Bentz, Anette

Interaktive und spielerische Einführung in algorithmische Denkweisen

Grey, Jan [Hrsg.]; Schmitz, Denise [Hrsg.]; Gryl, Inga [Hrsg.]; Best, Alexander [Hrsg.]; Kuckuck, Miriam [Hrsg.]; Humbert, Ludger [Hrsg.]: Informatische Bildung in der Grundschule. Befunde, Diskussionen, Erfahrungen. Bad Heilbrunn : Verlag Julius Klinkhardt 2025, S. 175-192

Anette Bentz

Interaktive und spielerische Einführung in algorithmische Denkweisen

Abstract

Eine effektive Einführung in die Programmierung erfordert einen wissenschaftlich fundierten, altersgerechten und zielgerichteten Unterricht, der das Interesse der Schüler:innen weckt. Wir schlagen daher eine Unterrichtseinheit vor, die sich auf eine schrittweise Einführung in das algorithmische Denken sowie auf eine aufeinander aufbauende Sequenz von Lernstufen stützt. Das Hauptziel dieser Lehrsequenz ist eine umfassende Einführung in die Grundlagen und Anwendungen der Blockprogrammierung. Hierbei werden zunächst Methoden ohne Nutzung von elektronischen Geräten *(unplugged)* und danach Methoden mittels elektronischer Geräte verwendet. Jede Lernphase dieser Einheit hat das Ziel, eine solide Basis für einen reibungslosen Übergang von der unplugged Arbeit zur Online-Blockprogrammierung zu schaffen. Die Lernmaterialien sind konsequent auf das Ziel ausgerichtet und zeichnen sich durch eine kohärente Struktur aus und die Komplexität der Aufgaben steigt schrittweise an. Dieser Ansatz ermöglicht die Anwendung fundamentaler Konzepte wie Sequenzen und Schleifen. Gleichzeitig bietet er Potenzial für die Einführung weiterführender Konzepte wie Bedingungen, Funktionen und Variablen. Die Lehrsequenz ist in vier Stufen gegliedert: Basteln, Bewegungsspiele, Brettspiele und Roboterprogrammierung. In der Anfangsphase erstellen die Schüler:innen eine programmierbare Umgebung, die der realen Umgebung ähnelt. Durch physische Aktivitäten entdecken sie diese Umgebung auf spielerische Art und Weise. Dabei wird das Verständnis für Programmierung spielerisch vertieft. Anschließend integrieren die Lernenden Programmierblöcke in Brettspiele, wobei sie die Bewegungen von Spielfiguren planen und ausführen. Die letzte Lektion konzentriert sich auf die Programmierung von Robotern und bietet somit eine solide Grundlage, um zukünftigen Aufgaben im Bereich der Blockprogrammierung kompetent zu begegnen.

1 Einleitung

Die Integration von Informatik in den Lehrplan für Grundschüler:innen ist von entscheidender Bedeutung, da sie nicht nur grundlegende Kenntnisse in der digitalen Welt vermittelt, sondern auch ihre kognitiven Fähigkeiten, Problemlösungsfähigkeiten und kreativen Denkprozesse fördert (Wing 2006; Resnick 2017). Für eine optimale Unterrichtsgestaltung im Grundschulalter

doi.org/10.35468/6203-13

ist es essenziell, den angemessenen Abstraktionsgrad und geeignete Unterrichtsmethoden für diese Zielgruppe zu identifizieren und die Lerneinheiten entsprechend den Bedürfnissen junger Lernender anzupassen.

Der Informatikunterricht hat seit seiner Entstehung eine kontinuierliche Entwicklung und Anpassung an den technologischen Fortschritt sowie an die Bedürfnisse der Schüler:innen erfahren (Schubert 2011). Beispielweise wurde die Informatik ursprünglich vorwiegend im universitären Kontext unter Verwendung textbasierter Sprachen gelehrt. Um jedoch den Unterricht zielgruppengerecht gestalten zu können, insbesondere in der Sekundarstufe 1, mussten Konzepte und Inhalte angepasst werden. Hierbei wurden unter anderem die Blockprogrammierung (Scratch 2023) sowie *unplugged* Methoden (Bell & Vahrenhold 2018) integriert. Da Informatik zunehmend ein integraler Bestandteil der Grundschulausbildung wird, ist es erforderlich, die vorhandenen Lernmethoden an die jüngeren Schüler:innen anzupassen. Im Hinblick auf diese Veränderungen ist anzumerken, dass die grundlegenden informatischen Lerninhalte seit vielen Jahren nahezu unverändert geblieben sind, obwohl die Lehrmethoden einer stetigen Veränderung unterworfen sind. Aus diesem Grund wird in diesem Beitrag eine Unterrichtseinheit präsentiert, die speziell auf diese jüngere Zielgruppe abzielt und konstante Informatikkonzepte fokussiert.

