game Jam. Wenn du auf diesen Knopf tippst, nimmst du mit deinem Programm am Wettbewerb teil und kannst tolle Preise gewinnen. Beachte, dass es sich dabei um ein Spiel oder eine Geschichte rund um das Thema Weltraum handeln muss. Weitere Informationen findest du unter www.galaxygamejam.com. Im Rahmen des Kurses werde ich dir alle Werkzeuge zeigen, die du brauchst, um ein eigenes tolles Programm zu entwickeln, mit dem du am Wettbewerb teilnehmen kannst. Am Ende dieses Videos möchte ich dir noch einen Tipp geben: Wenn du dein Spiel hochgeladen hast und von mir höchstpersönlich Feedback dazu bekommen willst, dann poste den Namen

deines Programms in das Kursforum. Ich werde mir dann das Programm anschauen und dir Feedback dazu geben. Ich hoffe, du zeigst mir oder auch Freunden von dir ein paar deiner Programme. Vergiss nicht, genau wie du von anderen lernen kannst, können auch andere viel von dir lernen.

Kapitel 2– Bis zur Unendlichkeit

Video 2.0 Objekte zeichnen

Willkommen bei Kapitel 2. Als Wiederholung von Kapitel 1 werden wie ein neues Objekt mit 2 Aussehen erstellen. Wenn du schon weißt, wie das geht, kannst du dieses Video auch überspringen. Zuerst einmal erstellen wir ein neues Programm. Das Programm benenne ich Animation. Nun drücke ich auf den Plus-Knopf. Ich kann nun wieder aus den verschiedenen Optionen wählen. Anstatt aber diesmal ein Objekt aus der Medienbibliothek zu laden, tippe ich auf Bild zeichnen, um mein eigenes Objekt zu zeichnen. Auch das geht in Pocket Code. Es öffnet sich eine Zeichenumgebung und in dieser gibt es verschiedene Werkzeuge wie Pinsel, Radiergummi und viele mehr. In meinem Fall zeichne ich einen kleinen Hasen. Um das Bild als Objekt zu speichern, tippe ich auf das Pocket Code Symbol in der linken Ecke. Nun hab ich ein Hasen-Objekt mit 1 Aussehen. Da ich aber 2 Aussehen haben will, gehe ich auf Aussehen. Hier kann ich das Aussehen kopieren. Dazu tippe ich auf die linke Menütaste und schließlich auf kopieren. Wenn ich auf die Kopie dieses Objektes tippe, wird wieder Pocket Paint geöffnet. Hier radiere ich nun die Arme weg, damit ich andere Arme hinzuzeichnen kann. Um das Objekt zu speichern, tippe ich wieder auf das Symbol in der linken oberen Ecke. Perfekt, nun haben wir ein Objekt mit 2 Aussehen. Mit diesem Objekt werden wir im nächsten Schritt eine Animation machen.

Video 2.1 Schleifen

In diesem Video werde ich eine kleine Animation machen und dabei werden wir das Konzept von Schleifen kennenlernen. Schleifen sind ein wichtiges Werkzeug in der Programmierung, da sie uns viel Arbeit abnehmen können. Sehen wir uns dieses Programm an. In diesem Programm gibt es ein Objekt, das ich selbst gezeichnet habe. Dieses Objekt hat zwei Aussehen, einmal mit Pfoten

oben und einmal mit Pfoten unten. Ganz im Sinne von einem Trickfilm soll sich das Objekt bewegen, indem es zwischen den Aussehen wechselt. Schauen wir uns mal die Bausteine dazu an. Am Anfang verändere ich die Größen des Objektes, dann wartet es 1 Sekunde, bevor es sein Aussehen verändert. Dann wird wieder eine Sekunde gewartet, bis sich das Aussehen wieder verändert. Schauen wir uns das Programm mal an. Ok, es funktioniert, aber wäre es nicht schöner, wenn sich der Hase die ganze Zeit, wenn das Programm läuft, bewegen würde? Dazu brauchen wir ewig viele Bausteine….nein, ganz im Gegenteil, ein bestimmter Baustein reicht völlig aus. Ich gehe unter Steuerung und tippe hier auf den Baustein „Wiederhole fortlaufend“. Diesen platziere ich über meinem Aussehen-Bausteinen. Wenn dieser Baustein platziert wird, wird automatisch am Ende immer noch ein Baustein“ Ende der Schleife“ platziert. Schauen wir uns nun das Programm an.
Es ist gleich wie vorher, aber siehe da, der Hase hört nicht mehr mit der Bewegung auf. Alle Bausteine, die sich zwischen „Wiederhole fortlaufend“ und“ Ende der Schleife“, sich also in der Schleife befinden, werden unendlich Mal ausgeführt. Ein ähnlicher Baustein ist dazu „Wiederhole 10-mal“. Im Gegensatz zu „Wiederhole fortlaufend“ kannst du hier bestimmen, wie oft etwas ausgeführt werden soll. Durch Schleifen ersparen wir uns viel Arbeit und wir können ganz leicht etwas öfter ausführen lassen.

Video 2.2 Aufgabe: Wir programmieren eine Animation.
Probiere nun selber Schleifen in deinem Programm zu verwenden. Erstelle eine Animation mithilfe von Schleifen. Dein Objekt kann zum Beispiel das Aussehen verändern oder natürlich auch andere Sachen machen. Hier ein Tipp, wenn du ein selber gezeichnetes Objekt animieren willst. Zeichne zuerst einmal dein Objekt, kopiere dann das Aussehen und verändere dann die Kopie. Du kannst natürlich auch mehrere Objekte animieren. Ich bin schon gespannt auf deine Animation

Video 2.3 Kommunikation von Objekten
In diesem Video lernen wir, wie Objekte miteinander kommunizieren können. Ich habe hier ein Programm mit zwei Objekten. Beide Objekte habe ich aus der Medienbibliothek heruntergeladen. Ein Objekt ist ein Alien, das andere ein

