MASTERARBEIT | MASTER'S THESIS

Titel | Title

Entwicklung und Evaluation eines Methodenkoffers für den Programmiereinstieg: Ein Design-Based-Research Ansatz

verfasst von | submitted by

Maximilian Becker BEd

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

Master of Education (MEd)

Wien | Vienna, 2026

Studienkennzahl lt. Studienblatt | Degree programme code as it appears on the student record sheet:	UA 199 514 520 02
Studienrichtung lt. Studienblatt | Degree programme as it appears on the student record sheet:	Masterstudium Lehramt Sek (AB) Unterrichtsfach Informatik Unterrichtsfach Mathematik
Betreut von | Supervisor:	Univ.-Prof. Dipl.-Ing. Dr. Renate Motschnig

Abstract

Während die Programmierung zu den Kernthemen in der Informatik zählt, bringt der Einstieg in die Programmierung diverse Herausforderungen mit sich. Einerseits müssen den Lernenden logische Konzepte (z.B. Variablen, Schleifen, Algorithmen) nähergebracht werden. Andererseits bringen viele Schüler:innen unrealistische Vorstellungen (bspw. Spieleprogrammierung, Hacking) mit in den Unterricht. Daher ist ein realistischer, motivierender und festigender Einstieg in die Programmierung von großer Bedeutung für den Informatikunterricht.

Ziel dieser Arbeit ist es, einen Methodenkoffer zu entwickeln und zu evaluieren, der einen strukturierten Einstieg in die Programmierung als Unterrichtskonzept ermöglicht. Daraus ergibt sich die Forschungsfrage: "Inwiefern ist der Programmierung-Methodenkoffer für Schüler:innen der FMS geeignet, einen Einstieg in die Programmierung in einem Zeitraum von 32 Wochenstunden zu bieten?"

Zur Beantwortung dieser Frage wird der Forschungsansatz "Design Based Research" herangezogen. Im Rahmen des "ADDIE-Modell" wird dabei ein bereits bestehendes Unterrichtskonzept analysiert, überarbeitet, weiterentwickelt und später implementiert. Der erprobte Unterricht wird anschließend anhand von Testergebnissen, Gedächtnisprotokollen und Feedback evaluiert und optimiert. Somit kommt ein Methodenkoffer zustande, der in mehreren Entwicklungszyklen weiter ausgearbeitet wurde.

Die Ergebnisse zeigen, dass insbesondere die Ansätze des "Game-Based Learnings" eine motivierende Möglichkeit darstellen, Unterrichtsinhalte nachhaltig zu vermitteln. Der Einstieg in die Programmierung mittels "Pair Programming" führte ebenfalls zu einem stärkeren Erfolgsgefühl, da auch die Fehlerquote beim Programmieren niedriger ausfiel und die Lernenden nicht ausschließlich auf sich selbst angewiesen waren. Darüber hinaus erwies sich die Trennung von theoretischen Inhalten und praktischer Programmierung als sinnvoll, da sich die Lernenden dadurch besser auf den jeweiligen Themenbereich konzentrieren konnten.

Während der entwickelte Methodenkoffer aufgrund bestehender Limitationen der Forschung nicht als perfekte Lösung betrachtet werden kann, bieten die entstandenen Unterrichtsvorbereitungen dennoch eine fundierte Basis für den eigenen Unterricht und können entsprechend angepasst werden.

Inhaltsverzeichnis

1 Einleitung, Kontext

Der Einstieg in die Programmierung stellt sowohl Lernende als auch Lehrkräfte vor besondere Herausforderungen (Eckart Modrow und Kerstin Strecker 2016). Grundlegende, abstrakte Konzepte wie Variablen, Schleifen und Algorithmen erfordern ein Mindestmaß an logischem Denkvermögen und Kompetenzen zur Problemlösung. Gleichzeitig bringen viele Lernende unrealistische Vorstellungen von Programmierung in den Unterricht mit, welche häufig von populären Medien oder Computerspielen geprägt sind (Umfragen Schüler:innen 2020-2022). Diese Diskrepanz zwischen Erwartung und Realität führt des Öfteren zu Überforderung und Demotivation der Lernenden in den ersten Unterrichtseinheiten. Ein gelungener Einstieg in die Programmierung ist daher von zentraler Bedeutung für den weiteren Lernerfolg im Fach Informatik.

In den letzten Jahren entstanden zahlreiche Ansätze, um Lernenden den Einstieg in die Programmierung zu erleichtern. Dazu zählen visuelle Programmiersprachen, wie Scratch oder NEPO (Alexander Ruf u. a. 2014), handlungsorientierte Zugänge, wie Pair Programming (It-Agile GmbH, o. J.), sowie motivierende Lernformen aus dem Bereich des Game-Based Learning (Patrick Felicia 2014). Aufgrund dieser Vielfalt bleibt die Frage bestehen, welche Methoden und Inhalte sich in einem schulischen Kontext tatsächlich bewähren und wie sie ein Basiswissen zur Programmierung nachhaltig schaffen können. Besonders an berufsorientierten Schulen, an denen – wie die oben genannten Umfragen ergaben – Lernende sehr unterschiedliche Vorkenntnisse, Lernvoraussetzungen und sprachliche Kompetenzen mitbringen, zeigt sich ein hoher Bedarf an flexiblen und motivierenden Unterrichtskonzepten.

Um diesen Bedarf zu decken wurde im Rahmen dieser Masterarbeit ein neues Unterrichtskonzept entwickelt, welches den Einstieg in die Programmierung durch praxisnahe, motivierende und schüler:innenzentrierte Methoden verbessern soll. Grundlage bildet der Forschungsansatz des Design-Based Research (DBR), der sich durch einen engen Zusammenhang zwischen theoretischer Fundierung und praktischer Erprobung auszeichnet. Dieser Ansatz ermöglicht es, ein didaktisches Design in mehreren Zyklen zu konzipieren, praktisch umzusetzen und anhand von Rückmeldungen schrittweise zu optimieren. So werden nach dem ADDIE-Modell (Andrew DeBell 2020) verschiedene Phasen durchlaufen,

um den bisherigen Stand des Unterrichts zu analysieren, ein Konzept zu designen, dieses zu implementieren und im Anschluss zu evaluieren.

Für den Entwurf des Unterrichtskonzepts werden Daten aus eigenen Unterrichtsbeobachtungen, Gedächtnisprotokollen und Schüler:innenfeedbacks des Unterrichts von 2020 bis 2024 herangezogen, um Stärken und Schwächen des entwickelten Konzepts zu identifizieren und schrittweise zu optimieren.

Dabei wird sich an folgende Grundsätze gehalten: **Programmiergrundlagen** sollten spielerisch übermittelt und somit gefestigt werden. Diese werden außerdem mit einem Infotext und -bild versehen, welcher die Inhalte zusammenfasst. Weiters sollten **PC-Grundlagen** vermittelt werden, welche den Umgang mit einer Windows[1]-Umgebung erläutern und einen reibungslosen Ablauf hinsichtlich der Navigation in der Programmierumgebung ermöglichen. Die **Programmieraufgaben** sollten detaillierte und intuitive Anleitungen haben. Hinsichtlich des Inhalts sollte es eine spielerische Komponente geben, um die intrinsische Motivation zu fördern.

Somit ergibt sich folgende Forschungsfrage:

Inwiefern ist der Programmierung-Methodenkoffer für Schüler:innen der FMS geeignet, einen Einstieg in die Programmierung in einem Zeitraum von 32 Wochenstunden zu bieten?

Die Masterarbeit gliedert sich in einen theoretischen und einen empirischen Teil. Der theoretische Teil beleuchtet zentrale didaktische Ansätze des Programmierunterrichts und erläutert den methodischen Rahmen des Design-Based Research. Der empirische Teil beschreibt die Entwicklung, Durchführung und Evaluation des neuen Unterrichtskonzepts anhand der einzelnen ADDIE-Phasen (Analyse, Design, Development, Implementation, Evaluation) (Andrew DeBell 2020). Abschließend werden die Ergebnisse reflektiert und mögliche Implikationen für die Unterrichtspraxis diskutiert.

[1] Vgl. Betriebssystem

2 Theorie

2.1 Didaktische Ansätze zum Einstieg in die Programmierung

2.1.1 Herausforderungen

Die Literatur (Eckart Modrow und Kerstin Strecker 2016) setzt sich mit mehrjährigen Erfahrungen von Lehrpersonen und Professor:innen im Bereich der Informatik auseinander. Eingegangen wird unter anderem auf die Programmierumgebung. Ausschlaggebend sind etwa die zur Verfügung stehenden Befehlssätze - sowohl in einer visuellen (z.B. Scratch (Abb. 1) oder NEPO), als auch in einer codebasierten Programmiersprache (z.B. Python oder Java). Einerseits sollte die Programmierumgebung elementare Befehle bieten, um Zustände und Variablen zu verändern. Andererseits muss es auch Kontrollstrukturen geben, welche diese Befehle miteinander verknüpfen können - etwa in Schleifen oder Bedingungen.

In dieser Lektüre zeigt sich auch, dass diese elementaren Inhalte keine Schwierigkeit im Einstieg bieten. Schubladen werden als geeignetes Beispiel gebracht, um Variablen darzustellen. Dabei wird von den einfachen Variablentypen ausgegangen und Abstrakte Datentypen (ADTs) außen vorgelassen. Auch einfache Algorithmen werden gut verstanden. Hier wird etwa auf eine Tabelle gesetzt (Abb. 2), welche in jeder Codezeile den Zustand jeder Variable bestimmt. Lernende haben keine Schwierigkeiten, solch eine Tracetabelle korrekt auszufüllen.

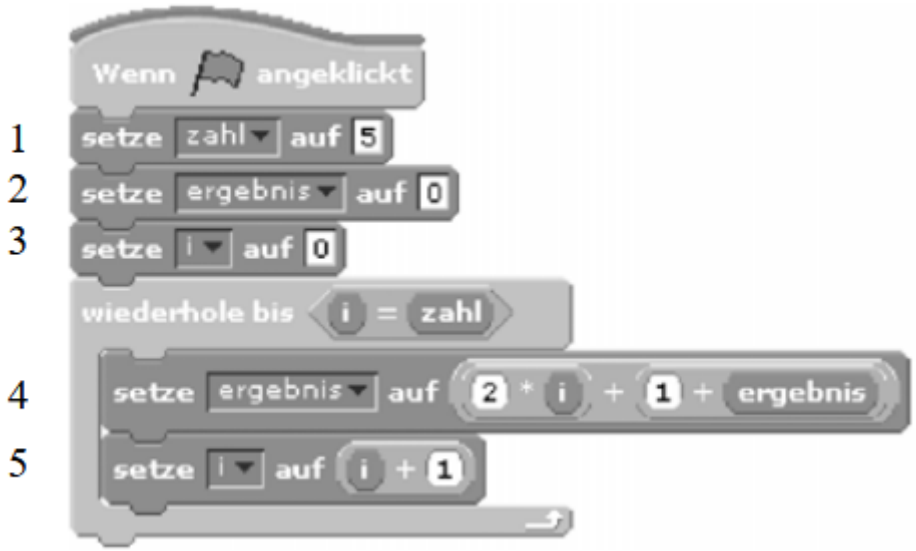


Abbildung 1: Algorithmus in Scratch (Modrow und Strecker 2016 S. 166)

Programmzeile	zahl	ergebnis	i
1	5	unbekannt	unbekannt
2	5	0	unbekannt
3	5	0	0
4	5	1	0
5	5	1	1
4	5	4	1
5	5	4	2
4	5	9	2
5	5	9	3
4	5	16	3
5	5	16	4
4	5	25	4
5	5	25	5

Abbildung 2: Algorithmustabelle (Modrow und Strecker 2016 S. 166)

Die ersten Probleme zeigen sich laut Modrow und Strecker, wenn Lernende komplexe Aufgaben selbstständig lösen müssen. Hier verlieren einige Schüler:innen nämlich den Anschluss. Sie haben zwar kein Problem, diese Algorithmen im realen Leben anzuwenden (etwa die schriftliche Addition oder Verschlüsselung per Vigenére). Doch diese Abläufe in eine Programmiersprache zu übersetzen, bringt Schwierigkeiten mit sich. Anstatt den bekannten Elementarbefehlen muss nun nämlich eine Komposition aus vielen dieser kleineren Befehle entstehen. Hier macht es Sinn, das Programm in kleine Einzelprobleme zu zerlegen und diese einzelnen Aufgaben auch als eigene Befehle oder Methoden zu schreiben. So verliert man den Überblick nicht, das eigentliche Ziel sollte nämlich in Sicht bleiben.

Die gegebene Aufgabe sollte außerdem einen Startzustand und einen Endzustand bestimmen. Der von den Lernenden zu programmierende Code sollte dann verschiedene Startzustände in den gewünschten Endzustand bringen können. Welche Befehle könnten also genutzt und kombiniert werden, um diesen Endzustand zu erreichen? Hier muss ein abstrakter Denkvorgang implementiert werden.

2.1.2 Visuelle Programmiersprachen

Bereits einige Studien widmeten sich dem Einsatz von visuellen Programmiersprachen im Unterricht. So wurden etwa zwei verschiedene Gruppen von Schüler:innen untersucht, welche Programmierkurse in Vorbereitung auf das Abitur in Informatik besuchten. In einem Vergleich wurde festgestellt, dass jene Schüler:innen, welche sich mit Snap! (vgl. visuelle Programmiersprache) auseinandersetzten, durchschnittlich bessere Leistungen erbrachten. Diese Gruppe hatte die Konzepte und Kompetenzen (v.a. bezüglich Algorithmik) besser erlernt und konnten auch Programmieraufgaben schneller lösen, als jene Gruppe, welche den Vorbereitungskurs in Java besuchten (Ralf Romeike und Sven Jatzlau 2017).

In einer weiteren Studie (Alexander Ruf u. a. 2014) wurde der Einsatz von Scratch (vgl. visuelle Programmiersprache) und Karol (vgl. textbasierte Programmiersprache) im Bezug auf die intrinsische Motivation und Programmierkenntnisse verglichen. In beiden Hinsichten konnte die Klasse, welche Scratch als Programmiersprache einsetzte, besser abschneiden[2].

2.1.3 Textbasierte Programmiersprachen

Studien zeigen, dass Einsteiger:innen in die Programmierung durch textbasierte Sprachen mit höheren kognitiven Anforderungen konfrontiert sind. Sowohl die Syntax, als auch die Strukturen spielen dabei eine große Rolle. Gomez, Moresi und Benotti untersuchten Schüler:innen zweier Schulen - die Lernenden einer Schule brachten bereits Erfahrung zu visuellen Programmiersprachen mit sich, diejenigen der anderen Schule nicht. Während hinsichtlich der Semantik die Schüler:innen mit Vorerfahrung deutlich besser abschneiden konnten, wurde in Bezug auf die Syntax der Programmiersprache kein Unterschied festgestellt (Marcos J. Gomez u. a. 2019).

[2] Motivation: Höhere Werte in allen Kategorien (interest/enjoyment, perceived competence, perceived choice, pressure/tension) bei Lernenden der Scratch-Klasse mit $p = 0{,}047$. Programmierkenntnisse: Höhere Durchschnittswerte sowohl bei Pre-Test als auch bei Post-Test mit $p < 0{,}004$.

Trotz bewiesener Vorteile visueller Programmierung wird in jeglichen Bildungsbereichen ein textbasierter Einstieg favorisiert (Get IT, o. J.; Programmieren lernen 2022). Insbesondere wird Python häufig für den Einstieg empfohlen und gilt laut einer aktuellen Statistik als eine der verbreitetesten Programmiersprachen (Statista, o. J.).

Neben der Sprache selbst gilt auch die Art der Programmierung als wichtiger Faktor in der Wissensvermittlung. Lernenden mit wenig Erfahrung oder geringem Selbstvertrauen in die eigenen Fähigkeiten wird der Bottom-Up Ansatz empfohlen. Bei diesem wird erst ein einzelnes Modul entwickelt und dieses dann erweitert, um eine Gesamtlösung zu entwickeln („Was versteht man unter Top-Down oder Bottom-Up Verfahren?", o. J.). Dies reduziert die anfängliche kognitive Belastung und erleichtert den Einstieg in die abstrakteren Inhalte. Im Gegensatz dazu befasst sich der Top-Down-Ansatz zuerst mit der Entwicklung eines Gesamtkonzepts und erst im Anschluss mit den einzelnen Programmteilen. (Eckart Modrow und Kerstin Strecker 2016)

Somit zeigt sich, dass textbasierte Programmiersprachen fachlich relevant sind. Jedoch sollte der Einstieg im didaktischen Sinne besonders sorgfältig geplant werden, um eine Überforderung und den daraus resultierenden Motivationsverlust zu verhindern.

2.1.4 Pair Programming als kollaborative Lernform

Pair Programming (It-Agile GmbH, o. J.) beschreibt die Aufteilung des Coding in zwei grundlegende Rollen: Pilot und Navigator. Der Pilot ist für die Programmierung des Codes verantwortlich. Es ist die Rolle, welche als einzige den Code verändern darf. Der Navigator überwacht dabei die „Korrektheit des Codes" und ist auch für die Designgebung verantwortlich. Diese beiden Rollen sind sich im Projekt ebenbürtig werden regelmäßig getauscht.

Einige Studien unterstreichen den qualitativen Effekt von Pair Programming im Unterricht (Brian Hanks u. a. 2011; Jo E. Hannay u. a. 2009). In einer Studie (Patrick Korber und Renate Motschnig 2021) wurde mithilfe qualitativer Forschungsmethoden[3] untersucht, inwiefern sich die Methode des Pair Programming auf die Motivation und Fähigkeiten der Lernenden auswirkt. In einem kleinen Umfang von 14 Beteiligten konnte eine bessere Punktezahl bei entsprechenden Tests erreicht werden, wenn die Schüler:innen in Paaren programmierten, als in einem alleinigen Versuch. In offenen Fragen machten die Lernenden auf die

[3] Systematic Literature Review, Questionaires, Focus Inverviews

zusätzliche Motivation aufmerksam, die bei einer Zusammenarbeit erfolgte. „Ich hätte ohne meinen Partner viel länger gebraucht. [...] Es war eher lustig anstatt deprimierend, wenn wir beide keine Ahnung hatten, wo wir im Code sind und was das Problem sein könnte." (Patrick Korber und Renate Motschnig 2021).

Allerdings könnte es in solch einem sozialen Akt auch zu Konflikten kommen. Einerseits könnte die Arbeitsteilung unausgewogen sein, andererseits würden die verschiedenen Visionen der Beteiligten zu Problemen führen. Dabei gilt zu erwähnen, dass sich besonders diejenigen Schüler:innen, welche bessere Leistungen in der Programmierung zeigen konnten, dieser Methode besonders kritisch gegenüberstanden. Korber konnte außerdem feststellen, dass in einer textbasierten Programmiersprache die Vorteile des Pair Programming besonders herausstechen würden (Patrick Korber und Renate Motschnig 2021).

2.1.5 Motivationsorientierte Ansätze

Da der Einstieg in die Programmierung häufig mit Frustrationerfahrungen verbunden ist, wird „Game-Based Learning" als didaktischer Ansatz diskutiert, um Inhalte in einem motivierenden und spielerischen Kontext zu vermitteln (Patrick Felicia 2014; Son Le und Peter Weber 2011). Game-Based Learning bezeichnet dabei den Einsatz vollständiger Lernspiele, bei denen fachliche Inhalte integraler Bestandteil der Spielmechanik sind.

Dieser Ansatz erlaubt es, Strategien zu entwickeln und logische Denkprozesse zu fördern, ohne dass der theoretische Inhalt im Vordergrund steht. Gleichzeitig ermöglicht der spielerische Rahmen, eine stärkere emotionale Einbindung der Lernenden, wodurch die intrinsische Motivation sowie die Bereitschaft zur weiteren Auseinandersetzung mit dem Thema gesteigert werden können. *„By virtue of their ability to actively engage, and stimulate players to retain the information presented to them, Serious Games offer an alternative way for educators to present material to learners."* (Felicia 2014, S. 8)

Autor:innen weisen darauf hin, dass Game-Based Learning eine sorgfältige didaktische Planung erfordert, um eine Überlagerung der Lernziele durch spielerische Elemente zu vermeiden.

2.1.6 Der grundlegende Einstieg

Zusammenfassend zeigen die in Kapitel 2.1 dargestellten Ansätze, dass der Einstieg in die Programmierung einer gezielten didaktischen Gestaltung bedarf. Insbesondere der

Übergang von abstrakten theoretischen Konzepten zur praktischen Umsetzung in einer Programmiersprache stellt für viele Lernende eine zentrale Herausforderung dar.

Motivationsorientierte Zugänge wie Game-Based Learning können dazu beitragen, diesen Übergang zu erleichtern, indem Programmierinhalte in einen sinnstiftenden und spielerischen Kontext eingebettet werden. Dadurch kann die intrinsische Motivation der Lernenden gefördert und die Bereitschaft zur aktiven Auseinandersetzung mit programmierbezogenen Problemstellungen erhöht werden.

Darüber hinaus kommt der Wahl der Programmiersprache im Einstieg eine entscheidende Bedeutung zu. Für Anfänger:innen sollte zunächst das Verständnis algorithmischer Abläufe im Vordergrund stehen, während syntaktische Anforderungen möglichst reduziert werden. In diesem Zusammenhang bieten visuelle Programmiersprachen geeignete Voraussetzungen für einen niederschwelligen Einstieg.