Programmierung ist ein zentraler Bestandteil der Informatikausbildung. Der Prozess des Programmierens schult vielfältige Kompetenzen, wie das Verständnis von digitalen Prozessen. Außerdem werden durch die Programmierung auch Denkprozesse gefördert, indem komplexe Probleme in kleinere, besser handhabbare Teile zerlegt und schrittweise gelöst werden. Dieser Ansatz lässt sich unter dem Begriff *Computational Thinking* (CT) zusammenfassen (Wing 2006). In der Fachliteratur wird eine positive Korrelation zwischen Programmieren und Problemlösungsfähigkeiten beschrieben (Brackmann u. a. 2017; Moreno-Leon u. a. 2015; Gomes 2017). Folglich lässt sich ableiten, dass die Fertigkeiten zur Lösung von Problemen, die beim Programmieren erworben werden, nicht nur in diesem spezifischen Bereich von Nutzen sind, sondern auch bei der Bewältigung von Herausforderungen in anderen Lebensbereichen, insbesondere wenn Probleme in überschaubare Schritte unterteilt und schrittweise angegangen werden, hilfreich sind.

Außerdem kann das Programmieren die Teamarbeit und Zusammenarbeit fördern, da Projekte in diesem Bereich oft eine gemeinschaftliche Zusammenarbeit erfordern (Resnick 2017). Das kann dazu beitragen, dass Lernende soziale Fähigkeiten wie effektive Kommunikation, Zusammenarbeit und Konfliktlösung entwickeln. Des Weiteren eröffnet die Programmierung den Schüler:innen die Möglichkeit, ihre kreative Seite zu entfalten, indem sie Technologie als Ausdrucksmittel nutzen und ihre Ideen mithilfe von Programmierung zum Leben erwecken (Resnick 2017).

doi.org/10.35468/6203-13

Der Zugang zur Programmierung für Grundschüler:innen ist vielschichtig. Einige Tools ermöglichen einen haptischen Einstieg, indem beispielsweise Roboter durch die Verwendung von Knöpfen oder Puzzleteilen programmiert werden können (B-Bot 2023; Kubo 2023). Hier werden algorithmische Denkweisen trainiert, die allerdings keinen Transfer zur Blockprogrammierung im Onlinemodus beinhalten. Im Gegensatz dazu bieten Online-Plattformen wie *Scratch* (Scratch 2023) eine Programmierumgebung, in der die Blockprogrammierung im Zentrum steht. Diese Plattformen bieten nicht nur eine Fülle von weiterführenden Lerninhalten, sondern auch Animationen, Farben und Töne (Scratch 2023; Code.org 2023; Open Roberta 2023; Alice 2023). Diese Vielfalt könnte möglicherweise Einsteiger und unerfahrene Lehrkräfte von dem Lernziel, nämlich der Entwicklung des algorithmischen Denkens, ablenken.
Die präsentierte Unterrichtseinheit geht auf diese Aspekte ein, indem sie haptische Spiele mit einem Transfer zur Blockprogrammierung kombiniert und gleichzeitig eine reduzierte Programmierumgebung anbietet. Verschiedene Lehrmethoden werden dabei integriert und ein Schwerpunkt auf algorithmische Grundkonzepte gelegt. Im Vergleich zu ähnlichen *unplugged* Materialien, die oft auf spezifische Anwendungsfälle oder Spiele ausgerichtet sind, zeichnen sich die vorliegenden Materialien durch ihre konzeptionelle Ausrichtung auf flexible Anwendungen aus und bieten dadurch vielseitigere Einsatzmöglichkeiten. Dies erlaubt eine effektive Anpassung an das Alter, Vorwissen und die Präferenzen der Zielgruppe. In der Literatur wird beschrieben, dass *unplugged* Methoden eine besondere Effektivität zeigen, wenn sie mit digitalen Methoden kombiniert werden, da sie den Lernenden ermöglichen, die Verbindung zwischen *unplugged* und *plugged* Einheiten herzustellen (Bell & Vahrenhold 2018). Dementsprechend endet diese Einheit mit einer digitalen Anwendung. Im Folgenden werden relevante Studien zusammengefasst und Forschungsfragen formuliert. Anschließend wird die Intervention detailliert beschrieben und eine abschließende Betrachtung der Thematik vorgenommen.