Knopf. Daraus will ich eine kleine Steuerung machen. Wenn ich auf den Knopf drücke, soll sich der Alien nach oben bewegen. Das heißt, der Alien muss irgendwie wissen, dass der Knopf angetippt wurde; die Objekte müssen miteinander kommunizieren. Dazu gehe ich zuerst einmal zum Objekt Knopf unter Skripten. Die ersten zwei Bausteine sind nur für die Position des Knopfes. Sie platzieren den Knopf unter dem Alien. Welchen Baustein brauche ich als nächstes? Wenn ich auf den Knopf tippe, soll etwas passieren, als nächsten Baustein brauche ich somit den „Wenn angetippt“-Baustein. Nun muss der Knopf dem Alien irgendwie mitteilen, dass er angetippt wurde. Das geht nun mit dem Baustein „Verschicke an alle“. Die Nachricht, die verschickt wird, muss allerdings noch definiert werden, deswegen drücke ich hier auf Nachricht1 und dann auf Neu . Die Nachricht meines Knopfes soll lauten: Geh nach oben. Diese Nachricht wird nun an alle meine vorhandenen Objekte geschickt, sobald der Knopf angetippt wird. Nun muss ich allerdings noch zum Alien gehen und ihm sagen, was er überhaupt machen soll, wenn er diese Nachricht empfängt. Dazu wähle ich den „Wenn ich empfange“-Baustein aus. Wenn ich nun die Nachricht „Geh nach oben“ empfange, dann soll der Alien etwas machen. Das werde ich nun programmieren. Ich nehme zwei Mal Gleite Bausteine, denn wenn der Knopf angetippt wird, soll der Alien nach oben und dann wieder nach unten gleiten. Schauen wir uns das Programm einmal an. Siehe da, es funktioniert. Damit Objekte miteinander kommunizieren können, brauchst du also nichts anderes als diese beiden Bausteine. Jetzt bist du an der Reihe. Probiere eine Steuerung mit verschiedenen Knöpfen zu machen. Bei jedem Knopf soll dein Objekt etwas anderes machen. Versuche mindestens 2 verschiedene Knöpfe einzubauen.

Video 2.4 Aufgabe: Eine kleine Button-Steuerung

Jetzt bist du an der Reihe. Probiere eine Steuerung mit verschiedenen Knöpfen zu machen. Bei jedem Knopf soll dein Objekt etwas anderes machen. Probiere mindestens 2 verschiedene Knöpfe einzubauen.

Video 2.5 Sensoren einbauen

Jetzt wird es spannend, wir lernen die Sensoren von unserem Smartphone einzusetzen. Das eröffnet uns viele neue Möglichkeiten. Dadurch können wir unser Programm interaktiv machen und in weiter Folge tolle Spiele programmieren. In

diesem Video kannst du wieder direkt mitmachen. Erstelle ein neues Programm und ein neues Objekt. Mein Objekt ist ein Pinguin aus der Medienbibliothek. Wir wollen dieses Objekt nun mit der Neigung unseres Smartphones bewegen. Neigen wir also unser Smartphone nach rechts, dann soll sich auch unser Pinguin nach rechts bewegen. Dazu brauchen wir zuerst denn“ Wenn Programm gestartet“-Baustein. Dann gehen wir unter Bewegung und wählen den“ Ändere X um 10“-Baustein aus. Den Wert können wir wieder verändern, indem wir auf ihn tippen. Im Formeleditor gibt es neben den Zahlen viele andere Optionen wie Funktionen, Logik oder auch Sensoren. Tippe auf Sensoren, um zu sehen, welchen Sensoren wir von unserem Smartphone einsetzen können. Wir nehmen hier den Neigungssensor in X-Richtung. Tippe auf ok, um die Änderungen zu speichern. Schauen wir uns unser Programm einmal an. Es scheint noch nicht zu funktionieren, obwohl ich das Smartphone bewege, bewegt sich das Objekt nicht. Das lässt sich dadurch erklären dass der X-Wert von unserem Objekt nur einmal um den Neigungswert des Sensors verändert wird. Wir wollen, dass der Wert X ständig geändert wird, wenn wir unser Smartphone gerade bewegen. Wir brauchen somit den Baustein „Wiederhole fortlaufend“. Probieren wir jetzt das Programm wieder aus, siehe da, es funktioniert. Wenn wir unser Smartphone neigen, bewegt sich auch der Pinguin. Wenn du genau hinschaust, siehst du allerdings noch, dass sich der Pinguin in die falsche Richtung bewegt. Neigen wir das Smartphone nach links, bewegt er sich nach rechts. Um dies zu ändern, musst du noch einmal zum Formeleditor gehen. Setz ein vor Neigung_X, um alle Werte umzudrehen. Wenn wir jetzt noch einmal unser Programm anschauen, bewegt sich der Pinguin in die richtige Richtung. Was müssen wir nun machen, damit sich der Pinguin auch nach oben und unten bewegt? Genau, wir machen das gleich noch einmal, nur diesmal mit dem Y-Wert und Y-Sensor. Nun bewegt sich der Pinguin auch nach oben oder unten. Wenn wir allerdings unser Smartphone zu stark bewegen, bewegt sich der Pinguin aus dem Bildschirm heraus. Auch das können wir noch ändern, indem wir den Prall vom Rand ab Baustein hinzufügen. Auch diesen musst du innerhalb der Schleife platzieren, da das Objekt ja immer vom Rand abprallen soll. Super, wir können auch Zensoren einsetzen. Ein Tipp noch am Ende.

Natürlich kann es aber auch gewollt sein, dass sich das Objekt gegengleich bewegt. Beachte auch, dass du Sensoren verstärken oder verringern kannst.

Zum Verstärken multipliziere den Wert des Sensors mit einer Zahl, zum Verringern dividiere den Wert. Sollte sich das Objekt zum Start bewegen, dann musst du die Sensorwerte verringern, indem du sie durch eine Zahl dividierst. Sollte die Bewegung zu gering sein, kannst du sie stärker machen, indem du die Sensorwerte mit einer bestimmten Zahl multipliziert. Welche Zahl das ist, musst du selber ausprobieren. In diesem Fall scheint Neigung_X sehr gut zu passen.