Auch die gewählte Lernform beeinflusst den Einstieg in die Programmierung maßgeblich. Kollaborative Ansätze wie Pair Programming ermöglichen es Lernenden, Programmieraufgaben gemeinsam zu bearbeiten, voneinander zu lernen und Unsicherheiten im Lernprozess abzufedern.

Weiters existieren noch zahlreiche andere didaktische Ansätze, mit welchen man den Unterricht gestalten könnte. Spieleentwicklung, wie später in den Umfrageergebnissen zu lesen ist, stellt einen großen Motivator der Lernenden dar, kann aber ohne Vorwissen zu einer starken Komplexität heranwachsen (Till Sobioch 2024). Der Einsatz von Einplatinenrechnern oder Robotics wäre ebenfalls möglich, insofern diese an der Schule vorhanden sind. Allerdings soll das Unterrichtskonzept eine mögliche Basis für einige verschiedene Schulen bieten, somit wird von einem Minimum an Ressourcen ausgegangen.

Insgesamt lassen sich aus den dargestellten Ansätzen zentrale didaktische Prinzipien ableiten, die für einen erfolgreichen Einstieg in die Programmierung relevant sind. Diese bilden die Grundlage für das im weiteren Verlauf der Arbeit entwickelte und erprobte Unterrichtskonzept.

3 Methodik und Forschungsdesign

Um das Unterrichtskonzept zu entwerfen, testen und zu verbessern, und somit die Forschungsfrage zu beantworten, wird als methodischer Rahmen „Design Based Research" (DBR) eingesetzt. Dieses bietet sich durch die Verknüpfung von Entwicklung der Unterrichtsmaterialien mit der Gewinnung wissenschaftlicher Erkenntnisse als eine sinnvolle Forschungsmethode an. Mithilfe des ADDIE-Modells werden die Inhalte zyklisch analysiert, neu designt und entwickelt und somit verbessert.

3.1 Design Based Research

DBR ist ein entwicklungsorientierter Forschungsansatz, der wissenschaftliche Erkenntnisgewinnung und die Lösung konkreter Bildungsprobleme im Sinne einer gestaltungsorientierten Praxisforschung miteinander verbindet. Im Unterschied zu anderen Forschungsmethoden, welche bestehende Praxis in erster Linie evaluieren, verfolgt DBR das Ziel, durch einen iterativen Gestaltungsprozess innovative Lösungen zu entwickeln und dabei gleichzeitig wissenschaftlich fundiertes Wissen zu generieren.

DBR entstand dabei als Antwort auf die Kritik, dass traditionelle empirische Bildungsforschung zu praxisfern sei. „Design research emerged mainly in response to criticism of the lack of practical application of empirically and analytically orientated teaching and learning research's findings." (Dieter Euler und Peter Sloane F. E. 2014). Stattdessen strebt DBR einen synergetischen Dialog zwischen Forschung und Unterrichtspraxis an - eine Balance zwischen dem konzeptionellen „Erfinder" und dem empirischen „Detektiv", die gemeinsam innovative Lernlösungen entwickeln. (Andrea Burda-Zoyke 2017)

Euler (Dieter Euler und Peter Sloane F. E. 2014) beschreibt die zentralen Merkmale von DBR wie folgt:

- **Zentral leitende Frage (key question):** Im Zentrum steht nicht die Frage, *ob* eine Intervention wirkt, sondern *wie* sie so gestaltet werden kann, dass sie das angestrebte Ziel im gegebenen Kontext bestmöglich erreicht. *„How should an intervention be formulated within the context of a teaching concept (method) to advance specific (concretely recognizable) social competencies (objectives)?"* Im Fall

dieser Arbeit: Wie kann ich den Einstieg in die Programmierung didaktisch besonders motivierend, ansprechend und nachhaltig gestalten?

- **Entdecken, Entwickeln und Erproben innovativer Lösungen für ungelöste Probleme der Praxis *(discovering, developing and testing innovative solutions for unsolved practical problems)*:** Finde ein Problem, dessen Lösung einen innovativen Ansatz benötigt. Dabei geht es nicht darum, jene Realitäten zu beforschen, welche schon existieren, sondern jene welche sein könnten. In etwa: Warum gestaltet sich der Einstieg generell schwer? Was könnte anders gemacht werden, um den Einstieg leichter und ansprechender zu gestalten?
- **Theorie-basierte Entwicklung *(theory-based development)*:** Die Entwicklung von innovativen Lösungen gestützt durch bereits existierende, wissenschaftliche Belege, aber auch durch die Theorie von erfahrenen Fachleuten. Wie bereits im Kapitel 2.1 gezeigt, gibt es bereits Ansätze, welche auch sinnvoll Anwendung finden. Welche davon finden Platz im „Methodenkoffer" dieser Arbeit? Welche Methoden finden Kolleg:innen in der Informatik sinnvoll?
- **Starker Praxisbezug durch iterative Design-Zyklen *(high practical relevance by means of iterative design cycles)*:** Die innovative Idee wird durch mehrere Design-Zyklen verbessert. Dabei wird nicht jede kleine Idee getestet, sondern Zyklen auf Basis der besten Ideen weiterentwickelt und erprobt. Die vielversprechendsten Designs werden im Anschluss korrigiert. Welche bereits getestete Unterrichtsmethoden können auf welche Art verbessert werden?
- **Zusammenarbeit zwischen Forschung und Praxis (cooperation between science and practice):** In den unterschiedlichen Phasen der Design-Entwicklung werden auch erfahrene Fachleute involviert. So sollten auch andere Zugänge als die des Forschers, Einfluss nehmen. So erwartet man eine realistischere praktische Anwendung. Im Rahmen dieser Arbeit wird dies durch den Einbezug von Kolleg:innen und die Analyse in Seminaren umgesetzt.
- **Gezielte Theorien als Ergebnisse (area-specific theories as targeted results):** Das Design sollte nicht nur für den spezifischen, beforschten Kontext ein Ergebnis liefern, sondern auch im breiteren Spektrum anwendbar sein. So sollte man die Ergebnisse nicht als „technologische Anleitung", sondern als *„design principle"* formulieren.

Ist der „Methodenkoffer" also vielseitig anwendbar? Kann man einzelne Zugänge auch in anderen Themen wiederverwenden? Die Ergebnisse dürfen keineswegs als Allheilmittel für den Informatikunterricht verstanden werden, vielmehr als mögliche Lernpfade, die als Basis für den eigenen Unterricht dienen können.

3.1.1 ADDIE Modell

Während Dieter Euler ein gängiges Phasenmodell (Dieter Euler 2014, S. 69) beschreibt, in welcher Ideen gefunden, selektiert und anschließend realisiert werden, besteht im Fall dieser Arbeit bereits Unterrichtsmaterial, welches verbessert werden sollte. Somit wird sich im Laufe der Forschung an das „*ADDIE Modell*" („ADDIE Model", o. J.) gehalten. Dieses beschreibt eine zyklische Herangehensweise in 5 Stufen (Abb. 3).

1. **Analysis**: Es wird vom „Detektiv" die aktuelle Situation analysiert, um herauszufinden, welche Lücken gefüllt werden können. Dabei sollte man sich grundlegende Fragen stellen, etwa „wer die Lernenden sind, welche Ziele man erfüllen möchte und welche Ressourcen zur Verfügung stehen". In dieser Arbeit werden dazu die bisherigen Unterrichtseinheiten mithilfe von Gedächtnisprotokollen und Feedbackbögen reflektiert.
2. **Design**: Auf die Analyse wird nun vom „Erfinder" das bestmögliche Design aufgebaut, um die gefundenen Lücken zu füllen. Dabei geht man möglichst „systematisch und spezifisch" vor. Es werden also Unterrichtseinheiten neu geplant mit genau definierten Zielen, Rahmen und Inhalten.
3. **Development**: In dieser Phase arbeiten der „Detektiv" und der „Erfinder" gemeinsam am inhaltlichen Teil und setzen das Design konkret um. „Der Rahmen wird gefüllt".
4. **Implementation**: Die neu entwickelten Unterrichtseinheiten werden fertig aufbereitet und getestet. Dabei spielt nicht nur die Instruktion, sondern auch das Testen eine große Rolle, um den finalen Schritt umsetzen zu können. Im Zuge der Forschung wird dies im Rahmen eines Unterrichts mit einer Gruppe von 10 Schüler:innen an einer Polytechnischen Schule geschehen.
5. **Evaluation**: Schließlich werden die neu entwickelten Inhalte anhand der Implementierung analysiert. Dabei können zum Beispiel Testergebnisse, Gedächtnisprotokolle oder Feedbacks zum Einsatz kommen. Da DBR ein iterativer Prozess ist, stellt die Evaluation keinen finalen Schritt dar, sondern den Abschluss

eines ersten Entwicklungszyklus. Anhand dieser Evaluation könnte man bei Schritt 2 - Design wieder neu ansetzen.

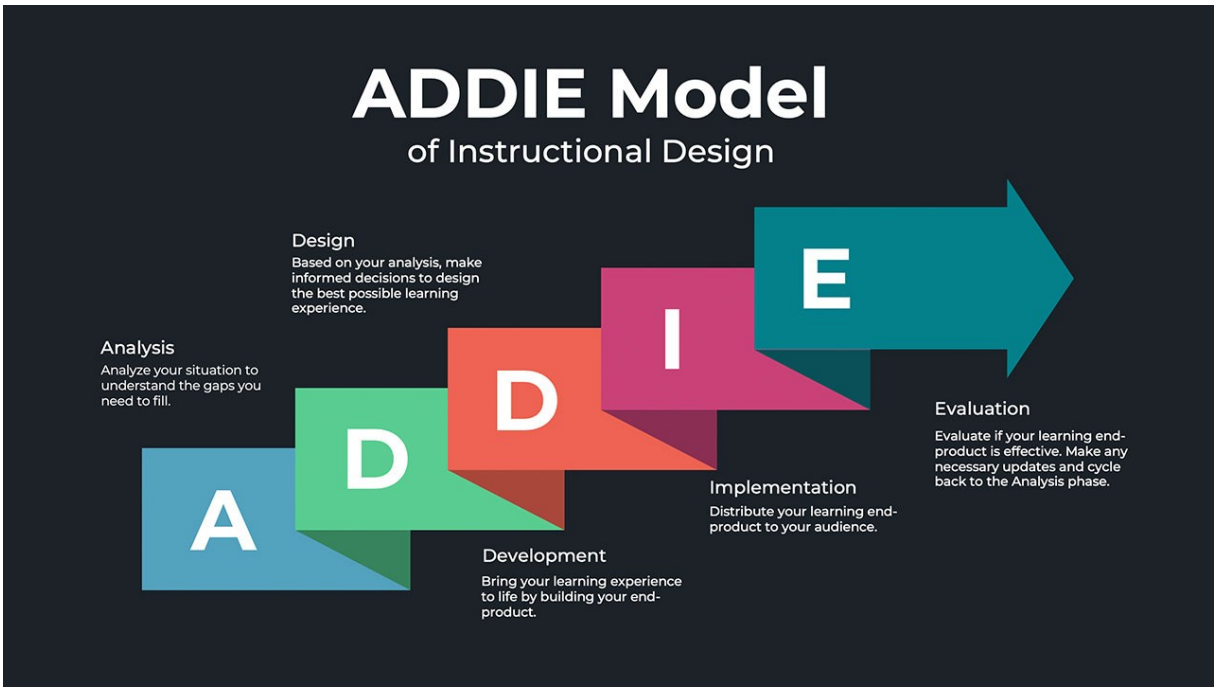


Abbildung 3: (Andrew DeBell 2020)

4 Reflexion der bisherigen Unterrichtseinheiten

Es folgt die erste Phase des ADDIE-Modells: „Analysis". Es werden diejenigen Unterrichtseinheiten, welche in den Schuljahren 2020/21 bis 2023/24 verwendet wurden mithilfe jeglicher Gedächtnisprotokolle sowie den von Schüler:innen ausgefüllten Feedbackbögen aus den Schuljahren 2022/23 und 2023/24 analysiert. Jegliche Dokumente - Präsentationen, Angaben, Tests, Feedbackbögen - können im Anhang gefunden werden.

4.1 Bisherige Unterrichtseinheiten

4.1.1 Einführung

In den Jahren 2022/23 bis 2023/24 wurde eine Präsentation gehalten, welche die grundlegenden Inhalte der Programmierung näherbringen sollten: Struktur, Variablen, Funktionen, Konditionen und Schleifen, Befehle. Dabei verwendete ich die Programmiersprache *Java*, um den jeweiligen Code zu erklären.

Bereits hier werden die Schüler:innen „ins kalte Wasser geworfen". Auf Folie 3 (Abb. 4) sehen sie die ersten Codezeilen, um die Struktur zu verstehen. Dabei sind einige Befehle noch komplettes Neuland. Diese Überwältigung führte schon bald zu Unverständnis und schließlich Demotivation.

```java
public class MainClass {
    // Variablen definieren
    int anzahl = 5;
    float pi = 3.14;
    String message = "Hallo Welt";
    char atsign = '@';

    public static void main(String[] args){
        // MAIN FUNKTION

        // Aufrufen einer anderer Funktion
        output();

        // Aufrufen einer anderen Funktion mit Argumenten
        output(7);

    }

    void output () {
        // Schleife, die mitzählt
        for (int i = 1; i <= anzahl; i++) {
            // Wird 5 mal ausgeführt, weil anzahl = 5

            // Message in Konsole ausgeben
            System.out.println(i + message);
        }
    }

    void output (int anzahl2) {
        // Schleife, die mitzählt
        for (int i = 1; i <= anzahl2; i++) {
            // Wird 7 mal ausgeführt, weil anzahl2 = 7

            // Message in Konsole ausgeben
            System.out.println(i + message);
        }
    }
```

Abbildung 4: Code aus Folie 3

Auch die anschließenden Inhalte werden zu schnell überflogen - insgesamt dauerte die Präsentation durchschnittlich 50 Minuten. Es werden die Definitionen von Variablen, der Aufbau von Funktionen mit Parametern und return-Statements, verschiedenste Operatoren, Schleifen und Konditionen innerhalb kürzester Zeit erläutert und treffen auf wenig Verständnis.

Begleitend zur Präsentation erhielten die Lernenden ein PDF, welches die Inhalte der Präsentation verschriftlichte, sowie ein digitales Vokabelheft, das von den Lernenden als Übung ausgefüllt und ergänzt werden sollte.

4.1.2 Scratch

Ausschließlich im ersten Unterrichtsjahr 2020/21 wurde den Schüler:innen ein Einstieg in einer visuellen Programmiersprache dargelegt. In diesem Jahr musste aufgrund der Corona-Maßnahmen die Hälfte der Gruppe Aufgaben von zuhause aus erledigen. Die andere Hälfte durfte dem Unterricht in Anwesenheit beiwohnen. Für die unterrichtsferne Gruppe wurde dabei ein kleines Aufgabenblatt für Scratch zusammengestellt. Weitaus mehr Zeit konnte aufgrund der zeitlichen Gegebenheiten nicht in den Programmierunterricht investiert werden.

4.1.3 Processing

In den zwei Jahren darauf (2021/22 bis 2022/23) wurde für ein erstes Beispiel die Programmiersprache *Processing (vgl. textbasierte Programmiersprache)* verwendet. Diese bietet simplere Befehle als Java und kann mittels der „void draw"-Methode auf einfachere Art und Weise einen grafischen Output generieren.

In der Angabe finden die Schüler:innen eine fünf Seiten lange Übersicht an Befehlen inklusive zahlreicher Beispiele zur Anwendung. Mit dieser Hilfestellung und einer Schritt-für-Schritt-Anleitung sollen sie nun ihr erstes, eigenes Programm schreiben: Ein schwarzes Fenster, welches ausgehend vom Mauszeiger bunte „Sterne" erzeugt (Abb. 5). Bei einem Mausklick soll eine Momentaufnahme des Fensters abgespeichert werden.

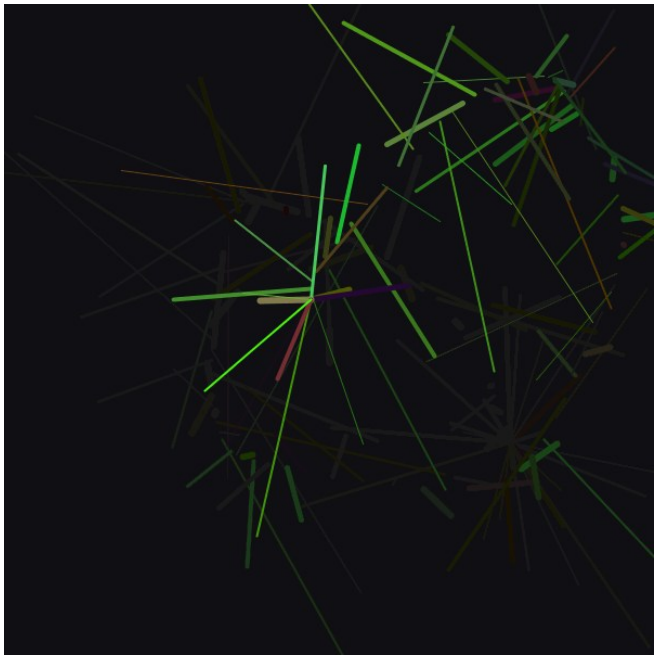


Abbildung 5: Screenshot "Stern"

Reflexion der bisherigen Unterrichtseinheiten

Als zweite Aufgabe sollen die Lernenden das Spiel *Pong* programmieren. Dabei kann auf der jeweils linken und rechten Seite ein Paddel in vertikaler Richtung gesteuert werden, welches dem jeweils anderen einen Ball zuspielt, bis ein Spieler ein Tor erzielt. In unserem Fall wird das Spiel nur von einem einzelnen Benutzer ausgeführt, welcher gegen sich selbst spielt (Abb. 6). Dafür werden ganze Codeteile zur Verfügung gestellt - schriftlich, aber noch nicht vorprogrammiert.

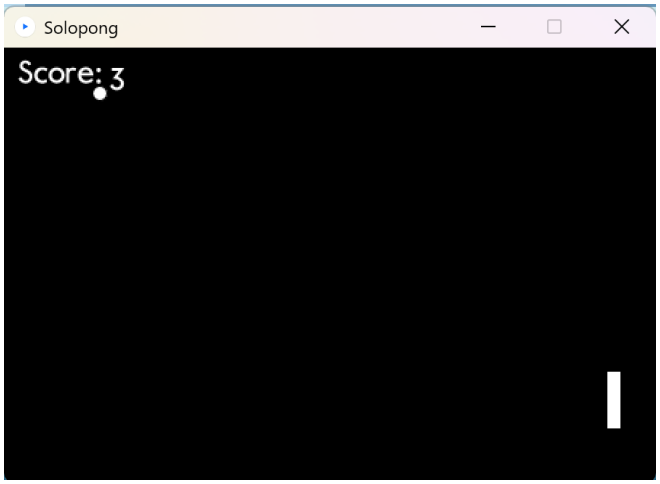


Abbildung 6: Screenshot "Pong"

Das Ziel dieses ersten Unterrichtstages war es, die Lernenden mit den Grundlagen der Programmierung vertraut zu machen und sie bereits die ersten Programmteile selbstständig entwickeln zu lassen. Dabei sollte ein Erfolgsgefühl vermittelt werden, gestützt mit Materialien, welche sie auch in Zukunft als Referenz verwenden dürften.

Abschließend wurde ein Test durchgeführt, um die Selbstständigkeit der Lernenden zu überprüfen. Dabei wurde an das Programm *Sterne* angelehnt ein Beispiel gegeben und sollte von Grund auf programmiert werden. Bisherige Codes und Materialien durften dabei verwendet werden.

Die Lernenden waren vom schieren inhaltlichen Umfang derart überwältigt, sodass die meisten Codezeilen Schritt für Schritt vom Lehrenden vorprogrammiert wurden. Dies führte zu einer derartigen Frustration beider Seiten, sodass die Lernenden möglichst schnell mit dem Thema abschließen wollten.

Da es außerdem doch grobe Unterschiede zwischen Processing und Java gibt, wurde aufgrund der zusätzlichen inhaltlichen Belastung 2023/24 auf den Unterricht von Processing gänzlich verzichtet.

4.1.4 Java

In anschließenden Einheiten wurde ein Übergang von Processing zu Java angestrebt. Dabei wurde jedes Jahr auf ein neues mathematisches Beispiel eingegangen.

2021/22 sollten zum Einstieg die Methoden *add*, *multiply*, und *connect* geschrieben werden, welche zwei Zahlen addiert bzw. multipliziert. *connect* hingegen verknüpft zwei Zeichenketten miteinander. Anschließend wurde gemeinsam die Klasse *Bruch* definiert. Inhaltlich wurde hier bereits ein Konstruktor verwendet, welcher eine *Exception* wirft, sollte der Nenner 0 sein. Es wurden auch drei Methoden definiert, welche einerseits den Dezimalwert des Bruchs oder den Bruch als Zeichenkette (z.B. „3 / 4“) zurückgibt, aber auch den Bruch kürzen kann.

2022/23 wurden die 4 Grundrechnungsarten programmiert, indem man mittels Befehlen („/+“, „/-„, „/*“, „/:“) eine Rechenart auswählen und anschließend die zwei zu verknüpfenden Zahlen eingeben kann.

In den Jahren 2021 bis 2023 wurde der sogenannte „Magic 8 Ball“ programmiert. Dieser war ein beliebtes Spielzeug, welches in den 50er Jahren produziert wurde. Dabei handelt es sich um eine Kugel, welcher man eine Frage stellte. Anschließend wurde eine bejahende, verneinende oder weiterführende Antwort gegeben (Abb. 7). Im Unterricht sollte dem Programm über die Konsole eine Frage eingegeben werden, daraufhin wird eine zufallsgenerierte Antwort aus einem Array ausgegeben.