2 Relevante Studien

Die hier vorgestellte Unterrichtseinheit baut auf etablierten Konzepten auf, integriert und erweitert diese, um eine kohärente und zielgerichtete Unterrichtssequenz zur Förderung des *Computational Thinking* zu gestalten. Im Unterrichtsdesign wird besondere Sorgfalt darauf verwendet, nicht nur die Materialien an das Alter der Schüler:innen anzupassen, sondern auch ihre individuellen Interessen zu berücksichtigen. Daher werden im Folgenden verschiedene Interessenprofile vorgestellt, die sowohl im Kontext von Gender als auch über Geschlechtsaspekte hinausgehen. Der inhaltliche Schwerpunkt liegt auf Schleifen, da sie ein fundamentales Konzept in der Informatik dar-

doi.org/10.35468/6203-13

stellen. Obwohl sie oft in den ersten Unterrichtsstunden behandelt werden, können sie in verschiedenen Altersgruppen zu Verständnisschwierigkeiten führen. Jeder dieser Aspekte (*Computational Thinking*, Programmierung, Berücksichtigung von Interessen, Fokussierung auf Schleifen) wird im Folgenden kurz erläutert und mit einschlägiger Literatur verknüpft.

Das Erlernen der Programmierung bringt zahlreiche Vorteile mit sich. Einerseits trägt es zur Teilhabe an einer sich immer stärker digitalisierenden Welt bei und fördert das Verständnis für die technologischen Aspekte, die unsere Umgebung prägen. Zusätzlich ist es mit der Entwicklung von Kompetenzen im Bereich des *Computational Thinking* (CT) verknüpft (Wing 2006; Kanaki & Kalogiannakis 2023; Cernochová u. a. 2020). *Computational Thinking* bezeichnet die Fähigkeit, komplexe Probleme mit Hilfe von Prinzipien und Methoden der Informatik durch Analyse, Strukturierung und Lösung zu bewältigen. Dies umfasst die Fähigkeiten zur Dekomposition, Mustererkennung, Abstraktion und algorithmischem Denken. Das Ziel besteht darin, eine allgemeine Denkweise zu entwickeln, um Probleme unabhängig von einer bestimmten Programmiersprache oder einem konkreten Algorithmus zu analysieren und Lösungen zu entwerfen. Daher können diese Fähigkeiten auch bei der Untersuchung komplexer Probleme in anderen Lebensbereichen von Nutzen sein (Wing 2006).

Verschiedene Initiativen setzen sich für die Entwicklung und Erforschung von Unterrichtsmethoden in der Grundschule ein, wie beispielsweise (Bildungsportal NRW 2023; Barradas u. a. 2021; Cernochová u. a. 2020). In diesem Kontext findet der erste Kontakt mit Informatik physisch statt und ist in der Regel computerfrei. Diese Unterrichtsform wird auch als *unplugged* bezeichnet und ist besonders geprägt durch die Arbeit der *CS Unplugged* Initiative (CS Unplugged 2023). *Unplugged* Programmierung bezieht sich auf eine Methode, bei der grundlegende Konzepte des Programmierens vermittelt werden, ohne dass dabei ein Computer oder digitale Technologie zum Einsatz kommt. Stattdessen werden Aktivitäten und Spiele genutzt, um abstrakte Konzepte und Denkweisen der Programmierung auf physische und interaktive Weise zu veranschaulichen. Dabei zeigt sich, dass Teilnehmer:innen beim Erleben von *unplugged*-Konzepten komplexere Programme erstellen (Bell & Vahrenhold 2018; Munasinghe u. a. 2023). Daher konzentriert sich die Lerneinheit insbesondere auf *unplugged*-Einheiten. Durch die Einführung der Blockprogrammierung wurde der Zugang zur Programmierung erleichtert. Diese vereinfacht das textbasierte Programmieren durch die Organisation von Blöcken per *Drag-and-Drop*. Somit werden Syntaxprobleme minimiert. Ein bekanntes Beispiel dieser Art der Programmierung ist die Plattform *Scratch* (Scratch 2023). Seit 2007 wird sie weitreichend in schulischer und außerschulischer Bildung eingesetzt. Verschiedene Forschungsprojekte beschäftigen sich mit der Programmierung in *Scratch*. Dazu gehören die verschiedenen Herangehensweisen von Mädchen

doi.org/10.35468/6203-13

und Jungen (Simon & Geldreich 2017), die Qualität des Codes (Scratch 2023) und mögliche Missverständnisse, die während des Programmierens auftreten können (Vanicek u. a. 2022). Zusätzlich ist der Einsatz der Blockprogrammierung in der Bildung weitreichend und es gibt viele Projekte und Anwendungen, die diese Technik nutzen. Dazu gehören beispielsweise die Erstellung von Geschichten, das Programmieren von 3D-Welten sowie von Robotern (Steamon.eu 2022; Sphero Edu 2021; Alice 2023; Open Roberta 2023).