Video 2.6 Verfolgungsjagd

In dieser Aufgabe musst du die Konzepte von diesem Kapitel geschickt einsetzten. Deine Aufgabe ist es, ein kleines Spiel zu programmieren. Schauen wir es uns mal an. In dem Spiel gibt es einen Start-Knopf und wenn man auf diesen tippt, beginnt das Spiel. Das Ziel des Spieles ist, so lang wie möglich mit dem Ufo auf dem Planeten zu bleiben. Für dieses Spiel brauchen wir somit Schleifen, Sensoren und müssen Nachrichten verschicken. Bevor du mit dem Spiel anfängst, überlege dir, wie das Programm aufgebaut werden muss. Zuerst muss nur mal der Start-Knopf gezeigt werden. Erst wenn dieser angetippt wird, zeigen sich die anderen Objekte. Der Planet muss sich fortlaufend bewegen, wie genau und in welche Richtung bestimmst natürlich du. Das Ufo muss dann mithilfe von Sensoren gesteuert werden. Hier kannst du das Spiel natürlich schwieriger machen, indem du zum Beispiel die ?? von den Sensoren vertauscht. Natürlich kannst du auch eigene Objekte einsetzen.

Kapitel 3– Lass uns spielen

Video 3.1 Der Zufall

Ein mächtiges Werkzeug ist der Formeleditor in Pocket Code. In diesem Video zeige ich euch, was es für Möglichkeiten gibt. Schauen wir uns dieses Programm an. Es besteht aus einem blauen Hintergrund und einem Objekt. Unser Objekt ist ein Vogel und wir wollen, dass dieser Vogel auf dem Bildschirm herumfliegt. Die Bewegung können wir mit einer Schleife sowie mit Gleiten-Bausteinen programmieren. Mit dieser Methode bewegt sich der Vogel allerdings immer gleich, wäre es nicht schöner, wenn er sich immer anders bewegen würde. Einmal noch oben, einmal rechts, dann nach links…..ein zufälliges

Verhalten wäre perfekt. Auch das ist möglich in Pocket Code. Wir tippen auf den Wert X, um zum Formeleditor zu kommen. Im Formeleditor können wir neben Zahlen auch ganze Rechnungen eingeben. Außerdem haben wir schon gesehen, dass wir auch Sensoren als Wert nehmen können. Was auch möglich ist, sind mathematische Funktionen. Wenn wir auf Funktionen tippen, sehen wir, welche Möglichkeiten es gibt. Wie du siehst, gibt es einige Funktionen, eine der wichtigsten für uns ist die Funktion Zufall. Wenn wir auf sie tippen, wird sie ausgewählt. In der Zufallsfunktion gibt es zwei Werte, der erste Wert ist dabei der minimale Wert, der zweite der Maximale. Immer wenn die Funktion aufgerufen wird, wird ein Wert zwischen den definierten Minimum und Maximum ausgegeben. Dieser Wert wird von Pocket Code komplett zufällig generiert. Beim ersten Mal kann er zum Beispiel -250 dann 200, dann 100 sein…. und so weiter. Probieren wir die Zufallsfunktion auch für den Y-Wert einzusetzen. Hier nehmen wir als Grenzen -700 und 700?, das heißt, es treten zufällige Werte zwischen -700 und 700? auf. Schauen wir unser Programm einmal an. Siehe da, unser Vogel bewegt sich nun verschieden am Bildschirm herum. Immer wenn der Baustein ausgeführt wird, wird aufgrund der Zufallsfunktion ein anderer Wert ausgeführt. Wir können auch noch die Geschwindigkeit der Bewegung zufällig machen. Hier tippen wir auch den Wert vor Sekunde und im Formeleditor wählen wir wieder die Zufallsfunktion aus. Als Grenze setzen wir 0,5 und 3 Sekunden. Wenn wir das Programm starten, sehen wir, dass sich nun der Vogel unterschiedlich schnell bewegt. Einmal gleitet er zum Beispiel in einer Sekunde zu einer bestimmten Position, ein anderes Mal dauert das Gleiten vielleicht 5 Sekunden. Wir sehen also, dass wir mit der Zufallsfunktion ein interessantes Verhalten programmieren können.

Video 3.2 Aufgabe Reaktionsspiel

Probiere mithilfe der Zufallsfunktion ein Reaktionsspiel zu programmieren. Das Objekt taucht nur für eine bestimmte Zeit an zufälligen Positionen auf. Wenn man schnell genug ist und das Objekt berührt, wird ein Klang abgespielt sowie das Aussehen verändert. Du brauchst hier die Zufallsfunktion sowie Schleifen. Viel Spaß beim Tüfteln.

Video 3.3 Wenn Anweisungen (if)

Manchmal ist es notwendig, dass Objekte dann etwas Bestimmtes machen, wenn ein bestimmtes Ereignis auftritt. Wir reden von sogenannten Wenn-Anweisungen…..wenn ein bestimmtes Ereignis eintritt, dann passiert etwas. Sehen wir uns das mal bei einem Beispiel an. Wir haben hier ein Glas, das auf einem Tisch steht. Mit einem Sensor messen wir die Umgebungslautstärke. Immer wenn nun eine bestimmte Lautstärke erreicht wird, dann zersplittert das Glas. Huuu Ahh. Das Ereignis, das auftritt, ist die Lautstärke über einem bestimmten Wert und wenn dieses Ereignis auftritt, dann ändert das Glas sein Aussehen. Schauen wir, wie wir das im Programm umsetzen können. Wir haben ein Glas Objekt, das zwei Aussehen hat. Zuerst einmal passen wir die Größe des Objektes an. Nun brauchen wir den Baustein für die Wenn-Anweisung. Diesen finden wir unter Steuerung. Wir tippen auf ihn und platzieren ihn unter „Setze Größe auf“: Nun müssen wir die Bedingung definieren. Hier tippen wir auf den Wert, um zum Formeleditor zu kommen. Wir gehen zu Sensoren, um die Umgebungslautstärke zu messen. Wenn die Lautstärke nun einen bestimmten Wert übersteigt, dann soll etwas passieren. Dieser Wert muss jetzt noch definiert werden. Wenn wir auf Berechnen gehen, sehen wir zuerst einmal die Werte, die unser Sensor misst. Wenn ich lauter werde, werden auch die Werte höher. Anhand der Werte kann ich nun eine passende Grenze wählen. 60 scheint mir eine gute Grenze zu sein. Um die Grenze zu definieren, müssen wir unter Logik gehen. Hier gibt es die Zeichen wie ist gleich, ist nicht gleich, kleiner als, kleiner gleich, größer als und größer gleich. Wir nehmen das Zeichen größer und tippen gleich die Zahl 60 mit ein. Wenn die Umgebungslautstärke nun größer als 60 ist, diese Bedingung also wahr ist, dann wird das spezielle Event ausgeführt. Unter Berechnen können wir gleich testen, wann diese Bedingung wahr ist. Gehen wir nun zurück, um das Verhalten zu definieren, das ausgeführt werden soll, wenn diese Bedingung erfüllt ist. Das Glas soll zerspringen, deswegen gehen wir auf Aussehen und wählen hier den „Setze Aussehen“-Baustein. Diesen platzieren wir unter der Wenn-Anweisung. Immer wenn nun also die Umgebungslautstärke den Wert 60 überschreitet, dann soll das Glas zerspringen. Das Aussehen muss somit auf Glas kaputt gesetzt werden. Damit es allerdings wirklich funktioniert, brauchen wir noch eine Schleife, da der Sensor ja ständig arbeiten muss. Wir platzieren den „Wiederhole fortlaufende“-Baustein über der Wenn-Anweisung. Testen wir nun das Programm. Es