Abbildung 7: Magic 8 Ball („Where Did the Idea for the Magic 8 Ball Come From?“, o. J.)

Reflexion der bisherigen Unterrichtseinheiten

2023/24 wurde vorerst auf den mathematischen Einstieg verzichtet. Da in diesem Jahr noch keine Vorerfahrung durch Processing zustande kam, wurde erst auf die Objektorientierung eingegangen. Mit neuen inhaltlichen Materialien und einem kurzen Einstieg in die Modellierung sollten Klassen angelegt werden, welche über Ein- und Ausgabefunktionen mittels Konsole funktionieren (Abb. 8).

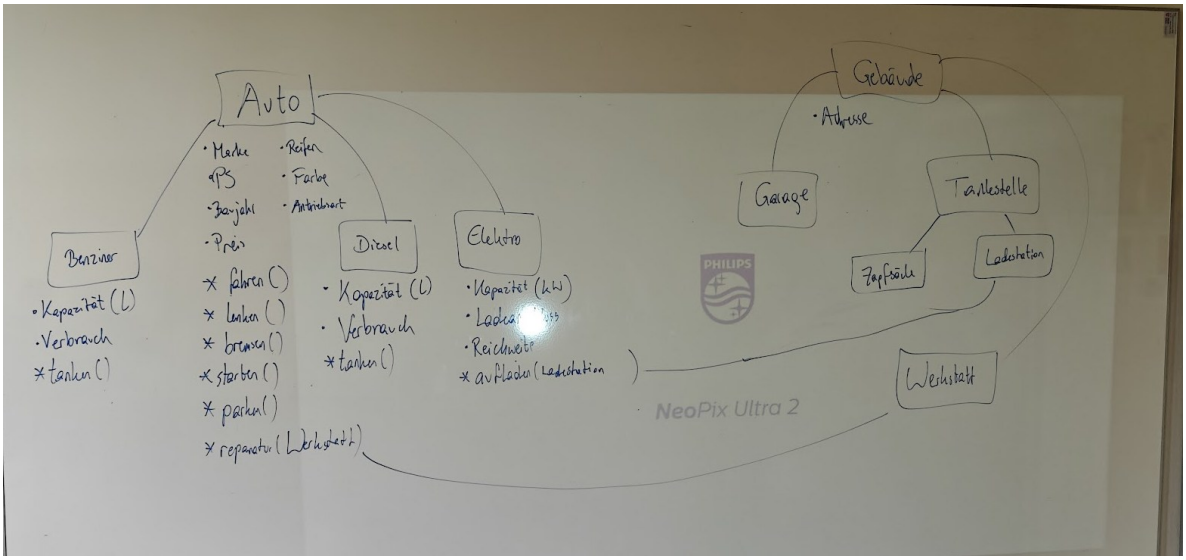


Abbildung 8: Diagramm zur Modellierung der verschiedenen Klassen

Der Code für die Klasse „Auto“ könnte etwa folgendermaßen aussehen.

```
public class Auto {
      // Variablen
      int ps, baujahr, preis;
      String reifen, antriebsart, farbe, marke;
      boolean gestartet = false;
      int speed = 0;
      Garage g;

      final String PROBLEMTEXT = "Der " + marke + " ist noch nicht
gestartet";

      // Konstruktor
      public Auto (int ps, int baujahr, int preis,
                  String reifen, String antriebsart, String farbe,
String marke) {
            this.ps = ps;
            this.baujahr = baujahr;
            this.preis = preis;
            this.reifen = reifen;
            this.antriebsart = antriebsart;
            this.farbe = farbe;
            this.marke = marke;
      }

      // Methoden
```

```
     public void fahren (int speed) {
          if (gestartet) {
               // Speed Wert festlegen
               this.speed = speed;
               System.out.println("Der " + this.marke + " fährt mit
" + speed + "km/h!");
          } else {
               System.out.println(PROBLEMTEXT);
          }
     }
     public void bremsen () {
          if (gestartet) {
               this.speed = 0;
               System.out.println("Der " + this.marke + "
bremst!");
          }
     }

     public void lenken (String s) {
          if (gestartet) {
               if (s != "") {
                    System.out.println("Der " + this.marke + "
lenkt nach " + s + "!");
               } else {
                    System.out.println("Der " + this.marke + "
lenkt geradeaus!");
               }
          }
     }
     public void starten () {
          gestartet = true;
          System.out.println("Der " + this.marke + " startet!");
          if (g != null) { // Wenn die Garage g nicht leer ist
               g.ausparken();
               g = null;
          }
     }

     public void parken () {
          gestartet = false;
          System.out.println("Der " + this.marke + " parkt!");
     }

     public void parken (Garage g) {
          if (g.parken()) { // Ist in Garage noch Platz?
               gestartet = false;
               this.g = g; // Garage g speichern
               System.out.println("Der " + this.marke + " parkt in
Garage " + g.getName() + "!");
          }
     }


}
```

Nachdem Java eine Objektorientierte Programmiersprache ist und für einen „Bottom-up-Ansatz“ schlecht geeignet ist, fällt es schwer, einfache Programme möglichst simpel zu formulieren. Auch die verwendete Programmierumgebung „eclipse“ kann durch die Vielzahl an Funktionen überfordernd wirken. Die Navigation durch die einzelnen Projekte, das Erstellen von Klassen und das Ausführen eines Programms stellten unnötige Hindernisse dar und lenkten vom eigentlichen Inhalt ab: Dem Programmieren. Somit musste man etwa extra eine Klasse mit einer „public void main“-Methode erstellen, um die gewünschten Objekte zu erstellen und Methoden auszuführen. In Zukunft sollte hier eine einfachere Sprache mit einer simpleren Programmierumgebung gewählt werden.

4.1.5 Test

In den Jahren 2021/22 und 2022/23 wurde zum Abschluss ein Test abgehalten. Dafür wurden alle bisherigen Materialien und Lösungen bereitgestellt. Sollte ein Kind beim ersten Test nicht anwesend sein, so konnten sie dies mittels eines zweiten Tests nachholen.

Im ersten Test mussten die Lernenden eine Methode erstellen, welche alle Primzahlen bis zum eingegebenen Parameter *n* ausgibt. Die Angabe beinhaltet eine Definition zu Primzahlen sowie eine Erläuterung des Modulo-Operators (%). Dieser gibt immer den Rest einer Division aus (beispielsweise 7%5 ergibt 2). Weiters wurde jeder Schritt von der Erstellung des Projekts bis zum Schreiben der Methode genau angeleitet und mit Beispielen untermauert.

Der zweite Test beinhaltet analog zum ersten die Aufgabe, alle Teiler einer Zahl *n* auszugeben. Auch hier wurde inhaltlich die Definition eines Teilers und Modulo-Operators erläutert. Die Angaben erfolgten in ähnlicher Weise.

2023/24 wurde kein Test durchgeführt, da die Zeit dafür nicht ausreichte. Daher wurden ausschließlich die Abgaben zur Bewertung herangezogen.

4.2 Feedback der Schüler:innen

Über Feedbackbögen wurden immer wieder Rückmeldungen eingeholt. Diese fielen durchwegs negativ aus. Auf die Frage, welche Inhalte einem am besten gefallen, folgten zwischen Antworten wie „Processing Übung Name“ und „Programmierung“ auch die Rückmeldungen „nix“ oder „alles was mein Kopf kaputt gemacht“. Eine Schülerin antwortete auch „Alles, aber brauche meine Ruhe um es zu verstehen und es zu mögen“.

Reflexion der bisherigen Unterrichtseinheiten

Der Frage „Welche Inhalte hast du nicht so gut verstanden?“ folgten die Antworten „eclipse“, „Programmierung Java“ oder schlicht „alles“. Auch Punktebewertungen bezüglich des Verständnisses der Inhalte und der möglichen Wiedergabe in einer folgenden Stunde untermauerten diese Rückmeldungen.

Schließlich wurden die Lernenden befragt, an welchem Thema sie die folgende Woche arbeiten möchten. Rückmeldungen wie „Java“, „Programmierung“ oder auch „die letzten Themen die ich nicht verstanden habe“ deuten auf eine Lernbereitschaft hin.

5 Erster Methodenkoffer

Auf Basis der bisher erlangten Kenntnisse, Feedback von Schüler:innen, eigenen Gedächtnisprotokollen, sowie Feedback von Kommiliton:innen wurde der zweite und dritte Schritt des ADDIE-Modells umgesetzt: „Design" und „Development". Dabei wurde von minimalen Ressourcen ausgegangen: PCs mit Editor, Browser und Internetzugang sowie ein vorhandenes Kommunikationstool zwischen Schüler:innen und Lehrperson. Die Zielgruppe änderte sich hierbei nicht, da das finale Unterrichtskonzept am selben Schulstandort ausgetestet wird. Allerdings könnte dieses auf weitere Schultypen und Altersgruppen angepasst werden.

Somit ist der folgende „Methodenkoffer" entstanden. Dieser wurde in drei Kategorien unterteilt:

- Grundlagen der Programmierung
- Schreibtraining
- Programmieraufgaben

Eine der wichtigsten Erkenntnisse führte dazu, dass die theoretischen Grundlagen abseits der tatsächlichen Programmierung stattfinden sollten (Ira Diethelm u. a. 2010). Anstatt sich neben den Inhalten auch noch auf Syntax und Semantik konzentrieren zu müssen, haben die Lernenden nun die Möglichkeit, sich auf die wesentlichen Lernziele zu fokussieren.

Um die Inhalte für die Schüler:innen zentral zu verwalten wurde ein EduVidual-Kurs angelegt. Dieser diente dazu, Informationen und Inhalte bereitzustellen, Abgaben einzurichten und Übungen abzuhalten.

Es wird nun auf die einzelnen Kategorien eingegangen. Im Anschluss ist der Ablaufplan für die entsprechenden Einheiten zu finden.

5.1 Grundlagen der Programmierung

Um den Einstieg möglichst simpel zu halten, legte man sich auf vier verschiedene Grundlagen fest: Schleifen, if/else-Konditionen, Algorithmen sowie Arrays/Sets. Für jede der vier Grundlagen wurde sowohl ein kurzer Informationstext verfasst[4] als auch ein Diagramm erstellt, um einen guten ersten Überblick zu schaffen, aber auch bei zukünftigen

[4] Die Infotexte wurden mit Unterstützung durch KI verfasst.

Unklarheiten eine gute Möglichkeit zu geben, sich schnell an das Gelernte zurück zu erinnern. Die ersten 5 Minuten der Einheiten wurden jeweils dazu verwendet, diese Informationstexte gemeinsam durchzulesen. Alle Grafiken sind im Anhang zu finden.

5.1.1 Schleifen

5.1.1.1 Infotext

*In der Programmierung stoßen wir oft auf Situationen, in denen sich Dinge **wiederholen**. Stell dir vor, du möchtest einem Computer sagen, er soll 100-mal denselben Satz schreiben. Statt denselben Befehl 100-mal einzutippen, verwenden wir **Schleifen**.*

*Sie helfen uns, Code **kürzer, klarer und effizienter** zu schreiben. Mit einer Schleife können wir Aufgaben **automatisieren**, die sich wiederholen - wie z. B. das Durchlaufen einer Liste, das Zeichnen von Mustern oder das Wiederholen einer Rechenaufgabe.*

*Schleifen können entweder so lange laufen, bis ein **Counter** erreicht ist oder eine **Bedingung** nicht mehr erfüllt wird.*

5.1.1.2 Ziele

Das Ziel dieser Einheit ist es, unterschiedliche Arten von Schleifen (while, for, do-while) kennenzulernen, die Funktionsweise zu verstehen und diese Prinzipien anwenden zu können. Dabei sollte das Wissen nicht auf direktem Wege vermittelt, sondern durch Beobachtungen erkannt und verstanden werden.

5.1.1.3 Inhalte

Das Wissen wird anhand von mehreren Runden des Kartenspiels „UNO“ (Mattel Shop, o. J.) mit speziellen Regeln nähergebracht. Das Spiel wird dabei nur zwischen der Lehrperson und einem einzelnen Lernenden stattfinden. Diese spielerische Variante soll die Motivation der Lernenden in dieser ersten Einheit fördern und das grundlegende Wissen somit nachhaltig verankern.

Das generelle Ziel des Spiels „UNO“ ist es, alle Handkarten in vier verschiedenen Farben loszuwerden. Man kann die anderen Spieler:innen durch Aktionskarten („+2“, „+4“) neue Karten ziehen lassen.

Erster Methodenkoffer

Die neu geschaffenen Regeln wurden vom Spielleiter angekündigt, aber nicht erläutert: „Bei einer blauen +2 ziehe ich normal 2 Karten. Bei den anderen Farben habe ich jeweils eine besondere Art zu ziehen. Findet sie heraus und schreibt sie auf einen Zettel." Die Schüler:innen müssen also von sich aus erkennen, welche neuen Regeln existieren und die Funktionsweise dieser Regeln erklären können. Die Regeln lauten wie folgt:

- Wird eine blaue „+2" Karte gelegt, muss der nächste Spieler zwei Karten ziehen.
- Wird eine rote „+2" Karte gelegt, muss der nächste Spieler so lange Karten ziehen, bis dieser weniger als 6 Karten in der Hand hat. (while-Schleife)
- Wird eine gelbe „+2" Karte gelegt, wird eine Karte vom nächsten Spieler gezogen. Wenn dieser Spieler weniger als vier Karten auf der Hand hat, zieht er weiter, bis diese Bedingung erfüllt wird. (do-while-Schleife)
- Wird eine grüne „+2" Karte gelegt, wird eine Zahl gewürfelt. Der nächste Spieler zieht entsprechend viele Karten. (for-Schleife)

Sobald die Lernenden zum Großteil ein Verständnis der neuen Regeln zeigen, wird diese Runde fertig gespielt und mit den Lernenden nachbesprochen. Dabei werden die Parallelen zwischen dem Karten ziehen und der verschiedenen Arten von Schleifen gezogen. Im Anschluss kann je nach verbliebener Zeit eine neue Runde mit allen Schüler:innen gespielt werden, in denen die Schleifen angewandt werden.

Den Abschluss der Einheit bildet eine kurze Präsentation der verschiedenen Schleifenarten. Hier werden die Arten der Schleifen erstmals benannt und mit dem Spiel in Verbindung gebracht. Es wird auch der Prozess dieser Schleifen visuell dargestellt. Die Vernetzung zwischen dem einfachen Ablauf des Spiels und dem komplexeren Ablauf einer Schleife soll als Basis für ein besseres Verständnis dienen.

5.1.2 if/else-Konditionen

5.1.2.1 Infotext

*In der Programmierung müssen wir oft **Entscheidungen treffen**. Stell dir vor, du möchtest einem Computer sagen: "Wenn es regnet, nimm einen Regenschirm - sonst nicht." Genau dafür verwenden wir **if/else-Konditionen**.*

*Mit solchen Bedingungen können wir den Ablauf unseres Programms steuern. Der Computer prüft dabei, ob **eine Aussage wahr ist**, und führt dann entsprechend **verschiedene Anweisungen** aus.*

Zum Beispiel:

Wenn eine Zahl größer als 10 ist, gib „zu groß" aus.

Sonst gib „passt" aus.

5.1.2.2 Ziele

Das Ziel dieser Einheit ist es, die Funktionsweise von Konditionen zu verstehen, die Begriffe „Bedingung", „wahr", „falsch" in diesem Kontext zu kennen und gegebene Abläufe bei im Vorhinein bestimmten Bedingungen anwenden zu können.

5.1.2.3 Inhalte

Auch in dieser Einheit bietet die Basis der Wissensvermittlung ein Gesellschaftsspiel: „Werwölfe von Düsterwald" („Werwölfe von Düsterwald | Asmodee Deutschland", o. J.). In diesem (besonders bei Jugendlichen beliebtem) Rollenspiel nehmen alle Beteiligten verschiedene Rollen an, grundlegend aber sind sie entweder ein „Dorfbewohner" oder ein „Werwolf". Während die „Werwölfe" verdeckt arbeiten und versuchen, die „Dorfbewohner" zu eliminieren, versuchen diese, die Identitäten der „Werwölfe" zu erraten, indem sie diese Personen beschuldigen einer zu sein.

Zusätzlich zu den Rollen werden an die Schüler:innen auch eine der folgenden Bedingungen verteilt:

- Wenn du beschuldigt wirst, musst du sprechen. Ansonsten schweigst du.
- Wenn du ein Werwolf bist, musst du laut sprechen. Ansonsten flüsterst du.
- Wenn du ein Dorfbewohner bist, musst du immer deine Finger kreuzen. Ansonsten nicht.
- Wenn du eine Frage stellst, musst du ganz hoch reden. Ansonsten redest du tief.
- Wenn du von etwas überzeugt bist, musst du am Ende jedes Satzes „safe" sagen. Ansonsten sagst du am Ende „oder so".
- Wenn du ein Werwolf bist, darfst du nur Fragen stellen. Ansonsten darfst du keine stellen.

- Wenn du ein Dorfbewohner bist, musst du den Leuten immer in die Augen schauen. Ansonsten musst du auf den Boden oder die Decke schauen.
- Wenn du beschuldigt wirst, musst du alles lustig finden. Ansonsten bleibst du ernst.
- Wenn du ein Werwolf bist, musst du jedes Mal den Sessel verrücken, wenn du redest. Ansonsten nicht.
- Wenn du Dorfbewohner bist, musst du deinem rechten Sitznachbar immer zustimmen. Ansonsten widersprichst du ihm immer.
- Wenn du Werwolf bist, musst du ganz schnell sprechen. Ansonsten sprichst du ganz langsam.
- Wenn du Dorfbewohner bist, musst du dauernd mit deinem rechten Fuß wippen. Ansonsten wippst du mit deinem linken Fuß.

Im Anschluss an die Spielrunde werden die Bedingungen aufgeklärt und durchbesprochen. Es wird dabei betont, dass jede Bedingung einen gleichen Aufbau hatte:

wenn Bedingung, dann [...], ansonsten [...]

Es wird also immer eine Bedingung gegeben, welche wahr oder falsch sein könnte, je nachdem wurde ein bestimmter Ablauf durchgeführt. Nun wird mittels einer Präsentation an die „if/else-Konditionen" herangeführt und offiziell benannt.

5.1.3 Algorithmen

5.1.3.1 Infotext

*Ein Algorithmus ist eine **Schritt-für-Schritt-Anleitung**, mit der ein Problem gelöst oder eine Aufgabe erledigt werden kann. In der Programmierung bedeutet das: Wir überlegen uns einen klaren Plan, den der Computer Schritt für Schritt ausführen kann.*

*Stell dir vor, du möchtest einem Roboter erklären, wie man einen Kuchen backt. Du musst **jeden einzelnen Schritt** genau angeben - vom Einschalten des Ofens bis zum Verzieren des Kuchens. Genauso funktionieren Algorithmen.*

*Ein Algorithmus ist also nicht unbedingt schon ein fertiges Programm - sondern eher die **Idee** oder das **Konzept** dahinter. Erst wenn wir diesen Algorithmus in eine Programmiersprache übersetzen, kann der Computer ihn ausführen.*

Gute Algorithmen sind klar, logisch aufgebaut und führen zuverlässig zum Ziel - egal, ob es darum geht, eine Liste zu sortieren, den kürzesten Weg zu finden oder ein Spiel zu steuern.

5.1.3.2 Ziele

Das Ziel dieser Einheit ist es, sowohl eine komplexe Aufgabe in einzelne, abarbeitbare Schritte zerlegen zu können, als auch diese Schritte möglichst klar und genau definieren zu können. Dabei soll auch ein Verständnis aufgebaut werden, warum dies nötig ist.

5.1.3.3 Inhalte

In dieser Einheit wird die Klasse in zwei bis drei Gruppen aufgeteilt. Jede Gruppe erhält zwei Aufgaben, welche möglichst genau durchzuführen sind.

Aufgabe 1: Zur Einführung bekommt jede der Gruppen ein gemischtes Kartendeck mit 20 Karten. Dieses soll aufgelegt werden und mittels Tauschmanövern zuerst nach Farbe und dann nach Größe sortiert werden (bspw. Gelb → Rot → Grün → Blau und 0 bis 9 aufsteigend). Pro Gruppe sollen dabei ein bis zwei Personen jeden Tauschprozess dokumentieren und mitzählen. Der Fairness halber wird diese Aufgabe in drei Durchläufen stattfinden und die Summe aller Tauschprozesse ermittelt. Ziel jeder Gruppe ist es, möglichst wenige Tauschmanöver zu benötigen. Im Anschluss muss der hier verwendete Algorithmus verschriftlicht werden. Dieser wird von der Lehrperson ausgetestet und auf Genauigkeit überprüft. Sollte die Beschreibung zu ungenau sein, dürfen die Schüler:innen dies korrigieren.

Aufgabe 2: Inspiriert von einem viralen Video (Josh Darnit 2017) bekommt jede Gruppe 10 Minuten Zeit, um eine Anleitung zu schreiben, wie aus einer verschlossenen, bisher ungeöffneten Plastikflasche getrunken werden kann. Diese Anleitung wird anschließend von der Lehrperson getestet, dabei müssen die Anweisungen genauestens befolgt werden. Sollte einer der Schritte zu ungenau formuliert sein, wird entweder versucht, diesen mit einer strategischen Inkompetenz auszuführen, oder der Versuch wird abgebrochen. Die Gruppen erhalten nun eine zweite Chance, ihre Angaben detaillierter zu verfassen.