funktioniert… wenn eine gewisse Lautstärke erreicht wird, zerspringt das Glas. Damit das Glas wieder zum normalen Aussehen zurückwechselt wenn die Lautstärke unter dem Wert ist, müssen wir noch einen Baustein unter „Sonst“ setzen. Und zwar den Baustein „Setz Aussehen„: Glas. Wenn die Bedingung erfüllt wird, dann ist das Glas kaputt, ansonsten hat das Glas das normale Aussehen. Das ist die Magie hinter Wenn-Anweisungen im Programmieren, auch bekannt als If-statements. Wenn du das verstanden hast, dann probiere die Aufgabe zu lösen.

Video 3.4 Aufgabe: Ich verstecke mich

Probiere folgendes Programm zu entwickeln. Wir haben hier einen Panda und wenn man den Panda anschaut, versteckt er sich. Wir brauchen für dieses Programm somit den Gesichtserkennung-Sensor sowie eine Wenn-Anweisung.

Video 3.5 Physikalische Objekte

Jetzt wird es sehr spannend. Wir schauen uns die Physik-Bausteine von Pocket Code an, die uns eine Reihe an Möglichkeiten bieten. Mit diesen Bausteinen können wir Kollisionen machen oder auch Objekte mithilfe der Schwerkraft steuern. Schauen wir uns das an einem einfachen Bespiel an. Wir erzeugen ein neues Objekt in der Form eines Balles. Dieser Ball soll sich nun so wie ein Ball in der Realität verhalten. Wenn wir den Ball loslassen, soll er hinunterfallen. Dazu müssen wir ihn als physikalisches Objekt definieren. Das können wir mithilfe des Bausteines Setze Bewegungstyp auf „Aufprallen mit Gravitation“. Wenn wir unser Programm starten, sehen wir, dass der Ball nun langsam hinunterfällt. Damit er vom Rand abprallt, brauchen wir eine Schleife sowie den „Pralle vom Rand ab“-Baustein. Wir haben nun einen Ball, der hinunterfällt und am Rand auf- und abspringt. Dies passiert allerdings sehr langsam, aber das können wir leicht ändern, indem wir die Schwerkraft stärker einstellen. Dazu brauchen wir den Baustein „Setze Gravitation für alle Objekte auf“. Wenn wir zum Beispiel den Wert bei Y größer machen, dann fällt auch unser Objekt schneller. Was passiert, wenn wir bei Y den Wert aufsetzen und bei X einen Wert eingeben. Genau, dann wirkt die Schwerkraft auf die Seite. Wir können unser Objekt auch mithilfe der Schwerkraft steuern. Dazu nehmen wir als Werte die Werte von den Neigung Sensoren des Smartphones. Natürlich müssen wir

dann den Baustein in die Wiederhole fortlaufend-Schleife setzen, da wir ständig die Werte messen müssen. Siehe da, je nachdem, wie wir unser Smartphone halten, der Ball will ständig zu Boden fallen. Durch Multiplikation der Sensoren können wir die Werte der Sensoren noch verstärken. Beachte, dass der Schwerkraft-Baustein für alle Objekte gilt. Wenn wir also noch ein zweites physikalisches Objekt haben, dann verhält sich auch dieses gleich wie das erste.

Video 3.6 Kollisionen

Nun schauen wir uns Kollisionen an. Wir haben einen Ball, der ein physikalisches Objekt ist und mithilfe der Sensoren gesteuert wird. Um eine Kollision zu haben, brauchen wir nun ein anderes Objekt. Hier holen wir ein Objekt aus der Medienbibliothek. Dieses Objekt definieren wir nun als festes Objekt. Dazu brauchen wir den Baustein, „Setze Bewegungstyp auf“ und als Bewegungstyp definieren wir „andere prallen davon ab“. Nun prallt der Ball von diesem Objekt ab. Das Coole an physikalischen Objekten ist, dass wir Kollisionen messen können. Wenn nun eine Kollision mit dem Ball passiert, können wir ein bestimmtes Verhalten programmieren. Dazu brauchen wir den Baustein „Wenn physikalische Kollision mit“. Im Dropdown-Menü wählen wir unser Ball-Objekt aus. Nun können wir unserem Objekt Duster ein Verhalten geben, das ausgeführt wird, wenn die beiden Objekte miteinander kollidieren. In unserem Fall soll er einen Klang von sich geben sowie seine Farbe ändern. Wir können natürlich auch dem Ball ein Verhalten geben, wenn er mit Duster kollidiert. Dazu gehen wir zu unserem Ball-Objekt und wählen hier wieder den „Wenn physikalische Kollision mit“-Baustein aus. Diesmal wählen wir aber als Objekt Duster. Wenn die Kollision stattfindet, soll unser Ball kleiner werden. Probieren wir es aus. Tadaaaaa, wie du siehst, können wir so ganz leicht Kollisionen machen. In der nächsten Aufgabe kannst du versuchen ein Spiel mit Kollisionen zu entwickeln.