Nach diesem praktischen Input werden die Lernenden in die Erstellung eines Ablaufdiagramms eingeführt. Als Basis dazu dient die zugehörige Grafik der Einheit. Zugehörig zur ersten Programmieraufgabe (Ampel) soll ein Ablaufdiagramm für eine Ampel erstellt werden. Im Anschluss an diese Einheit kann den Schüler:innen eine Musterlösung zur Verfügung gestellt werden.

5.1.4 Arrays und Sets

5.1.4.1 Infotext

Ein **Array** ist eine **Liste**, in der wir Elemente speichern können. Dabei wird alles in einer **festen Reihenfolge** gespeichert. Die **Position** im Array wird mit einem **Index** bestimmt. Die **Länge** des Arrays kann **nicht verändert** werden.

Beispiel:

```
const alter = [14, 28, 33, 15, 38];
```

`alter[2]` wäre hier 33

Ein **Set** ist ebenfalls eine Liste, allerdings wird hier die **Reihenfolge nicht beachtet**. **Doppelte** Werte können hier **nicht** gespeichert werden. Die **Größe** des Sets kann **beliebig** angepasst werden.

```
const letters = new Set ();
letters.add("a");
letters.add("b");
letters.add("b");
```

Im Set `letters` wäre jetzt nur `["a", "b"]` enthalten.

5.1.4.2 Ziele

Ziel dieser Einheit ist es, Listen in der Programmierung einsetzen zu können, den Unterschied zwischen Array und Sets zu verstehen, mit einem Index arbeiten zu können. Es sollen außerdem Algorithmen angewendet werden können, um Elemente einzuordnen und zu suchen.

5.1.4.3 Inhalte

Diese Einheit ist in zwei Phasen unterteilt: Arrays und Sets.

Phase 1: Arrays Die Klasse wird in zwei bis drei Gruppen unterteilt. Jede der Gruppen erhält eine Mappe mit 4 Klarsichtfolien, welche jeweils in 4 A6-Abteile unterteilt sind. Die Abteile sind mit den Zahlen 0 bis 15 durchnummeriert (Abb. 9).

Abbildung 9: Mappe - Einheit Arrays

In diese Abteile werden zufällig 12 Spielkarten eingeordnet. Im Anschluss sollen Karten hinzugefügt, entfernt oder vertauscht werden - je nach Anweisungen der Lehrperson. Dabei sollten die Aufgaben entsprechend angeleitet werden, um folgende Fragen zu beantworten:

- Was passiert mit dem Array, wenn Karten hinzugefügt, entfernt oder vertauscht werden?
- Was passiert, wenn ein Platz schon belegt ist?
- Was passiert mit dem Platz, wenn eine Karte entfernt wird?
- Was passiert, wenn das Array voll ist? Kann ich es vergrößern?

Anschließend erhält jede Gruppe eine zweite, gleichermaßen gefüllte Mappe ohne Karten. Die erste Mappe soll nun sortiert werden, indem sie die Karten der Reihenfolge nach in die zweite Mappe einordnen. Dabei soll von der Lehrperson gezielt abgefragt werden, welcher Algorithmus von den Gruppen für diese Aufgabe verwendet wird.

Am Ende der Phase werden die Inhalte kurz nachbesprochen und Unklarheiten erläutert.

Phase 2: Sets Die ganze Klasse hat nun eine größere Zahl an Kleiderhaken sowie eine Kleiderstange zur Verfügung. Die Kleiderstange wird von zwei Schüler:innen gehalten. Alle anderen Teilnehmenden dürfen nun die Kleiderhaken mittels eines Malerbands und Edding mit einer beliebigen Zahl beschriften. Der Kleiderhaken sollte korrekt auf der Kleiderstange einsortiert werden. Dabei sollte die Phase entsprechend angeleitet werden, um folgende Fragen zu beantworten:

- Wie könnte ich einen Kleiderhaken am schnellsten einordnen?
- Was passiert, wenn eine Zahl öfters vorkommt?
- Was passiert, wenn die Kleiderstange zu schwer wird?
- Wenn ich ein bestimmtes Element entfernen will, wie lange dauert die Suche?

Am Ende der Phase werden die Inhalte, sowie die Unterschiede zwischen Arrays und Sets kurz nachbesprochen und Unklarheiten erläutert.

5.2 Schreibtraining

In diesem Teil der Unterrichtsplanung wird auf das Schreiben mittels einer QWERTZ-Tastatur geachtet, mit besonderem Fokus auf Sonderzeichen. Es werden außerdem Grundlagen der Dateiverwaltung behandelt mit besonderem Fokus auf Ordnerstrukturen und zip-Dateien. Weiters werden auf theoretischer Weise die Verwendung und Arten von Variablen behandelt. Abschließend werden diese und alle bisherigen Inhalte getestet, indem sie in Form einer Übung in die Praxis umgesetzt werden.

Wie sich später herausstellen wird, ist die Betitelung dieser Kategorie misslungen und wird auch inhaltlich im finalen „Methodenkoffer" angepasst.

5.2.1 Tastaturschreiben

5.2.1.1 Ziele

Ziel dieser Unterrichtseinheit ist es, dass die Lernenden jegliche Sonderzeichen auf einer QWERTZ-Tastatur finden und mithilfe der richtigen Tastenkombination diese Sonderzeichen schreiben können. Als Kann-Ziel wird hier festgelegt, dass dies bis Ende der Einheit möglichst flüssig und „blind" erfolgt.

5.2.1.2 Einführung

Die Schüler:innen erhalten eine Einführung in die QWERTZ-Tastatur mittels einer Bildschirmaufnahme, in welcher die einzelnen Sonderzeichen auf dieser Tastatur gefunden und die Tastenkombinationen erklärt werden (Max Becker 2026).

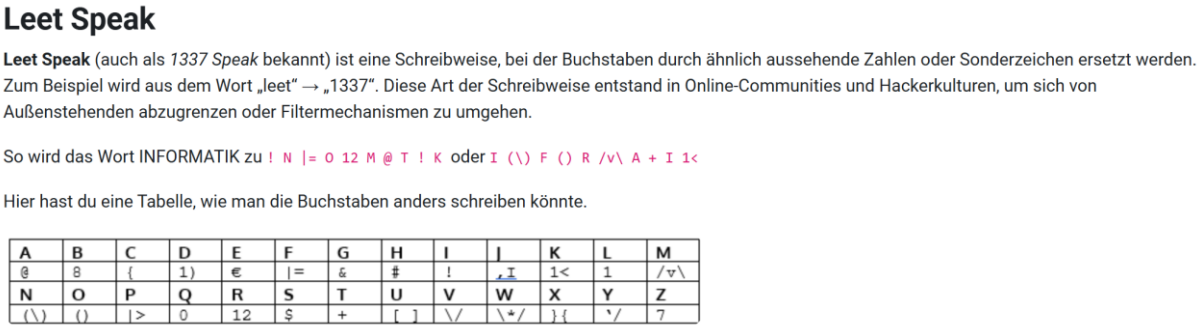
Anschließend sollten sie eine einführende Übung absolvieren, in welcher ein vorgegebener Text mit einigen Sonderzeichen genauso abgeschrieben werden soll. Der Text ist dabei nicht kopierbar.

```
Hallo! Ich bin Max_2008 und gehe in die 7. Klasse.
```

```
Meine E-Mail lautet: Max@schule-mail.de
Ich nutze oft Webseiten wie www.mathe-hilfe.de oder www.fragen-und-
antworten.net
Mein Stundenplan ist voll: Mathe (8:00 – 8:45), Englisch (9:00 – 9:45),
Bio (10:00 – 10:45).
Mein WLAN heißt „Schule-24“ und das Passwort ist geheim ;)
Ich speichere meine Dateien in Ordnern wie: C:\Schule\Hausaufgaben\
Wenn ich programmiere, schreibe ich z. B.: if name == "Max":
print("Hallo!")
```

5.2.1.3 Einheit „Leet Speak“

In Folge findet die Einheit „Leet Speak“ statt. Den Lernenden wird ein einführender Text, sowie eine Übersetzungstabelle vorgegeben (Abb. 10).



Leet Speak

Leet Speak (auch als *1337 Speak* bekannt) ist eine Schreibweise, bei der Buchstaben durch ähnlich aussehende Zahlen oder Sonderzeichen ersetzt werden. Zum Beispiel wird aus dem Wort „leet“ → „1337“. Diese Art der Schreibweise entstand in Online-Communities und Hackerkulturen, um sich von Außenstehenden abzugrenzen oder Filtermechanismen zu umgehen.

So wird das Wort INFORMATIK zu `! N |= 0 12 M @ T ! K` oder `I (\) F () R /v\ A + I 1<`

Hier hast du eine Tabelle, wie man die Buchstaben anders schreiben könnte.

A	B	C	D	E	F	G	H	I	J	K	L	M
@	8	(	1)	€	|=	&	#	!	,_I	1<	1	/v\
N	**O**	**P**	**Q**	**R**	**S**	**T**	**U**	**V**	**W**	**X**	**Y**	**Z**
(\)	()	|>	0	12	$	+	[ ]	\/	*/	}{	`/	7

Abbildung 10: Leet Speak, einführender Text und Übersetzungstabelle

Im Laufe dieser Aufgabe sind von den Schüler:innen fünf verschiedene Rätsel zu lösen. Die Antworten dieser Rätsel sollen per Übersetzungstafel in Sonderzeichen umgewandelt werden. Die Rätsel und Lösungen lauten wie folgt:

5.2.1.3.1 Willkommenstext

```
[ W3lc0m3 70 7h3 D4rk W3b ]
Du hast den Zugangspunkt gefunden – aber der Zugriff ist noch nicht
freigegeben.
Bevor du in unser N3tw0rk aufgenommen wirst, musst du deine Sk1llz
unter Beweis stellen.
Deine M1ss10n: Crack the code. Solve the riddles.
Achte auf jeden B1t – ein falsches Leerzeichen, und du wirst
gebl0ckt.
Nur wer alle T35t5 ohne 3rr0r besteht, wird ein Teil unseres
5ys73ms.
Hast du das Zeug, dich ins M41nFr4m3 zu h4ck3n?
```

5.2.1.3.2 Aufgabe 1

```
[ G4t3_01: P4$$w0rd_Pr0t0c0l ]
Let’s 57ar7 ez. Das erste |>a55wor7 !st GEHEIM. Wie /v\an s(#on
geme12kt h4t, 5prec#en \*/ir &€rn€ au|= englisch & mi7 1337-Speak.
\*/!€ L@[_]T€T D@$ P@$$\*/OR+ ?
```

Lösung:

```
$€{12€+
```

5.2.1.3.3 Aufgabe 2

```
[ G4t3_02: Tr4nsl4t10n_F41lur3 ]
Un5er 80T hat 1 P@55wor7 abgef@ng€n.
MICHAEL01
Kannst 1)u e5 sich€r€r ges7al7en?
```

Lösung:

```
/v\!{#@€1_01
```

5.2.1.3.4 Aufgabe 3

```
[ G4t3_03: C0d3n4m3_4uth ]
Für unser 5ch4tt3n-N3tw0rk ha5t du 3inen Codename b3k0mmen: NOOBIE
5chr€!be den Codename um.
```

Lösung:

```
(\)()()8!€
```

5.2.1.3.5 Aufgabe 4

```
[ G4t3_04: D4t4_3nc0d3r ]
Die Datei l0ck3d_f113.sql enthält die U5ern4mes un$erer 5erv€r. Alle
U5ern4mes sind in 1337-Speak ge$c#rie8en. Wie heißt AGENT SMITHY in
un$erem N3tw0rk?
```

Lösung:

```
@&€(\)+ $/v\!+#`/
```

5.2.1.3.6 Aufgabe 5

```
[ G4t3_05: M41nFr4m3_3ntr4nc3 ]
Le7z7e5 Le\/el. 7_u&an& nur für Vo11pro|=i$.
Der Zugriffscode zum System-Neustart muss korrekt eingegeben werden
- natürlich in L337. Er lautet DZX3FJW5KPQ9VU
```

Lösung:

```
1)7_}{3|=,I\*/51<|>0_9\/[_]
```

5.2.2 Navigation

5.2.2.1 Ziele

Die Lernenden wissen, was Dateitypen sind, können Dateistrukturen in zip-Dateien verpacken sowie ein Ordnersystem anlegen und verwalten.

5.2.2.2 Einführung

In dieser Einheit erhalten die Schüler:innen einen einführenden Text.

Während auf dem Smartphone meistens alles auf einem Ort zu finden ist, gibt es am PC dedizierte Ordner für verschiedene Arten von Dateien. Wer seine Dateien schnell finden will, hat meist eine ordentliche Ordnerstruktur.

Was ist eine Datei?

Eine Datei ist ein Objekt, das am PC abgespeichert werden kann. Die Beschriftung besteht immer aus einem Namen und einem Dateitypen, getrennt durch einen Punkt. Zum Beispiel:

```
Song 2.mp3
Tagebuch.docx
Homepage.html
```

Der Dateityp sagt dabei aus, was für eine Art von Objekt es ist.

Sobald der Text gemeinsam durchgegangen wurde, folgt die nächste Aufgabe.

5.2.2.3 Aufgabe Dateitypen

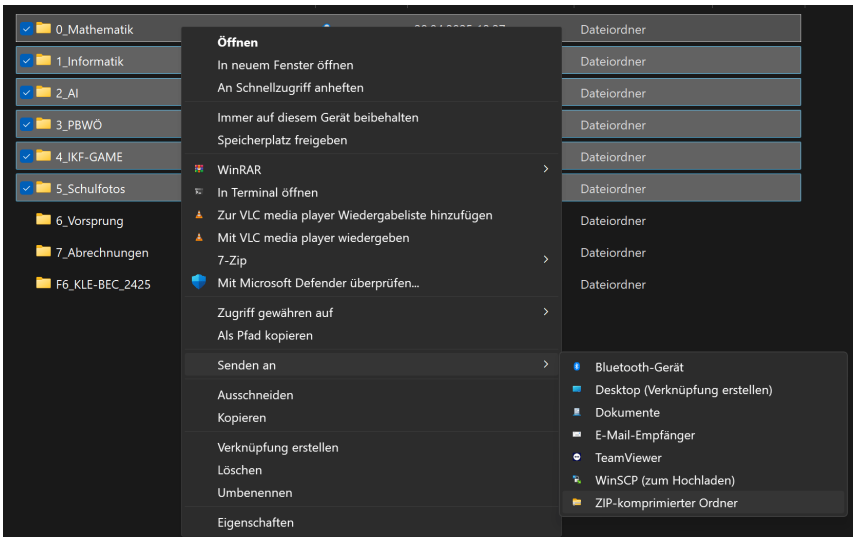
Die Lernenden sollten nun durch Recherche im Internet mindestens 5 verschiedene Dateitypen herausfinden, ihn einer Kategorie (Medien, Dokumente, Web) zuordnen und in Stichwörtern beschreiben, welche Form von Datei dieser Typ beinhaltet (bspw. Musikdateien für .mp3 oder Word-Dokumente für .docx). Diese Informationen werden in einer von EduVidual eingebauten Notizapp gesammelt, welche für alle Teilnehmer:innen des Kurses sichtbar ist. Die verschiedenen Dateitypen werden im Anschluss kurz von der Lehrperson mit der Klasse durchbesprochen.

5.2.2.4 Einführung .zip-Datei und Ordnerstruktur

Den Schüler:innen wird nun durch eine kurze Anleitung erklärt, wie mehrere Ordner und Dateien in Windows 10 in eine .zip-Datei verpackt werden kann.

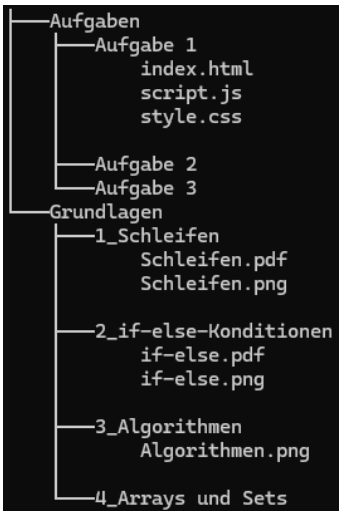
Wenn du eine ganze Ordnerstruktur als Datei haben möchtest, kannst du sie in eine .zip-Datei umwandeln.

In Windows 10 wähle alle Dateien und Ordner aus, die du benötigst --> Rechtsklick --> Senden an --> Zip-komprimierter Ordner



Anschließend sollen die Lernenden eine gegebene Ordnerstruktur im Ordner „Dokumente“ anlegen, diese als .zip-Datei verpacken und im Kurs mittels einer eingerichteten Aufgabe abgeben. Diese Ordnerstruktur wird im Laufe der Unterrichtseinheiten Verwendung finden.

*Erstelle im Ordner **Dokumente** den Ordner "Programmierung" und lege dort folgende Ordnerstruktur an. Die Dateien, die du brauchst, findest du alle in unserem eduvidual-Kurs. Gib die Ordnerstruktur als **.zip-Datei** ab!*



5.2.3 Variablen

5.2.3.1 Ziele

In dieser Einheit sollen die Schüler:innen verstehen, dass eine Variable einen Platzhalter in einem Programm darstellt. Sie sollten außerdem der Unterschied zwischen Variable und Konstante kennen.

5.2.3.2 Einführung

Anfangs erhalten die Schüler:innen ähnlich wie in den Grundlagen der Programmierung einen einführenden Text, sowie ein Diagramm, um eine Übersicht über das Thema zu geben.

Was ist eine Variable?

In der Programmierung ist eine Variable ein Platzhalter für Daten, die gespeichert und später verwendet werden können. Stell dir dabei eine Schublade vor, in die du etwas hineinlegen, herauslöschen und ersetzen kannst.

*In einer Variable kann bspw. eine **Zahl**, ein **Text** oder ein **Wahrheitswert** (true/false) stehen.*

Wie definiert man eine Variable?

Wenn sich die Variable ändern darf:

```
let name = "Max"; // Die Variable 'name' hat jetzt den Inhalt "Max"
name = "Anita"; // Der Inhalt hat sich jetzt auf "Anita" geändert
```

Wenn die Variable immer gleich bleiben soll:

```
const pi = 3.14;
```

Variablen verknüpfen

Wenn ich einen bestimmten Text ausgeben will, in dem Variablen verknüpft werden, kann ich das so machen:

```
let name = "Max"; // Variable definieren
let alter = 29; // Variable definieren
alert(`Hallo, mein Name ist ${name} und ich bin ${alter} Jahre
alt.`);
```

alert ist ein Befehl, um einen Text auszugeben. In der Klammer steht der Text zwischen zwei Hochkommas ``. Um die Variable zu verwenden, schreiben wir ${}, zwischen den geschwungenen Klammern steht der Variablenname.

Ausgabe: Hallo, mein Name ist Max und ich bin 29 Jahre alt.

Sobald der Text gemeinsam durchgegangen wurde, folgt eine praktische Übung.

5.2.3.3 Einheit Wortkärtchen

Jede:r der Lernenden erhält nun ein Kärtchen. Manche Kärtchen sind mit Inhalten beschriftet, andere noch nicht. Ein Kärtchen wurde mit einem roten Inhalt beschriftet, dieser steht für eine Konstante. Im Folgenden wird von der Lehrperson ein Lückentext vorgelesen, welcher bei jeder Wiederholung länger wird. Die Lücken werden mittels der Kärtchen gefüllt. Sollte ein zusätzlicher Inhalt benötigt werden, wird ein neues Kärtchen beschriftet. Die Inhalte der Kärtchen können geändert oder auch durch Formeln berechnet werden. Ein Lückentext könnte wie folgt lauten:

*Hallo, ich bin **Herr Becker**.*

*Hallo, ich bin **Herr Becker** und bin **29** Jahre alt.*

*Hallo, ich bin **Herr Becker** und bin im Jahr **2025** **29** Jahre alt.*

*Hallo, ich bin **Herr Becker** und bin im Jahr **2025** **29** Jahre alt, heute ist **Montag**.*

*Hallo, ich bin **Herr Becker** und bin im Jahr **2025** **29** Jahre alt, heute ist **Montag**, es ist ____ Uhr.*

Weitere Inhalte können improvisiert werden.

Sobald das Konzept vom Großteil der Gruppe verstanden wird, kann die Einheit abgeschlossen und mit der Klasse besprochen werden. Es sollte nun auch der Bezug zu Variablen hergestellt werden, indem der Infotext erneut durchgelesen wird.

5.2.3.4 Übung Variablenverknüpfung

Zum Abschluss dieser Einheit soll eine Übung im EduVidual-Kurs abgehalten werden. Dabei ist von den Lernenden entweder ein korrekter Befehl für eine gegebene Ausgabe, oder eine vollständige Ausgabe für einen Befehl zu ermitteln.

Gib den richtigen Befehl für die Ausgabe an oder zeige, was die Ausgabe sein wird.

Was ist die Ausgabe?

```
let name = "Herr Becker";
let faecher = "Mathematik und Informatik";
alert(`Hallo, ich bin ${name} und ich unterrichte ${faecher}.`);
```

Lösung

Hallo, ich bin Herr Becker und ich unterrichte Mathematik und Informatik.

Was ist die Ausgabe?

```
const pi = 3.14;
const euler = 2.718;
alert(`Die mathematische Konstante PI ist ${pi} groß. Die eulersche
Zahl ist mit ${euler} etwas kleiner.`);
```

Lösung

Die mathematische Konstante PI ist 3.14 groß. Die eulersche Zahl ist mit 2.718 etwas kleiner.