Video 3.7

Jetzt gilt es die Physik-Bausteine gezielt einzusetzen. Probieren wir dieses Spiel zu programmieren. Wir haben einen Ball und der Ball muss vom Ufo aufgehalten werden. Er prallt natürlich vom Rand und vom Ufo ab. Überlegen wir uns, was wir dazu brauchen. Das Ufo muss ein festes physikalisches Objekt sein und somit als Bewegungstyp „andere prallen davon ab“ haben. Zusätzlich

soll es mit der Neigung des Handys gesteuert werden, die X-Position muss somit mit der X-Neigung gesteuert werden. Der Ball bewegt sich und er hat als Bewegungstyp „Aufprallen mit Gravitation“. Da der Ball immer nach unten fliegen soll, müssen wir eine Schwerkraft in die Y-Richtung setzen. Damit er richtig abprallt vom Ufo, muss er eine Geschwindigkeit bekommen, wenn eine Kollision mit dem Ufo stattfindet. Du siehst also, dass wir für dieses Programm einige Konzept brauchen. Viel Erfolg beim Lösen.

Kapitel 4– Punkte und Eigenschaften

Video 4.1 Punktestand

In diesem Video beschäftigen wir uns mit Variablen. Doch was sind eigentlich Variablen? Variablen sind Platzhalter mit zugewiesenen Werten, die während des Programmablaufes verändert werden können. Sie werden häufig eingesetzt, um die Zeit einer Stoppuhr zu speichern oder einen Punktestand anzeigen zu lassen. Mithilfe von Variablen probieren wir nun einen Punktestand in unser Programm einzubauen. Wir haben hier ein Objekt und immer, wenn das Objekt angetippt wird, soll sich der Punktestand um eins erhöhen. Beim Start des Programms soll natürlich der Punktestand bei 0 sein. Um zu starten, gehen wir zur Kategorie Daten. Dort finden wir den Baustein „Setze Variable“. Bevor wir mit Variablen arbeiten können, müssen wir die Variable definieren, das machen wir mit diesem Baustein. Wir tippen auf „Neu“, um der Variable einen passenden Namen zu gaben. Da diese Variable den Punktestand speichern soll, nennen wir sie Punktestand. Zusätzlich soll sie von jedem Objekt genutzt werden können, deswegen wählen wir „Für alle Objekte“. Nun ändern wir noch den Startwert von unserem Punktestand Variable auf 0. Immer wenn unser Objekt nun angetippt wird, soll sich der Punktestand um eins erhöhen. Dazu platzieren wir nun den „Ändere Variable“-Baustein unter „Wenn angetippt“. Hier wählen wir die Variable Punktestand aus und als Wert wählen wir 1. Das einzige, was nun fehlt, ist, dass unser Punktestand auch am Bildschirm angezeigt wird. Dazu brauchen wir den „Zeige Variable“-Baustein, den wir zum „Wenn Programm gestartet“-Block hinzufügen. Hier wählen wir wieder unsere Variable aus und definieren die Position, an der sie angezeigt werden soll. Probieren wir das

Ganze nun aus. Es funktioniert, wir tippen auf das Objekt und der Punktstand erhöht sich. Damit andere wissen, was dieser Wert bedeutet, wäre es noch schön, eine Bezeichnung dieses Wertes zu haben. Dazu erzeugen wir einen neue Variable mit Namen: Bezeichnung und setzten diese auf den Wert: „Punktestand“. Wie du siehst, können Variablen als Werte auch Wörter haben. Diese Variable müssen wir nun neben unserem Punkstand anzeigen lassen.

Video 4.2 It’s the final countdown

Variablen sind ziemlich praktisch und erlauben uns unser Programm besser zu steuern. So können wir neben einen Punktestand zum Beispiel auch einen Countdown einbauen. Damit könnte man zum Beispiel Spiele programmieren, in denen man in einer bestimmten Zeit möglichst viele Punkte erreichen muss. Schauen wir uns das an einem Beispiel an. Wenn die Zeit in unserem Spiel abgelaufen ist, soll ein Game Over-Bildschirm angezeigt werden. Zuerst müssen wir wieder eine Variable definieren. Diese benennen wir Countdown. Den Startwert setzten wir auf 30. Immer wenn nun eine Sekunde vergeht, soll die Variable um 1 reduziert werden. Wir holen uns somit den „Wiederhole fortlaufend“-Baustein und in die Schleife setzen wir den „Warte 1 Sekunde“-Baustein. Wenn nun eine Sekunde gewartet wurde, soll sich die Variable ändern. Als nächsten Baustein wählen wir somit den „Ändere Variable“-Baustein. Als Wert, der geändert werden muss, wählen wir hier -1. Nun muss die Variable noch am Bildschirm gezeigt werden. Dazu platzieren wir den „Zeige Variable“-Baustein. Wenn wir unser Programm testen, sehen wir einen Countdown, der von 30 startet und jede Sekunde eins hinunterzählt. Wenn er allerdings 0 erreicht, läuft er einfach weiter. Wir wollen nun, dass bei 0 das Spiel aufhört und Game Over angezeigt wird. Hier brauchen wir wieder eine sogenannte Wenn-Anweisung. Wenn der Countdown 0 ist, dann zeige Game Over. Am besten eignet sich hier der Baustein „Warte bis X wahr“ ist. Aus Übersichtsgründen platzieren wir diesen bei einem neuen „Wenn Programm gestartet“-Block. Wir tippen auf den Wert, um die Bedienung zu ändern. Im Formeleditor gehen wir auf Daten, um unsere Variable auszuwählen und unter Logik holen wir uns ein „Ist gleich“. Nun tippen wir noch 0 ein und fertig ist unsere Bedingung. Das Programm wartet nun so lange, bis unsere Countdown-Variable 0 ist und dann soll etwas passieren. In unserem Fall wird eine Nachricht mit Titel Game Over verschickt. Nun

müssen wir ein Game Over-Objekt erstellen und hier entsprechende Bausteine hinzufügen. Wenn das Programm startet, soll es nicht da sein, und wenn es die Nachricht Game Over empfängt, dann soll es sich zeigen. Probieren wir das Programm wieder aus. Perfekt, wenn die Zeit nun vorbei ist, wird Game Over angezeigt. Natürlich können wir auch noch die anderen Objekte verbergen, wenn Game Over angezeigt wird. Neben einen Punktestand können wir somit auch einen Countdown einbauen. Überlege dir andere Situationen, in denen Variablen sinnvoll sind.

Video 4.3 Baue einen Punktestand ein

In dieser Aufgabe probieren wir mithilfe von Variablen einen Punktestand einzubauen. Geh zuerst unter Erkunden in der Startansicht und gib im Suchfeld die Zahl 13269 ein. Nun sollte ein Programm angezeigt werden. Lade dir dieses Programm herunter und öffne es anschließend. An diesem Programm musst du nun weiterarbeiten. Das Programm ist ein Spiel und Ziel des Spieles ist es, den blauen Alien zu fangen, ohne den grünen zu berühren.

Probiere hier nun einen Punktestand einzubauen. Immer wenn du den blauen Alien berührst, bekommst du einen Punkt, wenn du den anderen berührst, wird ein Punkt abgezogen. Du brauchst hierfür ein Variable sowie ein Punkte-Objekt. Wenn du willst, kannst du natürlich auch noch ein Zeitlimit einbauen. Viel Erfolg!

Video 4.4 Objekt Eigenschaften

Jedes Objekt hat gewisse Eigenschaften – und was wir mit diesen Eigenschaften machen können, schauen wir uns anhand von diesem Beispiel an. Wir haben als Objekt einen Luftballon mit zwei Aussehen. Den Luftballon blasen wir auf und bei gewisser Größe soll er platzen, also das Aussehen ändern. Wie schaut das dazugehörige Skript aus? Zuerst einmal setzen wir den Luftballon auf eine bestimmte Größe. Immer wenn wir ihn antippen, soll er nun die Größe verändern, die weiteren Bausteine sind also „Wenn angetippt“ und „Ändere Größe um“. In diesem Beispiel ändert er die Größe jedes Mal um 10. Nun wird es spannend; wir wollen, dass er bei einer gewissen Größe platzt. Dazu brauchen wir zuerst einmal eine Wenn-Anweisung. Wenn du eine bestimmte Größe erreichst, dann platze. Wir tippen somit auf den Baustein „Wenn … wahr ist dann…..“. In der Bedingung brauchen wir nun die Eigenschaft Größe des Ob-

jektes. Wir tippen auf den Wert, um zum Formeleditor zu kommen und tippen hier auf Objekt. Nun sehen wir alle Eigenschaften, die das Objekt hat. Unter physikalische Eigenschaften finden wir nun die Größe. Wir wählen diese aus und um unsere Bedingung abzuschließen, brauchen wir noch die Logik-Operatoren. Wir gehen unter Logik und wählen größer als. Als Wert nehmen wir 100. Wenn die Größe des Ballons größer ist als 100, dann soll nun etwas passieren. Der Ballon soll platzen, also das Aussehen wechseln. Bevor wir das Programm testen, können wir uns überlegen, was nun passieren wird. Wenn das Programm startet, ist die Größe des Ballons auf 70% der Originalgröße. Wenn wir auf ihn tippen, hat er 80%, beim nächsten Mal 90% dann 100% und dann 110%. Wir müssen auf den Ballon also 4-mal tippen, bevor er platzt. Probieren wir das nun aus. Super, es stimmt. Schauen wir uns noch schnell ein anderes Beispiel an, in dem wir die Eigenschaft von einem Objekt nützen können. In einem früheren Kapitel hatten wir die Aufgabe, ein kleines Pongspiel zu machen. In diesem Spiel hatten wir ein Ufo und einen Alien. Mit dem Ufo mussten wir den Alien abwehren. Wenn wir den Alien verfehlt haben, ist dieser einfach auf den Boden gefallen. Das können wir nun verbessern. Wir können ein Game Over einbauen, wenn das Ufo den Ball verfehlt. In anderen Worten, wenn die Y-Position des Balles einen gewissen Wert erreicht, dann soll Game Over angezeigt werden. Dazu brauchen wir nun wieder eine Wenn-Anweisung und als Bedingung nehmen wir die Eigenschaft Position_Y. Wenn die Position des Objektes nun kleiner oder gleich wie -450 ist, dann soll Game Over verschickt werden. Natürlich müssen wir ein entsprechendes Game Over-Objekt haben. Probieren wir es aus. Immer wenn wir den Ball nun verfehlen, ist das Spiel vorbei. Um das Spiel noch weiter auszubauen, könnten wir noch einen Knopf einbauen, der das Spiel wieder startet. Damit haben wir schon ein richtig nettes, kleines Spiel. Wie du sehen kannst, können wir uns die Eigenschaften von Objekten sehr gut zunutze machen. Nun bist du an der Reihe, probiere die nächste Aufgabe zu lösen.

Video 4.5 Aufgabe: Ausweichen Spiel

Für diese Aufgabe musst alles, was du bist jetzt gelernt hast, geschickt einsetzen. Probiere folgendes Spiel zu programmieren. In dem Spiel musst mit einem Objekt, in diesem Fall dem blauen Alien, einem herabfallenden Objekt, dem

grünen Alien, ausweichen. Generell brauchst du für dieses Programm 3 Objekte, eine Hauptfigur, ein Objekt, das hinunterfällt, und einen Game Over-Bildschirm.
Hier ein paar Hinweise, um die Aufgabe zu lösen. Der blaue Alien muss ein festes physikalisches Objekt sein. Zusätzlich muss er mit der Neigung deines Smartphones gesteuert werden. Das herunterfallende Objekt muss als dynamisches physikalisches Objekt definiert werden. Ganz am Anfang des Programms muss es ganz oben platziert werden. Beachte, dass du Objekte auch außerhalb des Bildschirms platzieren kannst. Da er ein dynamisches physikalisches Objekt ist, fällt er von selbst, allerdings musst er, wenn er den Bildschirm verlässt, wieder ganz oben außerhalb des Bildschirmes platziert werden. Das heißt, wenn seine Position eine gewisse Grenze überschreitet, soll er wieder ganz oben platziert werden. Da es spannender ist, wenn er immer an verschiedenen X-Positionen fällt, macht es Sinn für die X-Position eine Zufallsfunktion zu verwenden. Wenn eine physikalische Kollision passiert, dann soll Game Over verschickt werden. Das Game Over-Objekt soll dann angezeigt werden. Diese Aufgabe ist wirklich schon sehr schwer, wenn du sie lösen kannst, dann bist du wirklich schon ein Meister-Programmierer