Wie lautet der richtige Ausgabebefehl?

```
let loggedIn = true;
let username = "Maze0";
```

Ausgabe:

Ist der User Maze0 eingeloggt? true

Lösung

```
alert(`Ist der User ${username} eingeloggt? ${loggedIn}`);
```

Wie lautet der richtige Ausgabebefehl?

```
let produkt = "Laptop";
let preis = 979.99;
let anzahl = 2;
```

Ausgabe:

Produkt: Laptop, Preis: 979.99€, Stückzahl: 2

Lösung

```
alert(`Produkt: ${produkt}, Preis: ${preis}€, Stückzahl:
${anzahl}`);
```

Welches ist die richtige Ausgabe?

```
let score = 0;
alert(`Aktueller Punktestand: ${score}`);
score = score + 10;
alert(`Neuer Punktestand: ${score}`);
score = score * 10;
alert(`Neuer Punktestand: ${score}`);
```

Antwort A (richtig)

Aktueller Punktestand: 0

Neuer Punktestand: 10

Neuer Punktestand: 100

Antwort B

Aktueller Punktestand: 0

Neuer Punktestand: 0 + 10

*Neuer Punktestand: 0 + 10 * 10;*

Antwort C

Aktueller Punktestand: 0

Neuer Punktestand: 10

Neuer Punktestand: 0

Antwort D

Aktueller Punktestand: 0;

Neuer Punktestand: 10;

Neuer Punktestand: 100;

Welches ist der richtige Ausgabebefehl?

```
let auto = "Porsche Cayenne";
let ps = 575;
let baujahr = 2023;
let preis = 98000;
```

Ausgabe:

Ich fahre einen Porsche Cayenne mit 575 PS. Er ist 2 Jahre alt und kostet 98000€.

Antwort A (richtig)

```
alert(`Ich fahre einen ${auto} mit ${ps} PS. Er ist ${2025 -
baujahr} Jahre alt und kostet ${preis}€.`);
```

Antwort B

```
alert(`Ich fahre einen ${auto} mit ${ps} PS. Er ist ${2025 -
baujahr} Jahre alt und kostet ${preis}€.`)
```

Antwort C

```
alert('Ich fahre einen ${auto} mit ${ps} PS. Er ist ${2025 +
baujahr} Jahre alt und kostet ${preis}€.');
```

Antwort D

```
alert('Ich fahre einen ${Auto} mit ${PS} PS. Er ist ${2025 - Baujahr} Jahre alt und kostet ${Preis}€.')
```

5.2.4 Syntax

5.2.4.1 Ziele

Die Lernenden sollen in dieser Einheit übliche Zeichen in der Programmierung (bspw. „;“ und „//“) und deren Bedeutung kennen. Sie sollten ebenfalls Code mittels Einrückungen entsprechend formatieren können, um eine Übersichtlichkeit zu bewahren. Außerdem sollten sie dazu in der Lage sein, Codeabschnitte in vorgegebenen Dateien an der richtigen Stelle einzufügen und zu formatieren. Weiters sollten die Schüler:innen in der Lage sein, alle bisher gelernten, theoretischen Inhalte in den vorgegebenen Codezeilen zu erkennen und für die späteren Aufgaben in der Praxis anzuwenden.

5.2.4.2 Einführung

Auch in dieser Einheit erhalten die Schüler:innen einen einführenden Text, sowie eine kurze Übersicht etwaiger Zeichen und Formatierungen mit entsprechenden Codebeispielen (Abb. 11).

Eine Syntax beschreibt die Regeln und Struktur, wie du Code in einer bestimmten Programmiersprache korrekt schreibst. So gibt es eine bestimmte Art und Weise, Variablen zu definieren, Funktionen zu erstellen und Operatoren zu benutzen, damit der Computer den Code korrekt "lesen" kann.

z.B. weiß der Computer, was mit `let name = "Anna";` gemeint ist, aber `let name = anna` versteht er nicht.

Wichtige Zeichen

;	Am Ende eines Befehls muss ein ; stehen. Das zeigt, dass der Befehl zu Ende ist.	`let name = "Anna";`
//	Willst du einen Kommentar erstellen (Text, der vom Computer ignoriert wird), musst du // schreiben. Alles rechts davon wird vom Computer ignoriert.	`const pi = 3.14; // Das ist eine Konstante`

Einrückung (Tabulator) 	Öffnest du eine Klammer (), [] oder {} und beginnst eine neue Zeile, so wird die Zeile üblicherweise mit dem Tabulator eingerückt. So ist der Code übersichtlicher.	`let alter = 25;`
		`while (alter < 30) {`
		`  alert(alter);`
		`  alter = alter + 1;`
		`}`

Operatoren	Mit einem Operator kann ich einer Variable neue Werte zuweisen (=).
	Ich kann auch zwei Werte vergleichen (<, >, ==, <=, >=).
		`let alterMax = 29;`
		`let alterAnita = 25;`
		`if (alterMax > alterAnita) {`
		`  alert("Max ist alt");`
		`}`

Abbildung 11: Screenshot Einführung Syntax

5.2.4.3 Aufgabe Syntax

Nach einer kurzen Besprechung des Infotextes sollen die Lernenden folgende Aufgabe bearbeiten. Dazu sollte mit den Schüler:innen gemeinsam das Programm „Notepad++" („Notepad++", o. J.) auf den Arbeitsgeräten installiert werden. Dieses wird auch im Laufe der weiteren Programmieraufgaben als Entwicklungsumgebung verwendet, da es einfach zu installieren, plattformunabhängig und gratis ist, sowie eine einfache grafische Oberfläche besitzt.

Speichere die Dateien in deiner Ordnerstruktur unter `2_Schreibübungen/4_Syntax/` ab.

Öffne die Datei `script.js` in Notepad++ und schreibe folgenden Code an den richtigen Ort. Kommentiere mit `//` dazu, was die Befehle machen.

```
let name = "Anna";
const pi = 3.14;
alert(`Ich heisse ${name} und Pi ist ${pi} gross.`);
let liste = ["Apfel", "Banane", "Clementine"];
alert (liste[0]);
let a = 10;
let b = 20;
alert(a+b);
alert(a == b);
let age = 18;
if (age >= 18) {
  alert("Du bist volljährig");
} else {
  alert("Du bist minderjährig");
}
for (let i = 0; i < 5; i++) {
  alert(i);
}
```

Die Dateien index.html, style.css sowie script.js sind im Anhang zu finden.

Eine Lösung könnte folgendermaßen aussehen:

```
function start () {
      // START WIRD GEDRÜCKT

      let name = "Anna";    // Definiert Variable name mit Inhalt
"Anna"
      const pi = 3.14;// Definiert Konstante pi mit Wert 3.14
      alert(`Ich heisse ${name} und Pi ist ${pi} gross.`);  //
Ausgabe
      let liste = ["Apfel", "Banane", "Clementine"];  // Definiert
Array mit den Inhalten "Apfel", "Banane", "Clementine"
      alert (liste[0]);     //gibt das Element an Index 0 des Arrays
liste aus --> "Apfel"
```

```
     let a = 10;           // Definiert Variable a mit Wert 10
     let b = 20;           // Definiert Variable b mit Wert 20
     alert(a+b);           // Gibt die Summe von Variablen a und b
aus --> 30
     alert(a == b);        // Gibt aus, ob a und b gleich groß sind
--> false
     let age = 18;         // Definiert Variable age mit Wert 18
     if (age >= 18) {// Falls age größer oder gleich 18 ist
          alert("Du bist volljährig");    // Ausgabe
     } else {        // ansonsten
          alert("Du bist minderjährig");  // Ausgabe
     }
     for (let i = 0; i < 5; i++) {   // Schleife, die alle Zahlen
von 0 bis 4 durchläuft
          alert(i);  // Ausgabe
     }

     // ALLES BIS HIER WIRD AUFGERUFEN
}

// HIER WIRD NICHTS AUFGERUFEN
```

5.3 Programmieraufgaben

In den Programmieraufgaben [5]sollen die Lernenden sich nun der Praxis stellen. In einer ersten Aufgabe wird von der Lehrperson am Beamer vorprogrammiert und jeder Schritt erläutert. Dabei sollte auch regelmäßig das Ergebnis bei den Schüler:innen kontrolliert werden, um Abschreibfehler zu vermeiden.

Die zweite Aufgabe wird in der Art des „pair-Programming“ stattfinden. Wie bereits im theoretischen Teil erläutert, bietet diese Methodik erhebliche Vorteile für Programmieranfänger:innen, indem zwar mehr Zeit benötigt wird, die Ergebnisse allerdings einer höheren Qualität entsprechen. Die Teams dürfen sich dementsprechend selbstständig fügen, einigen wenigen Schüler:innen wird auch das Arbeiten ohne Partner:in erlaubt.

5.3.1 Ampel

5.3.1.1 Aufgabenstellung

*Du hast eine fertige Website mit einer Ampel gegeben. Die IDs für die Elemente lauten wie in der Liste unten. Deine Aufgabe: Schreibe in der Datei **script.js** einen Code, sodass die Ampel bei Start dem unten beschriebenen Ablauf folgt und sich ständig wiederholt.*

[5] Die dafür vorbereiteten Websites sind mit Unterstützung von KI entstanden.

#IDs und .Klassen

```
#red, .red: Oberer Kreis
#yellow, .yellow: Mittlerer Kreis
#green, .green: Unterer Kreis
#startBtn: Start
#stopBtn: Stopp
.light: Alle drei Kreise
.on: Dieser Kreis leuchtet
```

Ablauf

10 Sekunden Rot --> 2 Sekunden Rot&Gelb --> 10 Sekunden Grün --> 4mal Grün blinkend --> 2 Sekunden Gelb --> Wiederholung

Als Angabe sollten die Lernenden die drei vorgefertigten Dateien (index.html, style.css, script.js) als .zip-Datei herunterladen in einem entsprechenden Ordner in ihrem Dateisystem abspeichern. Zur Übersicht des Ablaufs sollte das eigene Ablaufdiagramm dienen.

5.3.1.2 Lösung

Diese Aufgabe wird in drei Stufen von der Lehrperson vorgearbeitet, es wird dabei nur die script.js-Datei verändert. Somit sollte für den einfachen Einstieg der „bottom up"-Ansatz verfolgt werden, insofern man kleine Elemente des Codes programmiert und auf diesen dann ansetzt, um das weitere Programm zu schreiben. Eine entsprechende Vorlage ist in den einzelnen Stufen ebenfalls im Anhang zu finden und wird am Ende jeder Stufe per EduVidual freigeschalten. Das finale Dokument sollte folgendermaßen aussehen:

```
// Speichert, ob die Ampel gerade läuft oder nicht
let ampelLaeuft = false;

// Speichert alle Zeitgeber (damit man sie stoppen kann)
let timer = [];

// Holt die Lampen aus dem HTML
const rot = document.getElementById("red");
const gelb = document.getElementById("yellow");
const gruen = document.getElementById("green");

// Diese Funktion macht alle Lampen aus
function allesAus() {
     rot.classList.remove("on");
     gelb.classList.remove("on");
     gruen.classList.remove("on");
}

// Diese Funktion schaltet bestimmte Lampen ein
function setzeLichter(rotAn, gelbAn, gruenAn) {
```

```
      if (rotAn) rot.classList.add("on"); else
rot.classList.remove("on");
      if (gelbAn) gelb.classList.add("on"); else
gelb.classList.remove("on");
      if (gruenAn) gruen.classList.add("on"); else
gruen.classList.remove("on");
}

// Diese Funktion startet den Ampel-Ablauf
function starteAmpel() {
      if (ampelLaeuft) return; // Wenn sie schon läuft: nichts tun
      ampelLaeuft = true;
      ablauf();
}

// Diese Funktion stoppt die Ampel
function stoppeAmpel() {
      ampelLaeuft = false;
      timer.forEach(clearTimeout); // Stoppt alle laufenden Timer
      timer = [];
      allesAus();
}

// Die Hauptfunktion für den Ablauf der Ampel
function ablauf() {
      if (!ampelLaeuft) return;

      // ROT an (10 Sekunden)
      setzeLichter(true, false, false);
      timer.push(setTimeout(() => {

            // ROT + GELB an (2 Sekunden)
            setzeLichter(true, true, false);
            timer.push(setTimeout(() => {

                  // GRÜN an (10 Sekunden)
                  setzeLichter(false, false, true);
                  timer.push(setTimeout(() => {

                        // GRÜN blinkt 4x (0.5s an, 0.5s aus)
                        blinkeGruen(0);

                  }, 10000)); // nach 10 Sekunden grün
            }, 2000)); // nach 2 Sekunden rot+gelb
      }, 10000)); // nach 10 Sekunden rot
}

// Funktion zum grünen Blinken (4 Mal)
function blinkeGruen(anzahl) {
      if (anzahl >= 4 || !ampelLaeuft) {
            // Danach GELB für 2 Sekunden
            setzeLichter(false, true, false);
            timer.push(setTimeout(() => {
                  ablauf(); // Neustart
            }, 2000));
            return;
```

```
        }

        // Grün aus
        gruen.classList.remove("on");
        timer.push(setTimeout(() => {
            // Grün an
            gruen.classList.add("on");
            timer.push(setTimeout(() => {
                blinkeGruen(anzahl + 1);
            }, 500));
        }, 500));
}

// Knöpfe mit Funktionen verbinden
document.getElementById("startBtn").addEventListener("click",
starteAmpel);
document.getElementById("stopBtn").addEventListener("click",
stoppeAmpel);
```

5.3.2 Hangman

Da die Lernenden diese Aufgabe in totaler Alleinstellung der Teams lösen sollten, wurde eine entsprechend ausführliche Angabe vorbereitet.

5.3.2.1 Aufgabenstellung

Ziel dieses Programmes ist es, das bekannte Spiel „Hangman" (*Galgenmännchen - Alternative Ways*, o. J.) zu produzieren. Dazu wurde ein Ablaufdiagramm des Programms, sowie Anweisungen inklusiver Codeteile und -erklärungen gegeben (Abb. 12, 13). Die genaue Aufgabenstellung ist im Anhang zu finden.

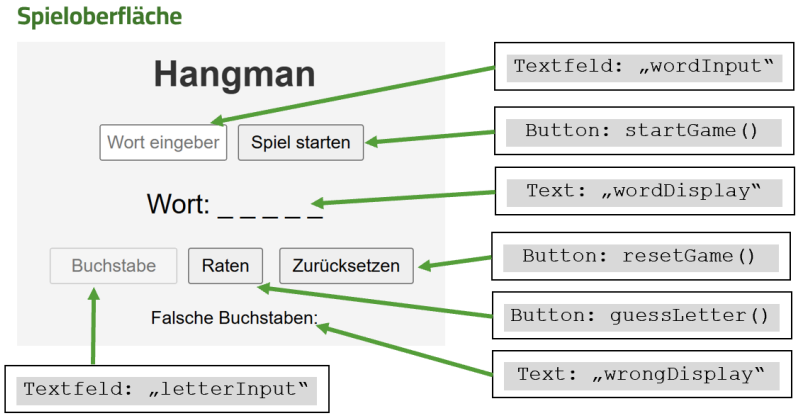


Abbildung 12: Angabe Hangman, Erklärung Spieloberfläche

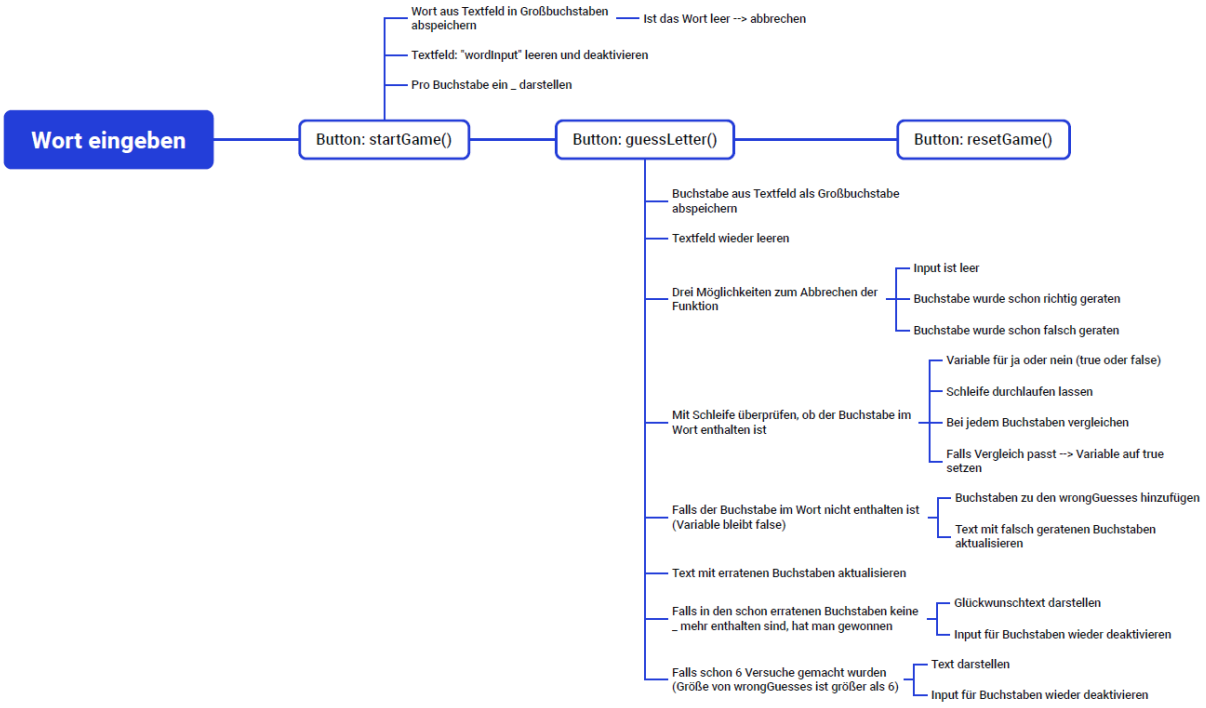


Abbildung 13: Angabe Hangman, Ablaufdiagramm

Am Anfang der Einheit werden die Paare ermittelt. Dabei wird ebenfalls festgelegt, wer welche Rolle einnehmen wird (Navigator, Programmer). Dabei werden die Aufgaben und der jeweiligen Rollen genau festgelegt. Im Laufe der Einheiten werden die Rollen ungefähr jede halbe Stunde getauscht.

5.3.2.2 Lösung

In dieser Aufgabe wird in zwei Stufen gearbeitet, die jeweiligen Musterlösungen werden nach Abschluss einer Stufe per EduVidual freigegeben. Der finale Code sollte folgendermaßen aussehen:

```
// Variablen
let word = "";  // Zu erratendes Wort
let guessedWord = []; // Schon erratene Buchstaben, ansonsten wird es
mit _ gefüllt
let wrongGuesses = [];     // Falsch geratene Buchstaben

// Variablen mit Elementen von HTML
let wordInput = document.getElementById("wordInput"); // Textfeld
wordInput als Variable speichern
let letterInput = document.getElementById("letterInput");  //
Textfeld letterInput als Variable speichern
let wordDisplay = document.getElementById("wordDisplay");  // Text,
der dargestellt wird --> Wort: _ _ _ _ _
let wrongDisplay = document.getElementById("wrongDisplay");      //
Text, der dargestellt wird --> Falsche Buchstaben: A C D
```

```
// Spiel starten, Funktion wird von Button "Spiel starten"
ausgeführt
function startGame() {
     word = wordInput.value.toUpperCase(); // Zu erratendes Wort als
Großbuchstaben abspeichern
     if (word === "") return;   // Ist das Wort leer, brechen wir ab

     guessedWord = Array(word.length).fill("_");     // Für jeden
Buchstaben wird ein _ ins Array gefüllt

     /*
     guessedWord: Array mit _ und den schon erratenen Buchstaben
     .join(" "): Alle Inhalte des Arrays werden zu einer
Zeichenkette zusammengefügt mit dem
          Trennzeichen " "
          z.B. guessedWord = ["A", "B", "C"] -->
guessedWord.join("_") ist A_B_C
     */
     wordDisplay.innerText = "Wort: " + guessedWord.join(" ");
     wordInput.value = ""; // Textfeld wieder leer machen, damit der
Spieler es nicht sieht

     letterInput.disabled = false;   // .disabled: wenn true --> man
kann nichts eingeben, wenn false --> man kann wieder was eingeben
     wordInput.disabled = true;
}

// Buchstaben raten, Funktion wird von Button "Raten" durchgeführt
function guessLetter() {
     let letter = letterInput.value.toUpperCase();   // Eingegebenen
Buchstaben abspeichern
     letterInput.value = "";    // Textfeld für Buchstaben wieder
leeren
     let correct = false;  // Variable, um zu überprüfen, ob man
richtig oder falsch liegt

     /*
     Drei Möglichkeiten zum Abbrechen:
          Input ist leer
          Buchstabe wurde schon mal richtig geraten
          Buchstabe wurde schon mal falsch geraten
     */
     if (letter === "" || guessedWord.includes(letter) ||
wrongGuesses.includes(letter)) {
          return;    // Abbrechen
     }

     /*
     Sollte der Buchstabe im zu erratenden Wort enthalten sein, muss
es angezeigt werden:
          Das zu erratende Wort Buchstabe für Buchstabe durchgehen
          Liegt man bei einem Buchstaben richtig, ist die Variable
auf true zu stellen und im Array guessedWord zu ersetzen.
     */
     for (let i = 0; i < word.length; i++) {
          if (word[i] === letter) {
```

```
                guessedWord[i] = letter;
                correct = true;
           }
     }

     /*
     Liegt man mit dem Buchstaben falsch, kommt der Buchstabe in das
Array mit den falsch geratenen Buchstaben
     Der Text mit den falsch geratenen Buchstaben wird aktualisiert.
     */
     if (!correct) {
          wrongGuesses.push(letter); // .push(letter) --> der
Buchstabe kommt in das Array
          wrongDisplay.innerText = "Falsche Buchstaben: " +
wrongGuesses.join(", ");   // .join(", ") --> siehe oben
     }

     wordDisplay.innerText = "Wort: " + guessedWord.join(" ");  //
wordDisplay aktualisieren

     /*
     Falls in den schon erratenen Buchstaben keine _ mehr enthalten
sind, hat man alle Buchstaben erraten.
          Glückwunschtext darstellen
          Input wieder deaktivieren
     */
     if (!guessedWord.includes("_")) {
          document.getElementById("message").innerText =
"Glückwunsch! Du hast das Wort erraten.";
          letterInput.disabled = true;
     }

     /*
     Falls schon 6 Fehlversuche gemacht wurden, wird das Spiel
abgebrochen.
          Text darstellen
          Input wieder deaktivieren
     */
     if (wrongGuesses.length >= 6) {
          document.getElementById("message").innerText = "Leider
verloren! Das Wort war: " + word;
          letterInput.disabled = true;
     }
}

// Spiel zurücksetzen, wird von Button "Zurücksetzen" gestartet
function resetGame() {
     word = "";
     guessedWord = [];
     wrongGuesses = [];
     wordInput.value = "";
     letterInput.value = "";
     wordDisplay.innerText = "Wort: _ _ _ _ _";
     wrongDisplay.innerText = "Falsche Buchstaben: ";
     document.getElementById("message").innerText = "";
     letterInput.disabled = true;
```

```
      wordInput.disabled = false;
}
```

5.4 Ablaufplan

Programmiergrundlagen		
Reihenfolge	**Titel**	**Beschreibung**
1	1_Schleifen	Kennen lernen von unterschiedlichen Schleifen durch ein Kartenspiel.
3	2_Bedingungen (if/else)	Bedingungen je nach Rolle im Gesellschaftsspiel "Werwolf" erfüllen
4	3_Algorithmen (zB Sortieren, Suchen)	Algorithmen selbst entwickeln durch spielerisches Lernen. Welche Gruppe kann schneller ein Deck sortieren mit unterschiedlichen Methoden? Können sie den Algorithmus in Worte fassen? Und: Warum das genaue Formulieren von Algorithmen notwendig ist. Anleitung schreiben zum Flasche trinken.
6	4_Sets / Arrays	Arrays mit Klarsichtfolien kennenlernen. Sets mit Kleiderstange kennenlernen. Unterschiedliche Anwendungen verstehen.
Schreibübungen		
Reihenfolge	**Titel**	**Beschreibung**
2	1_Sonderzeichen	Tastaturkürzel und Sonderzeichen, Escaperoom mit Leet-Speak
5	2_Navigation	Navigation durch Windows und Dateisysteme, .zip-Dateien
7	3_Variablen verknüpfen --> Eingabe/Ausgabe	Ausgabe und Variablen sind gegeben --> Ausgabe muss richtig formuliert sein
8	4_Einrückungen und Kommentare, Syntax	Einbauen und Formatieren von fertigem Code
Programmieraufgaben		
Reihenfolge	**Titel**	**Beschreibung**
9	1_Ampel	Ampelschaltung (Schleifen mit Start/Stop Button)
10	2_Hangman	Galgenmännchenspiel im Pair Programming

5.5 Benötigtes Material

- Beamer
- Spiele: UNO, Werwölfe von Düsterwald
- ausgeschnittene Bedingungen (siehe if/else-Konditionen)
- Wiederbefüllbare Wasserflasche mit Schraubverschluss
- 4-6 Mappen mit jeweils 4 Klarsichtfolien in A6 Abteile unterteilt
- Kleiderstange und Kleiderhaken
- Malerband / Kreppband, Edding
- Mit Inhalten beschriftete Kärtchen, vorgeschriebener Lückentext (siehe Einheit Wortkärtchen)

6 Auswertung der Fragebögen nach Durchführung des Unterrichts

Im Laufe mehrerer Wochen wurde der „erste Methodenkoffer" an einer Polytechnischen Schule Wiens beforscht und ausgetestet (Phase 4: „Implementation"). Dort wird der Fachbereich „IT" wöchentlich in insgesamt 8 Stunden unterrichtet. Dazu wurden die Einheiten wie vorbereitet in einer Gruppe von bis zu 10 Schüler:innen erprobt. Die Lernenden füllten jeweils zu Beginn und Ende des Projekts, sowie am Ende jedes Unterrichtstages einen Fragebogen aus, welcher in Folge mittels deskriptiver Statistik (Stat. Lex., o. J.) analysiert wird.

Die Lehrperson, welche den Unterricht durchführte, wirkte außerdem unter der Methode der „Teilnehmenden Beobachtung" („Teilnehmende Beobachtung", o. J.) mit. So wurden besondere Vorkommnisse und Verbesserungsvorschläge, sowie geplante und tatsächlich benötigte Unterrichtszeit seitens des Lehrenden notiert. Diese Methode bietet durch das starke Näheverhältnis zu den Subjekten und zum Forschungsgeschehen, sowie durch die Perspektive einer unterrichtenden Person eine ideale Ergänzung zu den Rückmeldungen der Lernenden, da hier die bildungstheoretischen Ansätze mit dem praxisnahen Bezug vereint werden.

Es folgt nun die Auswertung dieser Fragebögen und Beobachtungen (Phase 5: „Evaluation").

6.1 Vorwissen

Bei einer einführenden Befragung der 10 Schüler:innen (8 m., 2 w., Durchschnittsalter 15 J.) wurde ersichtlich, dass es sich hierbei um eine eher heterogene Gruppe handelt, welche bisher kaum Vorerfahrungen in die Programmierung mitbringt (Abb. 14). 50% der Gruppe hat schon Erfahrungen mit Windows als Betriebssystem gesammelt und alle Schüler:innen gaben an, zumindest etwas Erfahrung mit dem Schreiben auf einer PC-Tastatur zu haben. 8 von 10 Kindern bestätigten, sich mit logischen Abläufen eher bis gut zurecht zu finden. Im Gegensatz dazu verfügte eine Mehrheit von 80% über keine oder nur geringe Vorerfahrung hinsichtlich der Programmierung.

Vorerfahrung	Programmierung	Tastaturschreiben	Logische Abläufe	Windows
Keine	3	0	1	1
Wenig	5	0	1	4
Mittel	2	0	0	0
Etwas	0	7	7	3
Viel	0	3	1	2

Abbildung 14: Tabelle Vorerfahrung Intro

Hinsichtlich der Vorstellung des Unterrichts wird meist auf das Bild der popkulturellen Darstellung der Hackerszene Bezug genommen: „Viele Zeilen [...] grüner Text" oder „diese schwarz und grüne Hacker". Außerdem wurde von vier Befragten das Programmieren von Spielen erwähnt, drei benannten auch das Programmieren von Websites. Eine Schülerin betonte in Voraussicht: „codes, stress, versagen, erfolg".

Auf die Frage, was die Schüler:innen tatsächlich lernen wollen deckten sich die Antworten mit einigen Vorstellung insofern, dass sich 4 Personen für das Programmieren von Spielen interessierten, eine für Websites sowie eine für das Programmieren einer App. Während drei Befragte angaben, sich für das Lernen verschiedener Sprachen zu interessieren, möchte ein Schüler für seine Ausbildung auf der HTL vorbereitet werden.

Abschließend gaben 8 von 10 Personen an, ein hohes Interesse in Informatik zu besitzen (4 oder 5 von 5 Punkten). Somit ist die Lernmotivation der Gruppe trotz der geringen Vorerfahrung als hoch einzustufen.

6.2 Tag 1

Verfügbare Unterrichtszeit: 300 Minuten, Geplante Unterrichtszeit: 285 Minuten, Benötigte Unterrichtszeit: 240

6.2.1 Intro

Die erste Stunde dieses Unterrichtstages wurde dazu genutzt, das Projekt näher zu erläutern. Es wurden die Eduvidual-Konten erstellt und die Intro-Umfrage abgehalten. Es zeigte sich, dass das Anlegen der Konten komplizierter als erwartet war, jedoch war nach dieser Einheit alles funktionstüchtig. Tatsächlich wurde dieser Abschnitt 10 Minuten schneller als geplant durchgeführt.

6.2.2 Schleifen

Da das Thema „Schleifen" die erste inhaltliche Auseinandersetzung mit dem Thema Programmierung darstellte, wurde hier ein wichtiger erster Eindruck geschaffen. Die

Auswertung der Fragebögen nach Durchführung des Unterrichts

Schüler:innen nahmen das Thema äußerst positiv wahr. 6 von 7 Anwesenden konnten sich am Ende des Tages noch an Teile dieses Inhalts erinnern. Dabei wurde von 4 Personen herausgehoben, das Thema mithilfe des Spiels „UNO" besonders erfreulich umgesetzt wurde. Ein Schüler merkte an, Schleifen anfangs nicht zur Gänze verstanden zu haben, dies hat sich aber bis zum Ende des Tages gebessert.

Die Lehrperson notierte sich, dass das Thema generell schnell verstanden wurde. Die Präsentation dauerte kürzer als erwartet, dementsprechend wurden statt den geplanten 70 Minuten nur 40 mit diesem Thema verbracht. Ein Schüler bat um eine Zusammenfassung, die Infografik kam hier hilfreicherweise zum Einsatz.

6.2.3 Tastaturschreiben

Laut Feedback erschien diese Einheit weniger einprägsam - nur 2 von 7 Schüler:innen konnten sich am Ende des Tages an diese Inhalte erinnern. Es wurde ebenfalls nicht explizit als positiver Inhalt angesprochen, hingegen wurde es von einem Schüler als demotivierend hervorgehoben. Ein weiterer Schüler merkte allerdings an, dass das Einführungsvideo publik gemacht werden sollte, da es zu diesem Zeitpunkt keinen vergleichbaren Inhalt gab.

Seitens des Unterrichtenden kam es sowohl zu einigen technischen Fehlern als auch Genauigkeitsfehlern von Schüler:innen. Die häufigste Fehlerquelle kam seitens EduVidual zum Vorschein, da hier einige Satzzeichen falsch übernommen oder dargestellt wurden. Auch die Übersetzungstabelle wurde durch die Formatierung teils unleserlich präsentiert. Lernende teilten außerdem mit, dass die in „Leet-Speak" geschriebenen Wörter zu unleserlich waren. Da schließlich richtige Antworten von EduVidual als falsch markiert wurden, kam es laut Lehrperson beiderseits zu einer äußerst demotivierten Einheit. Insgesamt wurden statt den geplanten 85 Minuten insgesamt 100 Minuten mit diesem Thema verbracht.

Weiters wurde diese Einheit im Laufe eines Seminars an einige Kommiliton:innen und zwei Professorinnen präsentiert, welche als hauptmerkliches Feedback die Abwesenheit eines tatsächlichen Rollenspiels anmerkten. Es sollte sich für die Lernenden durch ein entsprechendes Design tatsächlich wie ein Spiel anfühlen, weniger als eine Abschreibübung. Auch wurde die Masse an Sonderzeichen kritisiert, was die Lösungen nicht mehr als Worte lesen lässt.

6.2.4 if/else-Konditionen

2 der Lerndenden gaben an, sich an die Inhalte der if/else-Konditionen zu erinnern. Außerdem wurde von 5 Schüler:innen angegeben, dass entweder dieses Thema oder das Spielen bzw. explizit das Werwolf-Spiel besonders gut entgegengenommen wurde. Von der Lehrperson wurde lediglich kommentiert, dass die Konditionen um eine Bedingung erweitert werden könnte, dass ein bestimmtes Wort nicht genannt werden darf. Statt den geplanten 75 Minuten wurden 65 Minuten benötigt, um diese Einheit abzuhalten.

6.2.5 Abschluss

Den Rückmeldungen der Lernenden nach zu urteilen, blieb der erste Unterrichtstag äußerst positiv in Erinnerung. 5 von 7 Schüler:innen stimmten der Verständlichkeit der Inhalte zu, 6 der Teilnehmenden behaupteten durch den Unterricht ein gesteigertes Interesse an Programmierung erhalten zu haben, ebenso viele befanden die Struktur des Unterrichts als klar und nachvollziehbar und gaben an genug Unterstützung durch Lehrmaterialien und durch die Lehrkraft erhalten zu haben. Während jeweils ein Schüler rückmeldete ein Videospiel codieren, sowie eine Website programmieren zu wollen, wünschte sich ein Schüler „alles was mit programmieren zu tun hat“ für eine zukünftige Einheit.

6.3 Tag 2

Verfügbare Unterrichtszeit: 150 Minuten, Geplante Unterrichtszeit: 130 Minuten, Benötigte Unterrichtszeit: 135

6.3.1 Intro

In einer kurzen Besprechung wurden die Inhalte des ersten Unterrichtstages in fünf Minuten wiederholt. So konnten die zuletzt abwesenden Schüler:innen in Erfahrung bringen, welche Inhalte nachgeholt werden müssten.

6.3.2 Algorithmen

Während die ersten Lernenden den Infotext selbst durchlesen sollten, wurde von der Lehrperson den neuen Schüler:innen das EduVidual-Konto eingerichtet.

6.3.2.1 Sortieralgorithmus

Zu Beginn der ersten Aufgabe wurden vom Unterrichtenden folgende Regeln vorgelegt:

- Es müssen Tauschmanöver sein, es ist also nicht erlaubt, eine Karte zwischen zwei anderen zu platzieren.
- Ziel ist es, die Zahl der Tauschmanöver möglichst gering zu halten.
- Es ist von den Gruppen die Anzahl der Tauschmanöver mitzuzählen.
- Am Ende des ersten Durchlaufs sollte ein Ablaufdiagramm entstehen, dieses wird zum späteren Zeitpunkt erläutert. So sollten sich die Gruppen auf das erstmalige Sortieren konzentrieren.

Es wurde angemerkt, dass eine Gruppe übersah, ein Diagramm zu erstellen, dieses wurde am Ende aller Abläufe angefertigt. Die abgegebenen Anleitungen wurden von der Lehrperson innerhalb von 5 Minuten durchgetestet und anschließend nachbesprochen. Insgesamt hat der erste Durchlauf 10 Minuten kürzer als erwartet gedauert, die anschließenden Durchgänge benötigten wiederum 10 Minuten länger. Insgesamt dauerte die erste Aufgabe mit 75 Minuten mit 5 Minuten marginal kürzer als geplant.

6.3.2.2 Anleitung zum Wasser trinken

Die Lehrperson legte der Klasse als Ziel fest, dass der gesamte Flascheninhalt im Magen des Testsubjekts landet. Außerdem wurde erläutert, dass die Flasche nicht auf einmal geleert werden sollte, da sonst ein Abbruch bevorstünde - ähnlich einer Exception eines Programms.

Direkt wurden einige Fragen gestellt: „Sind Sie Rechtshänder? Muss man schreiben, dass man die Flasche in die Hand nimmt? Weiß man, was gegen den Uhrzeiger ist? Mit welcher Flasche trinken Sie? Dürfen wir so eine Flasche nehmen?“. Jegliche Fragestellungen wurden vom Unterrichtenden aufgeklärt. Eine Gruppe testete ihre Anleitung an einem Mitglied aus, bevor sie sie abgaben.

Die Testung der Anleitungen wurde mittels Kamera mitgeschnitten und im Anschluss nachbesprochen. Das Video zur Inspiration der Aufgabe wurde ebenfalls präsentiert.

Insgesamt wurden 43 von geplanten 40 Minuten in Anspruch genommen.

6.3.2.3 Ablaufdiagramm

In der übrigen Zeit wurden die Schüler:innen instruiert, ein Ablaufdiagramm für eine Ampel zu kreieren. Dazu wurde beispielsweise ein bereits angefertigtes Diagramm eines anderen Projekts präsentiert. Die Abgabefrist läuft an einem weiteren Unterrichtstag ab.

6.3.3 Abschluss

Seitens der Schüler:innen ist 7 von 8 Personen der Inhalt bis zum Schluss in Erinnerung geblieben. Eine Person betonte „dass man sehr genau sein muss, sonst wird es nicht funktionieren“. 5 der Lernenden ist besonders die Anleitung zum Wasser trinken positiv in Erinnerung geblieben. „Dass Herr Becker unsere Anleitung selber getestet hat und gar nicht zurückgehalten hat, dass wir sehen, wie genau wir sein müssen.“, beschrieb ein Schüler als solche Erfahrung. Es wurde ebenfalls die unübliche Art des Unterrichtens als zusagend empfunden. Eine Person wünschte sich außerdem eine von der Lehrperson erstellte Anleitung. Zwei Schüler:innen merkten an, dass das Erstellen der Diagramme als demotivierende Erfahrung entgegengenommen wurde.

Der Abstimmung entsprechend wurde der zweite Unterrichtstag von den Lernenden als positiv aufgefasst. Während nur zwei der Schüler:innen angaben, das Gelernte direkt anwenden zu können, stimmten 4 Personen zu, die Aufgabenstellungen als angemessen schwierig zu empfinden - zwei stimmten dem nicht zu, der Rest stimmte neutral ab. In den restlichen Fragen bezüglich Verständlichkeit, Interessenssteigerung, Klarheit der Struktur sowie Unterstützung durch Unterrichtsmaterial und Lehrperson stimmten jeweils 6 Personen entweder zu oder voll und ganz zu.

6.4 Tag 3

Verfügbare Unterrichtszeit: 400 Minuten, Geplante Unterrichtszeit: 370 Minuten, Benötigte Unterrichtszeit: 372

6.4.1 Navigation

Nach einer kurzen Wiederholung der letzten Einheiten zu Schleifen, Konditionen und Algorithmen folgte nun die Einheit zu Navigation und Dateitypen. Entsprechend der Umfrageergebnisse ist diese Einheit keinen Schüler:innen in Erinnerung geblieben, nachdem sich in keiner der Antworten eine Referenz darauf befindet. Einer von 9 Schüler:innen bildet dabei eine Ausnahme, welcher die Aufgabe zu den Dateitypen als positiv empfand.

Die Lehrperson beobachtete hingegen, dass einige Lernende nicht verstanden, was ein Dateityp wäre und die Aufgabenstellung zu unklar formuliert wurde. Es wurden Missverständnisse geklärt - "Nein, Spotify ist kein Dateityp“ - und die Aufgabenstellung

angepasst. Anstatt 5 sollten die Schüler:innen nun 3 Dateitypen finden, um doppelte Antworten zu vermeiden mit einem benötigten Hinweis darauf, dass nun insgesamt 3 Antworten verlangt wurden und nicht pro Kategorie. Zusätzlich führte eine deutliche Latenz der App zu Frustrationen bei den Lernenden: „Ich hab den Dateityp als erstes gehabt!".

Auch im weiteren Abschnitt zur Erstellung einer Ordnerstruktur gelang ein technischer Fehler. Da die Anleitung für Schüler:innen nicht zugänglich war, musste dies korrigiert werden. So gilt in EduVidual der Punkt „Aktivitätsanleitung" für Lehrende, der Punkt „Beschreibung" für Lernende. Ein weiteres Problem stellte die Suche der benötigten Dateien dar. Einerseits wurden beispielsweise die Dateien für Programmieraufgabe 1 („Ampel") nicht gefunden, weiters wurden diese Dateien von manchen Schüler:innen nicht heruntergeladen, sondern angezeigt. Dementsprechend sollte in Zukunft für jegliche Dateien einer Projektaufgabe eine .zip-Datei zur Verfügung gestellt werden.

Insgesamt wurden 90 statt den erwarteten 55 Minuten für dieses Thema aufgewendet, wobei die Erstellung der Ordnerstruktur mit insgesamt 55 Minuten am meisten Zeit in Anspruch nahm.

6.4.2 Arrays und Sets

„Wir spielen ein kleines Spiel, bevor wir ein Thema begonnen haben." und „Dieses Spiel mit den Karten einordnen." wurde von zwei Schüler:innen als besonders positiv hervorgehoben, insgesamt drei Personen konnten sich somit an diese Inhalte am Ende des Tages erinnern.

Seitens des Unterrichtenden wurden dieser Einheit kaum Notizen zugeordnet. Diese Effizienz spiegelt sich auch mit 47 verbrauchten statt 75 geplanten Minuten wider.

Eine Unklarheit jedoch, welche in allen Gruppen vorkam, war die Anweisung zur Einordnung auf Index 8. Hierbei wurde von jeder Gruppe die Karte an Index 7 eingeordnet, da es sich um die 8. Position der Mappe handelte.

Es wurde ebenfalls angemerkt, die Kleiderhaken zur Übung der Sets bereits im Vorhinein zu beschriften, sodass auch zwei Zahlen doppelt vorkommen würden. Ebenfalls sollten die Regeln vor Beginn der Übung deutlicher dargelegt werden.

6.4.3 Algorithmen ff.

Nach einer 8-minütigen Arbeitsphase am Vortag, hatten die Lernenden nun 43 Minuten Zeit, um die Erstellung eines Ablaufdiagramms weiterzuführen. Diese wurde von den Lernenden

gemischt aufgenommen. Ein Schüler betrachtete diese Aufgabe als positive Erinnerung, eine weitere als demotivierend, mit der Anmerkung, zwar nicht die Aufgabe selbst, doch das Lernen der Programmiersprache weiterführen zu wollen. Weitere Schüler:innen merkten beim Unterrichtenden an, dass die Aufgabe zu unklar gestellt sei und sie nicht wüssten, welche Inhalte im Ablaufdiagramm benötigt würden. Die Lehrperson verweiste darauf, nach eigenem Ermessen zu arbeiten, sodass die Ergebnisse auch korrigiert werden könnten.