Kapitel 5– Zielgerade

Video 5.1 Klonen von Objekten

In diesem Video schauen wir uns eine tolle neue Funktion von Pocket Code an und zwar, wie wir Objekte klonen können. Das schauen wir uns anhand von einem einfachen Beispiel an. Wir haben ein Objekt Vogel und dieser Vogel gleitet zufällig über den Bildschirm. Diesen Vogel wollen wir nun klonen. Immer wenn wir den Vogel antippen, soll ein Klon erzeugt werden. Wir brauchen nun den „Erzeuge Klon“-Baustein, den wir unter Steuerung finden. Im Dropdown können wir auswählen, von welchem Objekt der Klon erzeugt werden soll. In diesem Fall von dem Objekt, in dem wir uns befinden, deswegen wählen wir hier „mir“ aus. Probieren wir das Programm einmal aus. Ja, es funktioniert. Super!. Der Klon macht allerdings noch nichts, auch das können wir ändern mit einem speziellen Baustein. Wir gehen unter Steuerung und finden hier den

Baustein „Wenn ich als Klon entstehe“. Jetzt können wir dem Klon ein Verhalten geben. In diesem Beispiel soll der Klon sein Aussehen ändern, die Größe und Farbe sowie sich um seine Achse drehen. Probieren wir das Programm noch einmal aus. Perfekt, alle unsere Klone haben nun ein anderes Verhalten. Das war nur ein Beispiel, wie du Klone verwenden kannst, ich bin mir sicher, dir fallen tolle Möglichkeiten dafür ein.

Video 5.2 Touchscreen Steuerung

Die neueste Version von Pocket Code erlaubt uns nun auch, den Touchscreen von unserem Smartphone besser einzusetzen. In diesem Video schauen wir uns an, wie wir ein Objekt mit dem Touchscreen bewegen können. Dazu brauchen wir zuerst einmal irgendein Objekt. Als nächsten Baustein wählen wir „Wenn der Bildschirm berührt wird“. Unter Bewegung finden wir nun den Baustein „Gehe zu“. Also Option gibt es hier Zufallsposition oder Berührungsposition. Wir nehmen Berührungsposition. Testen wir nun das Programm. Überall wo wir den Bildschirm berühren, wird unser Objekt platziert. Allerdings passiert noch nichts, wenn wir mit unserem Finger über den Bildschirm wischen. Das können wir nun ganz leicht ändern, indem wir einen“ Wiederhole fortlaufend“-Baustein platzieren. Nun können wir das Objekt mit unserem Finger steuern. Ein weiteres Feature ist, das gemessen werden kann, wo der Touchscreen berührt wird. Je nachdem, wo ich zum Beispiel den Touchscreen berühre, soll ein anderes Verhalten gezeigt werden. Dazu wählen wir zuerst den „Wenn…wahr ist“-Baustein aus. Als Bedingung gehen wir nun unter Gerät. Hier finden wir“ Bildschirm“_berührt. Ich wähle hier Bildschirm berührt_Y und > als 0, da mein Objekt nur dann etwas tun soll, wenn ich auf die obere Hälfte des Bildschirms tippe. In diesem Fall soll es dann einfach größer werden. Natürlich kann ich das auch noch andersrum machen. Berühre ich den Bildschirm in der unteren Hälfte, soll es kleiner werden.

Testen wir das Programm nun. Zur besseren Ansicht schalte ich auch die Achsen ein. Super! Je nachdem, wo ich den Bildschirm berühre, verändert mein Objekt seine Größe. Natürlich bewegt es sich auch auf die angetippte Position. So eine Touchscreen-Steuerung ist schon was Cooles, oder?

Video 5.3 Malstift

Als ganz neues Feature ist der Malstift hinzugekommen. Was man damit machen kann, schauen wir uns in diesem Video an. Wir haben hier ein Programm, in dem wir unser Objekt mithilfe der Gravitation steuern. Nun verwenden wir Bausteine aus der Kategorie Malstift. Zuerst einmal verwenden wir “Schalte Malstift“ ein. Wenn wir das Programm testen, sehen wir nun, dass unser Objekt eine Malspur hinterlässt. Mit den Malstift-Bausteinen können wir in unserem Programm also zeichnen.

Wir können das Programm nun erweitern, hierfür nehme ich die Bausteine „Wenn Bildschirm berührt“ „Wische Malspur Weg“. Testen wir nun das Programm. Siehe da, wir können nun eine Malspur zeichnen und wenn wir auf den Bildschirm tippen, wird die Malspur gelöscht sowie ein Stempel vom Objekt erzeugt.

Video 5.4 Fragen stellen

In der neuesten Pocket Code-Version können wir auch Fragen stellen. Wie das funktioniert, zeige ich euch hier. Ich habe hier ein Objekt und wenn das Programm startet, soll gleich eine Quizfrage gestellt werden. Hierfür gehe ich unter Aussehen und wähle den Frage-Baustein. Wenn ich auf „Wie ist dein Name tippe?“ kann ich nun meine eigene Frage erstellen. Meine Frage soll lauten „Wie heißt die Hauptstadt von Österreich?“. Die Antwort soll in einer neuen Variablen gespeichert werden, die ich Antwort1 benenne. Nun will ich überprüfen, ob die Antwort richtig ist und entsprechend richtig oder falsch zurückgeben. Ich nehme hier den Baustein „Wenn…wahr ist, dann…sonst“. Als Bedingung muss ich hier meine Variable nehmen, in der die Antwort gespeichert ist, und diese muss gleich sein wie die richtige Antwort, die in diesem Fall Wien lautet.