6.4.4 Variablen

Insgesamt drei Schüler:innen vermerkten besonders die Einheit zu Variablen im Feedbackbogen, wovon eine diese Inhalte besonders positiv, einer negativ hervorhob.

Die Einführung durch den Infotext wurde diesmal per Beamer zentrierter an der Tafel ausgeübt. Die Funktionsweise des Spiels wurde nach einer kurzen schriftlichen Erklärung als prinzipiell verstanden wahrgenommen (Abb. 15).

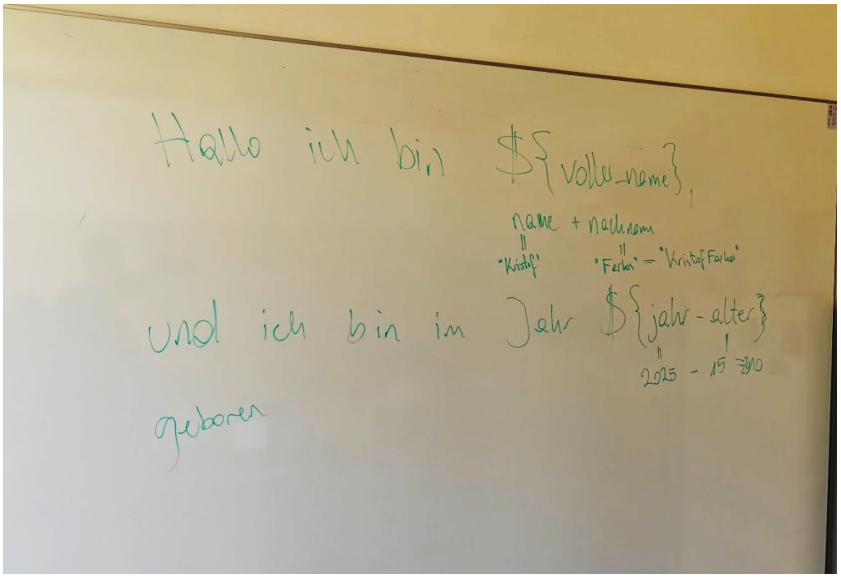


Abbildung 15: Erklärung Variablen, Spiel

Der Test zu diesem Thema warf eine technsiche Schwierigkeit bei Frage 2 auf, nachdem die Antwortmöglichkeit einen schriftlichen Fehler beinhaltete. Ebenfalls wurde festgestellt, dass die Fragestellungen in einer Testvorschau nicht sichtbar seien.

Ab Frage 3 war den Lernenden der gefragte Befehl unklar geworden. Hier wäre eine Entscheidung seitens des Unterrichtenden nötig, ob die Schüler:innen ihrem eigenen kritischen Denkprozess überlassen werden sollten, oder etwaige Beispiele doch eine sinnvolle Hilfestellung wären.

Auswertung der Fragebögen nach Durchführung des Unterrichts

Mit einer kurzen Nachbesprechung dauerte nun diese Einheit insgesamt 60 Minuten, 80 wurden eingeplant.

6.4.5 Syntax

Dem Feedbackbogen entsprechend konnte den Lernenden eine stark heterogene Meinung bezüglich der letzten Einheit des Tages entnommen werden. 6 Schüler:innen erwähnten die Aufgaben zu „Syntax" und „Programmierung" inhaltlich, 3 davon positiv, 3 negativ. Während eine Person kommentierte, nach einem langen Tag überfordert gewesen zu sein, wünschte sich eine Schülerin eine umfangreichere Aufgabe, welche gemeinsam mit der Lehrperson gelöst würde. Weiters wurde der Wunsch nach einer Musterlösung geäußert.

Dieses Feedback deckte sich mit den Notizen des unterrichtenden Lehrers. Durch mangelnde Einleitung verbreiteten sich einige technische Fehler in der Klasse. Einerseits wurde die Ordnerstruktur noch nicht für diese Übung angepasst, sowie kam es zu Problemen mit dem korrekten Öffnen der Datei „script.js". Es wurde angemerkt, das Programm „notepad++" noch vor dieser Einheit bereitzustellen, um dieser Schwierigkeit zu entgehen.

Alsbald die Lernenden zur tatsächlichen Programmierung einsatzfähig wurden herrschte große Unklarheit über die korrekte Position des Codes, sowie dessen Inhalts. Nach diesem Feedback und einer kurzen Pause wurden die Schüler:innen per Beamer angeleitet. So konnte die Klasse nun weiterarbeiten, wodurch sich ein neuer Fehler bemerkbar machte: Eine Schülerin öffnete die falsche „index.html"-Datei - nicht jene aus der eigens erstellten Ordnerstruktur, sondern aus dem Download-Ordner. Ein weiterer Schüler lud die Dateien mehrfach herunter, sodass die Datei „index.html" die Datei „script.js" und nicht die bearbeitete „script (2).js" aufrief.

Auch syntaktische Fehler machten sich in der Klasse breit: Während ein Schüler runde statt geschwungenen Klammern verwendete, schrieb ein weiterer die Funktion „function start () {...}" mehrfach unter der Annahme, diese Funktion für jeden einzelnen Befehl zu benötigen.

Nach 85 Minuten dieser Einheit wurde die restliche Zeit dazu verwendet, diese Aufgabe sowie das Ablaufdiagramm zur Aufgabe „Ampel" fertigzustellen. Währenddessen wurde vom Unterrichtenden zur Unterstützung jeglicher Fragestellungen eine Musterlösung der Programmieraufgabe an die Wand projiziert.

Insgesamt wurden 120 Minuten für diese Inhalte aufgewendet, also 50 Minuten mehr als ursprünglich geplant.

6.4.6 Abschluss

Der Umfrage konnten für Tag 3 durchwegs positive Rückmeldungen entnommen werden. 5 von 9 Personen stimmten zumindest zu, die Unterrichtsinhalte seien verständlich gewesen - eine Person stimmte dem nicht zu. Sechs Personen empfanden die Schwierigkeit der Aufgabenstellungen als angemessen und die Struktur des Unterrichts als nachvollziehbar. Jeweils 7 Personen meldeten, dass sie das Gelernte direkt anwenden konnten und der Unterricht das Interesse an Programmierung steigerte - 5 davon stimmten voll und ganz zu. 8 Personen stimmten ab, genug Unterstützung durch die Lehrkraft erhalten zu haben.

6.5 Tag 4

Verfügbare Unterrichtszeit: 300 Minuten, Geplante Unterrichtszeit: 275 Minuten, Benötigte Unterrichtszeit: 275

6.5.1 Intro

Wie in den letzten Unterrichtstagen wurden anfangs die bisher erarbeiteten Inhalte wiederholt. Die Lehrperson notierte hierbei, dass die Festigung durch eine spielerische Annäherung deutlich besser funktionierte.

6.5.2 Installation Notepad++

In einem gemeinsamen Effort wurde die Installation mit dem Unterrichtenden durchgeführt. Anschließend sollten die Lernenden die zur Verfügung gestellten Diagramme als Wiederholung und Vorbereitung auf die kommende Aufgabe studieren. Es wurde außerdem eine Musterlösung des Codes präsentiert, um den Schüler:innen Erwartungen zu setzen.

6.5.3 Ampel

In Folge wurde der Code zur Aufgabe „Ampel" von der Lehrperson vorprogrammiert. Eine erste Hürde stellte auch hier wieder das Dateimanagement dar, da nicht alle ihre „index.html"-Datei öffnen konnten bzw. die falsche Datei öffneten. Manche Lernenden verwendeten ihre vorher erstellte Ordnerstruktur nicht.

Weitere Fehler betraf ein ungenaues Abschreiben, beispielsweise durch das Schließen eines Kommentars mittels „/*“, der Absenz von Anführungszeichen oder der Missachtung von Groß- und Kleinschreibung.

Die Lehrperson weist außerdem auf ein Unverständnis von IDs und Klassen, sowie der Funktionsweise von geschlossenen Klammern hin. Diese sollten möglicherweise in Rollenspielen sowie in der Einheit zur Syntax besser beleuchtet werden.

Im weiteren Verlauf der Einheit sollten die Funktionen „Ablauf“ sowie „stoppeAmpel“ von den Schüler:innen selbstständig erarbeitet werden. Dies führte unter anderem zu Frustrationen, da der Code bei einem Klick auf den „Start“-Button nicht aufgerufen wurde - die Verknüpfung dieser Elemente würde erst im späteren Verlauf stattfinden. Weiters verwendete ein Schüler keine Schleifen für diesen Teil des Programms, da ihm die Funktionsweise entfiel.

Als schließlich die Buttons mit den Funktionen - nunmals wieder unter Anleitung der Lehrperson - verknüpft werden sollten, erläuterte der Unterrichtende erst die Abläufe der benötigten Funktionen „starteAmpel“ und „stopAmpel“.

Im weiteren Verlauf wurden noch zwei verschiedene Funktionsweisen kommentiert. Befehle zum Pausieren eines Programms sind in der verwendeten Sprache „javascript“ wesentlich komplizierter und über mehrere Zeilen einzubauen. Dies führte auch zu einigen Kapitulationen von Schüler:innen, wie man dem Feedback entnehmen kann. 50% der Befragten gaben das Programmieren der Timer als verwirrendes oder demotivierendes Erlebnis des Unterrichtstages an. Somit wurde am Tag des Unterrichts in dieser Aufgabe das Blinken des grünen Lichts aus der Anforderung entfernt, da dieser Ablauf in Verknüpfung mit den Timern zu einer erhöhten Komplexität des Programms führen würde. Nach einer erneuten Konkretisierung der Timer sollten diese nun von den Lernenden selbstständig eingebaut werden.

6.5.4 Abschluss

Den Antworten des Fragebogens ist vermehrt die Frustration der Schüler:innen zu entnehmen. Auf die Nachfrage eines positiven Erlebnisses antworteten 3 von 8 „kaum was“, „keines“, „nix“. 4 Personen hingegen bekundeten den Anfang der Programmierung als erfreulich. Zwei Personen kommentierten, von der Masse an Neuheiten überfordert gewesen zu sein. Eine Schülerin wünschte sich, die Inhalte näher erläutert zu bekommen.

Diese gemischten Ansichten zeigten sich auch in Bezug auf die Verständlichkeit der Inhalte, die Schwierigkeit der Aufgabenstellungen, die Interessenssteigerung in Programmierung, die Anwendungsmöglichkeiten des Gerlernten sowie die Struktur des Unterrichts. Lediglich die Unterstützung durch Lernmaterial und Lehrkraft wurden überwiegend positiv entgegengenommen.

6.6 Tag 5

Verfügbare Unterrichtszeit: 400 Minuten, Geplante Unterrichtszeit: 365 Minuten, Benötigte Unterrichtszeit: 326

6.6.1 Hangman

Nachdem die erste Unterrichtstunde des Tages zur selbstständigen Wiederholung sowie Vervollständigung aller bisherigen Abgaben freigegeben wurde, erfolgte eine Einführung in die abschließende Aufgabe „Hangman", in welcher unter anderem das bereits erstellte Ablaufdiagramm sowie die Funktionsweise des „Pair Programming" mithilfe der Inspiration eines Videos (Bran Van der Meer 2023) erläutert wurde.

Erneut stellte die korrekte Verwendung der Ordnerstruktur eine erste Hürde dar. Eine Gruppe konnte die Dateien aus der .zip-Datei nicht in den dafür vorgesehenen Ordner kopieren, eine andere Gruppe bearbeitete die Dateien direkt im .zip-Ordner und konnte die so erstellten Codezeilen nicht austesten. Eine andere Gruppe verlor ihren Code zur Gänze. Dies demotivierte die Beteiligten sichtlich, wie auch deren Feedback zu entnehmen ist. Hier empfiehlt es sich, diese Schritte vorerst zu präsentieren.

Weiters konnten manche Schüler:innen nicht eruieren, an welcher Stelle welcher Code zu schreiben wäre. Eine Gruppe ging fälschlicherweise der Annahme voraus, den Code aus der Anleitung ergänzen zu müssen, eine andere ergänzte den Code mit Erklärungen ohne deren Kommentierung. Demnach wurde die Anleitung nochmals näher erläutert.

Ebenfalls wurde diese Einheit von syntaktischen Fehlern geprägt, etwa fehlenden Einrückungen, Strichpunkten oder der korrekten Benennung von Variablen und Funktionen.

Andere Gruppen zeichneten sich durchwegs erfolgreicher aus, indem sie die gesamte Übung bereits nach 80 bzw. 130 der insgesamt 215 geplanten Minuten fertigstellten. Weitere Gruppen konnten sie nach 165 sowie 215 Minuten beenden.

Schüler:innen meldeten einige Verbesserungsvorschläge, etwa einen gemeinsamen Einstieg. Zwei Navigatoren verstanden den Inhalt ihrer Rolle nicht zur Gänze.

6.6.2 Abschluss

Bei einer abschließenden Leistungsfeststellung, in welcher sowohl theoretische Inhalte, als auch praktische Anwendungen wie Fehlersuchen oder Ausgaben erfragt wurden, konnten die Schüler:innen ein Ergebnis von durchschnittlich 9,55 von 17 Punkten erzielen, mit einer Spannweite von 7 bis 12 Punkten und einer Standardabweichung von ~1,54 Punkten.

Nachdem doch einige der Schüler:innen von diversen Fehlern geplagt wurden, vermeldeten 8 von 10 einen positiven Schluss des Projekts. Dabei hoben drei besonders die Methode des „Pair Programming" hervor.

Auch die anderen erfragten Aspekte des Unterrichts wurden durchwegs positiv benannt. Die Interessenssteigerung zur Programmierung erhielt 7, die direkte Anwendung des Gelernten, Struktur des Unterrichts sowie Unterstützung durch Material 8, die Unterstüztung durch die Lehrperson sowie die Verständlichkeit der Inhalte 9 Zustimmungen von Schüler:innen. Lediglich die Schwierigkeit der Aufgabenstellungen empfanden 5 als angemessen, 4 Personen stimmten diesem nicht zu.

6.7 Abschluss

Durch eine abschließende Befragung aller 10 Teilnehmenden bekamen die Schüler:innen die Möglichkeit, Feedback zum sämtlichen 5-tägigen Workshop abzugeben (Abb. 16, 17, 18). Während zwei Personen ihre Vorstellung hinsichtlich Programmierung nicht änderten, merkten drei Schüler die benötigte Genauigkeit an. Es wurde auch die Schwierigkeit und Langweiligkeit angesprochen, zwei Schüler erzählten doch auch von Erfolgserlebnissen, sobald der Code funktionierte.

8 Schüler:innen gaben außerdem an, weitere Inhalte im Programmierunterricht lernen zu wollen, unter anderem ein Timer, Chat, Spiele oder überhaupt das Lernen einer anderen Sprache, etwa Java.

Bezüglich der Erfahrung diverser Fähigkeiten steigerten sich die Selbsteinschätzungen deutlich. Auch die Inhalte des Workshops wurden durchwegs positiv entgegengenommen, mit Ausnahme der Aussage „Die Übungen haben mir geholfen, das Gelernte zu festigen." – hierfür stimmten 4 von 10 Personen nicht zu. Die größte Unzufriedenheit festigte sich mit 3

von 10 Stimmen im Schwierigkeitsgrad des Kurses. Die genauen Abstimmungen finden sich in den folgenden drei Tabellen.

Erfahrung	Programmierung	Tastaturschreiben	Logische Abläufe	Windows
Keine	0	0	0	0
Wenig	1	0	0	3
Mittel	0	0	0	0
Etwas	9	3	7	3
Viel	0	7	3	4

Abbildung 16: Tabelle Erfahrung Outro

Aussagen	Ich stimme voll und ganz zu	Ich stimme zu	Ich stimme nicht zu
Die Inhalte des Kurses waren für mich verständlich aufbereitet.	3	6	1
Die Materialien (z.B. Videos, Texte, Aufgaben) haben mir beim Lernen geholfen.	1	9	0
Die Aufgabenstellungen waren klar formuliert.	1	8	1
Die Übungen haben mir geholfen, das Gelernte zu festigen.	1	5	4
Die Plattform war einfach zu bedienen.	2	6	2
Der Zugriff auf Inhalte hat problemlos funktioniert.	1	3	6

Abbildung 17: Tabelle Aussagen Outro, keine Abstimmungen für Möglichkeiten "neutral" und "Ich stimme überhaupt nicht zu"

Zufriedenheit bzgl.	Sehr zufrieden	Zufrieden	Unzufrieden
Angaben	0	9	1
Aufgaben	1	7	2
Erklärungen und Methoden	3	5	2
Zeitmanagement	1	7	2
Schwierigkeitsgrad	0	7	3

Abbildung 18: Tabelle Zufriedenheit Outro, keine Abstimmungen für Möglichkeit "Sehr unzufrieden"

Die Lernenden schätzten ihr allgemeines Interesse in Programmierung mit durchschnittlich 3,9 von 5 Punkten ein, bei einer Spannweite von 2 bis 5 Punkten und einer Standardabweichung von ~1,044 Punkten.

Weiters würden sich einige der Befragten wünschen, die Inhalte näher zu erklären. Dies könnte einer Rückmeldung zufolge in einer gemeinsamen Einführung zum Code stattfinden. Auch die Timer der Übung „Ampel“ sollten gemäß zweier Schüler:innen geändert werden. Ein Schüler wünschte sich Referenzmaterialien, welche mit dem zu erarbeitendem Code nichts zu tun hat.

Abschließend wünschten sich einige, die Erklärung seitens der Lehrperson beizubehalten, ein weiterer Schüler möchte die Dateien und Aufgaben erhalten, einer Schülerin das Programmieren der Spiele.

7 Verbesserung Methodenkoffer

In Folge der abgehaltenen Einheiten wurde auf Basis des Feedbacks des Lehrenden und der Lernenden jede Einheit überarbeitet. Weiters wurde die Einheit „Variablen" in die „Grundlagen der Programmierung" verschoben. Der Teil „Schreibtraining" wurde in „PC-Grundlagen" umbenannt, um diesen inhaltlich treffender zu formulieren.

Es wurden außerdem für jede Einheit ein Word-Dokument bearbeitet bzw. angelegt, sodass dieses alle benötigten Infos und Texte beinhaltet. Weitere Elemente wurden nachbearbeitet, um unabhängig von den technischen Vorgaben von EduVidual arbeiten zu können. Somit ist es möglich den Unterricht auch auf andere Plattformen (MS Teams, Padlet, Google Classroom, etc.) zu verlegen.

Die benötigten Materialien änderten sich nicht.

7.1 Intro

In einer Einführung gilt es seitens der Lehrperson zwei Aufgaben zu erfüllen. Einerseits muss der Zugang zur Lernumgebung geschaffen werden. Idealerweise ist in dieser die Freigabe der Inhalte automatisch und zeitabhängig möglich. Diese ist entsprechend vorzubereiten und im Laufe der Einheiten anzupassen. Es sollte auch kontrolliert werden, ob alle Teilnehmenden diesen Zugang verwenden können. Weiters sollte auf allen Arbeitsgeräten das Programm „notepad++"[6] („Notepad++", o. J.) installiert werden, um in späteren Einheiten einen möglichst reibungsfreien Einstieg zu ermöglichen.

7.2 Grundlagen der Programmierung

7.2.1 Schleifen

Aufgrund der deutlich positiven Rückmeldungen wurde das Konzept dieser Einheit beibehalten. Die einzige Änderung betraf die Anpassung der geplanten Zeiten, somit es von 70 Minuten auf 40 Minuten gekürzt wurde.

[6] Vgl. Texteditor

7.2.2 if/else-Konditionen

Da auch diese Einheit mehrfach positiv bewertet wurde, sind an dieser Stelle kaum Anpassungen vorgenommen worden. Die benötigte Zeit wurde von 75 auf 65 Minuten korrigiert. Es wurden die Konditionen zum Spiel „Werwolf" um folgende Bedingung erweitert:

> *Wenn du ein Werwolf bist, darfst du das Wort „Werwolf" nicht verwenden, ansonsten darfst du das Wort „Dorfbewohner" nicht verwenden.*

7.2.3 Algorithmen

In dieser Einheit wurden die Abschnitte feiner differenziert sowie deren Dauer angepasst. Weiters sollte zwischen dem ersten und zweiten Ablauf zum Sortieralgorithmus das Ablaufdiagramm erläutert werden. Dazu kann das Musterbeispiel zum Ablauf der Programmieraufgabe „Ampel" herangezogen werden.

Zudem wurden die von der Lehrperson aufzustellenden Regeln schriftlich festgehalten. Die Übung zur Anleitung des Wasser Trinkens wurde um folgende Regeln ergänzt:

> *Sollte der Flascheninhalt ohne Absetzen geleert werden müssen, so wird der Versuch abgebrochen, als wäre es ein Fehler im Code.*
>
> *Die Anleitung soll vor einer Abgabe an einem Gruppenmitglied getestet werden.*

Es wurden die Materialien um eine vorgefertigte Anleitung ergänzt, welche ebenfalls ausgetestet werden könnte.