Wenn die Antwort richtig ist, soll nun das Objekt richtig sagen und wenn nicht, soll es falsch sagen. Schauen wir uns das Programm an. Die Frage erscheint und wenn ich als Antwort Wien gebe, kommt ein „Richtig“ von meinem Objekt. Ist die Antwort nicht Wien erscheint ein „Falsch“. Mit diesen Bausteinen kannst du somit ein tolles Quiz und dein Programm sehr interaktiv machen

Video 5.5 Aufgabe: Star-Wars-Spiel

In dieser Aufgabe gilt es, ein Spiel zu entwickeln, in dem schon einige verschiedene Konzepte in Verwendung sind. Schauen wir uns das Spiel einmal an. Wir haben ein Ufo, das wir mit der Neigung steuern, und immer wenn wir auf den Bildschirm tippen, schießt das Ufo einen Stern ab. Wenn dieser Stern den Alien trifft, bekommt man einen Punkt. Natürlich taucht der Alien immer an unterschiedlichen Stellen auf. Wie du siehst, ist dieses Programm schon ziemlich komplex. Bevor du mit dem Programmieren anfängst, überlege dir zuerst, was deine verschiedenen Objekte alles können müssen. Eine genaue Überlegung kann nachher viel Zeit sparen.

Ein Tipp, wie du die Sache mit dem Stern hinbekommst, damit er immer an der Stelle, wo das Ufo ist, erscheint. Zuerst brauchst du einmal eine Variable. Immer wenn der Bildschirm berührt wird, wird diese Variable auf den Wert der X-Position des Ufos gesetzt. Gleichzeitig soll nun der Stern erscheinen, und zwar an der Stelle, wo sich das Ufo befindet. Der Stern braucht somit für die richtige X-Position den Wert der Variablen. Der Y- Wert bleibt immer gleich, da sich das Ufo nicht rauf oder runter bewegt. Viel Spaß beim Tüfteln.

Video 5.6 Dein eigenes Programm

Yeah, endlich ist es soweit. Entwickle dein eigenes Programm. Dieses Programm kann eine Animation, ein Spiel oder auch etwas anderes sein. Die Bausteine, die wir in diesem Kurs nicht durchgenommen haben, kannst du natürlich auch verwenden, hier gilt es selber auszuprobieren, was diese machen. Alles ist erlaubt! Ich bin schon sehr gespannt auf dein Programm und würde mich freuen, wenn du es im Forum teilst!

P.S.: Du kannst mit deinem Programm dann natürlich am Galaxy Game Jam teilnehmen.

Anhang 2: Post-Test 1. Klasse AHS

Name:

Quiz

Rätsel 1

Wir haben 2 Objekte in unserem Programm und folgenden Code. Was wird unser Objekt „Katze“ machen wenn unser Programm gestartet wird?

Hund

Wenn angetippt
Ändere Größe um -50

Wenn Programm gestartet
Ändere Größe um 100
Verschicken an alle: „Achtung!“

Wenn ich empfange: Start
Verbergen

Katze

Wenn Programm gestartet
Drehe nach links 15 Grad

Wenn ich empfange: „Achtung!“
Ändere Größe um -50

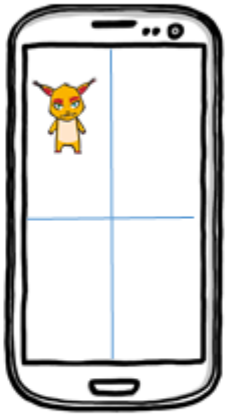
Wenn angetippt
Ändere Größe um 100

□ Das Objekt Katze dreht sich nach links
□ Das Objekt Katze wird kleiner

□ Das Objekt Katze wird größer
□ Das Objekt Katze dreht sich nach links und wird kleiner

Rätsel 2

Du willst dein Objekt an folgender Position auf den Bildschirm setzen (siehe Bild). Welche Koordinaten könnten dafür passen?



□ Setze an Position X: 400, Y: 600
□ Setze an Position X: -400, Y: -600

□ Setze an Position X: -400, Y: 600
□ Setze an Position X: 400, Y: -600

Rätsel 3

Wir haben eine Variable gegeben:

```
Punkte = 4
```

Welchen Wert hat die Variable Punkte nachdem folgender Code ausgeführt wird?

```
Wenn Programm gestartet

        Wiederhole 5 mal
                Ändere Punkte um 3
        Ende der Schleife
```

Die Variable Punkte hat den Wert: _____

Rätsel 4

Wir haben 2 Variablen gegeben:

```
Zähler = 0
Punkte = 3
```

Welchen Wert hat die Variable Punkte nachdem folgender Code ausgeführt wird?

```
Wenn Programm gestartet

Wiederhole bis Zähler = 2  wahr ist
        Ändere Punkte um 2
        Ändere Zähler um 1
Ende der Schleife
```

Die Variable Punkte hat den Wert: _____

Rätsel 5

Wir haben zwei Variablen gegeben und ein Objekt mit Namen Person:

```
a = 6
b = 3
```

Was wird unser Objekt Person machen wenn folgender Code ausgeführt wird?

```
Wenn a < b wahr ist, dann:
        Sprich „Hallo“
Sonst:
        Sprich „Mir geht’s gut
```

□ Das Objekt Person sagt: „ Hallo“ □ Das Objekt Person sagt: „ Mir geht’s gut“

Anhang 3: Feedback-Formular Schulklasse

Feedback

Wie hat dir diese Art des Lernens gefallen?

Ganz schlecht	Geht so	Gut	Sehr Gut

Wie hat dir diese Art des Programmierens gefallen?

Ganz schlecht	Geht so	Gut	Sehr Gut

Waren die Videos verständlich?

Nein, nicht sehr	Geht so	Ganz ok	Ja, sehr

Hast du zuhause Videos angeschaut oder programmiert?

Ja	Nein

Bist du weiterhin interessiert am Programmieren?

Nein, nicht sehr	Geht so	Ein bisschen	Ja, sehr

Was hat dir am besten gefallen?

Was hast du gelernt?

Was hat dir nicht so gefallen?