Im Anschluss sollte ein Ablaufdiagramm zur Programmieraufgabe „Ampel" erstellt werden. Da zur eheren Erklärung bereits eine Musterlösung herangezogen wurde, wird hier die Aufklärung etwaiger Unklarheiten erwartet. Es könnte dennoch als Referenz zur Beantwortung diverser Fragen genommen werden, solange dieses nicht unter den Lernenden verbreitet wird.

7.2.4 Arrays und Sets

Die Anleitung dieser Einheit wurde um den Hinweis ergänzt, einige Kleiderhaken bereits im Vorhinein zu beschriften. Weiters wurden die Zeiteinheiten angepasst.

7.2.5 Variablen

In dieser Einheit wurde neben der Anpassung der benötigten Zeit auch ein besonderer Fokus auf die Erklärung von Sonderzeichen, Befehlen und der Wichtigkeit von Groß- und Kleinschreibung gelegt.

Um der Demotivierung durch einen Test am Ende der Einheit entgegenzuwirken, wurde eine eigene Website[7] erstellt, in welcher das Quiz übersichtlicher gestaltet wurde. So können alle Eingaben auf einmal von den Lernenden überprüft und wenn nötig angepasst werden. Es wurden auch die Quizfragen entsprechend angepasst. Die Website ist unter http://variablentest.atwebpages.com/ erreichbar, sowie als Teil des Materials selbst hostbar.

7.3 PC-Grundlagen

7.3.1 Tastaturschreiben

Diese vielseitig kritisierte Einheit wurde grundlegend verändert. Um die Schüler:innen in ein tatsächliches Rollenspiel eines Escaperooms zu, ist das Material der Einheit entsprechend neu designed worden. Weiters wurde die Übersetzungstabelle angepasst, sodass nur noch 10 von 26 Buchstaben als Sonderzeichen darzustellen sind.

Um technische Fehler seitens EduVidual zu vermeiden, ist sowohl der Abschreibtext, als auch die „Gateway-Challenge" als Website programmiert und designed worden[8]. Diese sind sowohl über eine URL abrufbar (http://abschreibtext.atwebpages.com/ sowie http://g4t3w4y.atwebpages.com/), als auch als Teil des Materials selbst hostbar.

Damit die Schüler:innen die Challenge starten können, muss nun auf der ersten Website ein Abschreibtext genau eingegeben und überprüft werden. Ist der Text korrekt eingegeben, so schwärzt sich der Bildschirm und es erscheint eine Nachricht sowie eine Verlinkung zur zweiten Website. Auf dieser sind nun 5 Codes einzugeben, welche automatisch von der Website überprüft werden. Die Lösungen erhält man durch im Raum verteilte Rätsel (siehe Anhang).

[7] Diese Website ist mit Unterstützung von KI entstanden.
[8] Diese Websites sind mit Unterstützung von KI entstanden.

Sind alle Codes richtig eingegeben, so erscheint ein Banner mit den Worten „Access granted“. Damit ist vom Unterrichtenden einfach zu prüfen, welche Lernenden bereits die Übung abgeschlossen haben.

7.3.2 Navigation

Um jegliche Irrtümer zu vermeiden, wurde die Aufgabe „Dateitypen“ insofern überarbeitet, dass für jede der drei Spalten ein Beispiel bereits angegeben ist. Außerdem müssen nun von den Lernenden insgesamt nur 3 verschiedene Dateitypen sowie deren Kurzbeschreibung angegeben werden mit besonderem Hinweis darauf, dass es insgesamt 3 Stück seien sollten, nicht pro Spalte.

Zur Aufgabe der Ordnerstruktur sollten den Lernenden alle benötigten Dokumente in einer .zip-Datei zur Verfügung gestellt werden. Dazu zählen nun auch jene Dokumente und Ordner, welche erst in den weiteren Tagen benötigt werden. Das soll zukünftige Problemstellungen durch die Verwendung falscher Dateien vermeide. So ist nun folgende Ordnerstruktur zu erstellen (Abb. 19).

Abbildung 19: Einheit Navigation - Ordnerstruktur

Die korrekte Struktur ist durch die Lehrperson bei allen Schüler:innen mittels „tree“-Befehl in der Commandline zu überprüfen. Eine Anleitung hierzu wurde in der entsprechenden Vorbereitung vermerkt.

7.3.3 Syntax

Um die Inhalte des Infotextes auch anwendbar zu machen, wurde eine Vorübung erstellt. In dieser gilt es, den Lernenden besonders das Kommentieren, die Bedeutung geschwungener Klammern sowie die Einrückung näher zu bringen. Somit wird in 3 Stufen ein Code vorprogrammiert, welcher von den Lernenden abgetippt wird. Die Funktionsweise ist hierbei irrelevant, da es rein um die Formatierung und Syntax geht. Der Unterrichtende sollte nach jeder Stufe überprüfen, ob die Ergebnisse der Schüler:innen korrekt sind.

```
const name = "Max";
const alter = 29;
const aktuelles_jahr = 2026;

function berechneGJ (alter, aktuelles_jahr, name) {
```

```
      let gj = aktuelles_jahr - alter;
      if (gj > 0) {
            alert(´$(name) ist $(gj) geboren.´);
      } else {
            alert(´FEHLER´);
      }
}

/*
Stufe 1
---

const name = "Max";
const alter = 29;
const aktuelles_jahr = 2026;

function berechneGJ () {

}

Stufe 2
---

const name = "Max";
const alter = 29;
const aktuelles_jahr = 2026;

function berechneGJ (alter, aktuelles_jahr, name) {
      let gj = aktuelles_jahr - alter;
}

Stufe 3
---

siehe oben
*/
```

Eine der häufigsten Fehlerursachen betraf die Bearbeitung der korrekten Datei. Da diese bereits durch die Einheit „Navigation“ im richtigen Ordner liegen sollten und durch die Lehrperson überprüft wurden, ist hierfür nur mehr das Öffnen der korrekten Datei notwendig. Dies geschieht mittels Anleitung des Unterrichtenden über einen Beamer sowie anschließender Kontrolle. Da das Programm „notepad++“ bereits installiert ist, wird die Bearbeitung des Codes etwa durch automatische Einrückung und Einfärbung der Befehle nun leichter fallen.

Um den Schüler:innen den Einstieg in die Bearbeitung zu erleichtern, wurde der Code bereits kommentiert und die ersten Zeilen geschrieben:

```
function start () {
      // HIER DEINEN CODE SCHREIBEN
      let name = "Anna"; // Variable name hat den Inhalt "Anna"
```

```
      const pi = 3.14; // Konstante pi hat den Inhalt 3.14
      alert(`Ich heisse ${name} und Pi ist ${pi} gross.`); // Ausgabe
mit Variablen name und pi

      // HIER IST DEIN CODE ZU ENDE
}
```

7.4 Programmieraufgaben

7.4.1 Ampel

Um einige Unverständnisse aufzuklären, wurde zu Beginn der Einheit ein vorarbeitendes Intro hinzugefügt. In diesem werden die Themen IDs und Klassen, Verknüpfung von Buttons mit Funktionen sowie Timer nähergebracht.

Eine weitere Änderung betrifft die Entscheidung die Methode des „pair programming“ bereits in dieser Aufgabe anzuwenden. So werden nun die Lernenden in Teams aufgeteilt und die Rollen sowie der Ablauf mittels einer Präsentation erläutert. Ähnlich wie in der Einheit „Syntax“ sollte nun Zeit investiert werden, gemeinsam mit den Lernenden die korrekten Dateien zu öffnen. Anschließend sollten die Pärchen anhand einer neu erstellten Anleitung die Aufgabe programmieren. Dabei wurde in der Anleitung im Vorwort hervorgehoben, dass jede benötigte Codezeile beinhaltet ist. Außerdem wird durch entsprechende Markierung ersichtlich gemacht, ab welcher Codezeile die Buttons auch funktionsfähig wären.

Der Einstieg in diese Aufgabe sollte außerdem gemeinsam erfolgen. Dies ist nach Ermessen der Lehrperson zu erfolgen.

Eine weitere Änderung betrifft die Funktionsweise der Ampeln. Da die Timer den Code verkomplizieren, wurde das Blinken des grünen Lichts entfernt. Es wurde außerdem eine effizientere Methode gefunden, die Timer umzusetzen.

7.4.2 Hangman

Um eine möglichst erfolgreiche Arbeitsweise zu gewährleisten, dürfen sich die Schüler:innen in dieser Übung frei entscheiden, in welcher Methode sie arbeiten - „pair programming“ oder selbstständig. Anstatt die Dateien gemeinsam zu öffnen, sollte in dieser abschließenden Aufgabe nur noch die Kontrolle der geöffneten Dokumente stattfinden. Auch in der Anleitung dieser Aufgabe wurde angemerkt, dass kein zusätzlicher Code benötigt werde.

7.5 Abschluss

7.5.1 Leistungsfeststellung

Die Inhalte der Leistungsfeststellung sind in einer .docx-Datei einzusehen und wurden nicht angepasst. Diese ist in allen möglichen Programmen zur Abhaltung eines Tests oder Quizes durchführbar (MS Forms, Kahoot, etc.).

7.5.2 Ablaufplan

Grundlagen der Programmierung		
Reihenfolge	**Titel**	**Beschreibung**
1	1_Schleifen	Kennen lernen von unterschiedlichen Schleifen durch ein Kartenspiel.
3	2_Bedingungen (if/else)	Bedingungen je nach Rolle im Gesellschaftsspiel "Werwolf" erfüllen
4	3_Algorithmen (zB Sortieren, Suchen)	Algorithmen selbst entwickeln durch spielerisches Lernen. Welche Gruppe kann schneller ein Deck sortieren mit unterschiedlichen Methoden? Können sie den Algorithmus in Worte fassen? Und: Warum das genaue Formulieren von Algorithmen notwendig ist. Anleitung schreiben zum Flasche trinken.
6	4_Sets / Arrays	Arrays mit Klarsichtfolien kennenlernen. Sets mit Kleiderstange kennenlernen. Unterschiedliche Anwendungen verstehen.
7	5_Variablen verknüpfen --> Eingabe/Ausgabe	Ausgabe und Variablen sind gegeben --> Ausgabe muss richtig formuliert sein
PC-Grundlagen		
Reihenfolge	**Titel**	**Beschreibung**
2	1_Tastaturschreiben	Tastaturkürzel und Sonderzeichen, Escaperoom mit Leet-Speak
5	2_Navigation	Navigation durch Windows und Dateisysteme, .zip-Dateien
8	3_Einrückungen und Kommentare, Syntax	Einbauen und Formatieren von fertigem Code

Programmieraufgaben		
Reihenfolge	**Titel**	**Beschreibung**
9	1_Syntax und IDs	Ampelschaltung (Schleifen mit Start/Stop Button)
10	2_Hangman	Galgenmännchenspiel im Pair Programming

8 Diskussion, Implikation und Conclusio

Auch in einer renovierten Fassung des Progammierunterrichts konnten grobe Defizite festgestellt werden. Sei es die Fehleranfälligkeit und daraus folgende Frustration in Abschreibübungen oder ein selbstständiger Einstieg in das tatsächliche Coding. Diese Defizite und Lücken galt es zu Füllen.

Eine Vielfältigkeit an bereits erprobten und studierten Methoden boten dafür eine didaktische Grundlage. Die Eigenerkenntnis, das Coding vom inhaltlichen Verständnis zu trennen war besonders ausschlaggebend.

Im Laufe der Arbeit konnte die benötigte Anpassung des Unterrichts mithilfe von erneutem Feedback der Lernenden, aber auch detaillierter Notizführung des Lehrenden geschehen. Auch die Rückmeldungen von Studierenden und Kolleg:innen wurden in das finale Produkt eingebaut. Dabei wurden die problematischen Einheiten von Grund auf erneuert und auf etwaige missglückte Inhalte verzichtet. Jene Einheiten, welche als größtenteils positiv wahrgenommen wurden, wurden ebenfalls dem Feedback entsprechend angepasst und somit beibehalten.

Weiters sind nun alle Materialien derartig vorbereitet, sodass diese von Lehrpersonen auf jeglichen Plattformen weiterverwendet werden können. Dies dient vor allem der schulunabhängigen Anpassung an alle digitalen Lehr-/Lernsysteme, welche an den hoch diversen Bildungseinrichtungen Österreichs vorzufinden sind.

„Inwiefern ist der Programmierung-Methodenkoffer für Schüler:innen der FMS geeignet, einen Einstieg in die Programmierung in einem Zeitraum von 32 Wochenstunden zu bieten?“ lautete die Forschungsfrage. Diese Arbeit bietet einen beispielhaften Lösungsansatz, indem Methoden und Inhalte in einem zyklischen Prozess reflektiert, analysiert, aufgearbeitet, weiterverwendet und angepasst wurden, wodurch der finale „Methodenkoffer“ entstanden ist. Dieser zeigt schließlich auf, wie die theoretischen Grundsätze des ADDIE-Modells in die Bildung der Informatik übertragen werden können und welche bestehenden Prinzipien erhalten bleiben bzw. welche neuen Prinzipien den Unterricht verbessern.

Die finalen Einheiten dürfen allerdings nicht als ein perfekt aufbereiteter Unterricht angesehen werden, da jede Klasse, jede Schulstufe sowie jedes mögliche Klientel an verschiedenen Schulen sehr differenziert arbeiten und lernen. Für die Forschung dieser

Arbeit bestanden einige Limitationen, insofern der erste „Methodenkoffer" nur an 10 Personen sowie an nur einer einzigen Schulform ausgetestet werden konnte. Jedoch ist es möglich diese Unterrichtsvorbereitungen als Basis für den eigenen Unterricht zu verwenden und nach jeglichen Bedürfnissen anzupassen. Um die Forschung weiterzuführen, könnte der finale „Methodenkoffer" an weiteren Schulformen, in verschiedenen Schulstufen sowie in größerem Ausmaß getestet werden.

9 Literaturverzeichnis

„ADDIE Model". o. J. *Information Technology*. Zugegriffen 24. August 2025. https://www.uwb.edu/it/addie.

Alexander Ruf, Andreas Mühling, und Peter Hubwieser. 2014. *Scratch vs. Karel - Impact on Learning Outcomes and Motivation*. https://dl.acm.org/doi/epdf/10.1145/2670757.2670772.

Andrea Burda-Zoyke. 2017. *Design-Based Research in der Berufs- und Wirtschaftspädagogik – Rezeption und Umsetzungsvarianten.* Bwp@ Berufs- und Wirtschaftspädagogik. https://www.bwpat.de/ausgabe33/burda-zoyke_bwpat33.pdf.

Andrew DeBell. 2020. „What Is the ADDIE Model of Instructional Design?" *Water Bear Learning*, Januar 6. https://waterbearlearning.com/addie-model-instructional-design/.

Bran Van der Meer, Reg. 2023. *Pair Programming best-practices*. 11:02. https://www.youtube.com/watch?v=E4cg5mmvpwo.

Brian Hanks, Sue Fitzgerald, Renée McCauley, Laurie Murphy, und Carol Zander. 2011. „Pair programming in education: a literature review". *Computer Science Education* 21 (2): 135–73. https://doi.org/10.1080/08993408.2011.579808.

Dieter Euler und Peter Sloane F. E. 2014. *Design-based research*. Franz Steiner Verlag,.

Eckart Modrow und Kerstin Strecker. 2016. „5. Programmieren". In *Didaktik der Informatik*. De Gruyter Oldenbourg. https://doi.org/doi:10.1515/9783486720112-007.

Galgenmännchen - Alternative Ways. o. J. Zugegriffen 28. Dezember 2025. https://alternativeways.eu/de/galgenmaennchen/.

Gemini (Gemini). o. J. „Google Gemini". Zugegriffen 2. April 2026. https://gemini.google.com.

Get IT (get in IT). o. J. „Welche Programmiersprache solltest Du lernen?" Zugegriffen 27. Dezember 2025. https://www.get-in-it.de/magazin/bewerbung/it-skills/welche-programmiersprache-lernen.

Ira Diethelm, Christina Dörge, Claudia Hildebrandt, und Carsten Schulte, Hrsg. 2010. *Didaktik der Informatik: Möglichkeiten empirischer Forschungsmethoden und Perspektiven der Fachdidaktik; 6. Workshop der GI-Fachgruppe „Didaktik der Informatik"; 16. - 17. September 2010 in Oldenburg*. GI-Edition Proceedings 168. Workshop Didaktik der Informatik. GI.

It-Agile GmbH (it-agile GmbH). o. J. „Was ist Pair-Programming? Agile Development." Zugegriffen 13. August 2025. https://www.it-agile.de/agiles-wissen/agile-entwicklung/was-ist-pair-programming/.

Jo E. Hannay, Tore Dybå, Erik Arisholm, und Dag I. K. Sjøberg. 2009. „The effectiveness of pair programming: A meta-analysis". *Information and Software Technology* 51 (7): 1110–22. https://doi.org/https://doi.org/10.1016/j.infsof.2009.02.001.

Josh Darnit, Reg. 2017. *Exact Instructions Challenge - THIS is why my kids hate me. | Josh Darnit*. https://www.youtube.com/watch?v=cDA3_5982h8.

Marcos J. Gomez, Marco Moresi, und Luciana Benotti. 2019. *Text-based Programming in Elementary School: A Comparative Study of Programming Abilities in Children with and without Block-based Experience*.

Mattel Shop (Mattel Shop). o. J. „UNO-Karten - Das Kartenspiel für die ganze Familie". Zugegriffen 28. Dezember 2025. https://shopping.mattel.com/de-de/collections/uno.

Max Becker, Reg. 2026. *Sonderzeichen auf einer QWERTZ Tastatur*. 03:21. https://www.youtube.com/watch?v=MfN01gGj13I.

„Notepad++". o. J. Zugegriffen 2. März 2026. https://notepad-plus-plus.org/.

Patrick Felicia. 2014. *Game-Based Learning: Challenges and Opportunities*. Cambridge Scholars Publishing.

Patrick Korber und Renate Motschnig. 2021. „The effects of pair-programming in introductory programming courses with visual and text-based languages". *2021 IEEE Frontiers in Education Conference (FIE)*, 1–9. https://doi.org/10.1109/FIE49875.2021.9637186.

Programmieren lernen, Reg. 2022. *PROGRAMMIEREN LERNEN - So startest du!* https://www.youtube.com/watch?v=JG-gofSMWGw.

Ralf Romeike und Sven Jatzlau. 2017. *Herausforderungen durch neue Programmierkonzepte in blockbasierten Programmiersprachen*.

Son Le und Peter Weber. 2011. *Game-Based Learning*. https://l3t.tugraz.at/index.php/LehrbuchEbner10/article/view/79/38.

Stat. Lex. (Statista Lexikon). o. J. „Deskriptive Statistik - Statista Definition". Zugegriffen 2. April 2026. https://de.statista.com/statistik/lexikon/definition/49/deskriptive_statistik/.

Statista (Statista). o. J. „Beliebteste Programmiersprachen weltweit 2025". Zugegriffen 27. Dezember 2025. https://de.statista.com/statistik/daten/studie/678732/umfrage/beliebteste-programmiersprachen-weltweit-laut-pypl-index/.

„Teilnehmende Beobachtung". o. J. Zugegriffen 2. April 2026. https://www.ph-freiburg.de/quasus/was-muss-ich-wissen/daten-erheben/beobachtungsverfahren/teilnehmende-beobachtung.html.

Till Sobioch. 2024. „Untersuchung der SOLID-Prinzipien: Nutzen und Grenzen in der Spieleentwicklung". Bachelorarbeit. Technische Hochschule Ostwestfalen-Lippe,

Februar 16. https://www.th-owl.de/elsa/download/11119/11146/Till_Sobioch_Bachelorarbeit.pdf.

„Was versteht man unter Top-Down oder Bottom-Up Verfahren?“ o. J. Zugegriffen 13. August 2025. https://www.softguide.de/funktion/top-down-oder-bottom-up-verfahren.

„Werwölfe von Düsterwald | Asmodee Deutschland“. o. J. Zugegriffen 28. Dezember 2025. https://www.asmodee.de/produkte/werwoelfe-von-duesterwald.

„Where Did the Idea for the Magic 8 Ball Come From? | Britannica“. o. J. Zugegriffen 5. September 2025. https://www.britannica.com/story/where-did-the-idea-for-the-magic-8-ball-come-from.

10 Anhang

Um die strukturelle Integrität des Anhangs zu bewahren, sind alle Anhänge als .zip-Datei über folgendem Link zur Verfügung gestellt:

https://ucloud.univie.ac.at/index.php/s/i9LWNyoNAQatac6

11 Verwendung von KI

Im Zuge dieser Arbeit wurde die KI „Gemini“ (Gemini, o. J.) als Unterstützung zur Korrekturlesung herangezogen. Dafür wurden einzelne Textabschnitte auf Grammatik und Rechtschreibung sowie wissenschaftliche Formulierungen überprüft. Das Feedback der KI wurde zum Teil übernommen. Da das Schreiben und Korrigieren von über 60 Seiten Fließtext einen erheblichen Aufwand darstellen, konnte so die benötigte Arbeitszeit deutlich reduziert werden – das Korrekturlesen durch eine Person hätte den Abschluss um einige Wochen verzögert.

Auch die diversen Infotexte aus der Kategorie „Grundlagen der Programmierung“ wurden von einer KI erstellt und im Anschluss von mir überprüft und entsprechend angepasst.

Weiters wurden die Websites für die Unterrichtseinheit „Tastaturschreiben“ sowie „Variablen“ des finalen „Methodenkoffers“ mittels KI generiert. So konnte eine optisch ansprechende Oberfläche auch ohne tiefergehende Kenntnisse des Webdesigns gestaltet werden, welche auf die Lernenden motivationsfördernd wirken könnte.