Die Bedeutung von Präferenzen und Interessen für die Lernmotivation und den Lernerfolg wird in verschiedenen Kontexten betont (Bentz & Standl 2022; Michaelis & Weintrop 2022). Insbesondere das Interesse an einem Thema wird als entscheidender Faktor für den langfristigen Lernerfolg und die Motivation angesehen (Michaelis & Weintrop 2022). Das Ziel der präsentierten Unterrichtseinheit besteht darin, ein breites Spektrum von Lernenden für das Fach Informatik zu begeistern. Dies soll durch die Berücksichtigung von geschlechterspezifischen Präferenzen, die in der Literatur beschrieben werden, sowie anderen Präferenzen, die über das Geschlecht hinausgehen, im Kontext des faktischen und sozialen Lernens integriert werden. In der Literatur wird diskutiert, ob es Unterschiede in Bezug auf spezifische Interessen und Lernpräferenzen zwischen Mädchen und Jungen im Bereich der Informatik gibt (Buhnova & Happe 2020; Grabarczyk u. a. 2022; Marquardt u. a. 2023; Happe & Marquardt 2023; Bentz & Standl, 2023a). In diesen Studien wird berichtet, dass Mädchen häufiger soziale und interaktive Lernmethoden bevorzugen, indem sie beispielsweise Geschichten programmieren, während Jungen eine stärkere Präferenz für die Entwicklung von Spielen oder die Arbeit mit Objekten zeigen. Unabhängig vom Geschlecht zeigen sich diese Präferenzen auch im Hinblick auf die Vorlieben für Geschichten und soziale Interaktion. Diese Präferenzen werden als dramaturgische Präferenz (Dramatisten) sowie als Vorliebe für die Funktionsweise von Objekten und räumliche Anordnungen bezeichnet, die objektbezogene Präferenz (Patterner). Es wird betont, dass sie einen signifikanten Einfluss auf die Lernmotivation und die Herangehensweisen haben (Bentz & Standl 2022; Grabarczyk u. a. 2022; Gulz 2002; Resnick 2017; Bentz & Standl 2023a).

Um eine breite Gruppe von Lernenden für die Informatik zu motivieren, wird vorgeschlagen, in Lerneinheiten die Präferenzen bezüglich des Geschlechts sowie der Lernstile zu berücksichtigen (Gulz 2003; Rusk u. a. 2008; Grabarczyk u. a. 2022). Dies kann durch die Integration von Aufgaben erreicht werden, die sowohl einen sozialen als auch einen interaktiven Charakter haben und dabei einen klar strukturierten und objektbezogenen Ansatz verfolgen (Bentz & Standl 2022).

Die grundlegenden informatischen Konzepte wie Sequenzen, Schleifen, Bedingungen und Funktionen sind in der Informatikausbildung stabil und neh-

men einen wesentlichen Teil des Lehrmaterials ein. Besonders wichtig ist das Verständnis von Schleifen, da sie nicht nur eine unerlässliche Komponente des Programmierens darstellen, sondern auch in den frühen Unterrichtseinheiten behandelt werden. Jedoch zeigen Untersuchungen, dass Schüler:innen in verschiedenen Altersstufen – von der Sekundarstufe bis zur Universität Schwierigkeiten haben können, das Konzept der Schleifen zu verstehen oder anzuwenden (Mladenovic u. a. 2016; Grover & Basu 2017; Vanicek u. a. 2022; Bentz & Standl 2023b). Die vorliegende Lerneinheit strebt die vollständige, schrittweise und vielseitige Vermittlung von Schleifen in der Blockprogrammierung an, um eine solide Grundlage für fortgeschrittene Programmierkenntnisse zu schaffen.

3 Forschungsfragen

Vor dem Hintergrund der obigen Ausführungen ergeben sich die folgenden übergeordneten Forschungsfragen zum Thema Effektivität von Interventionen zur Blockprogrammierung:

- Forschungsfrage 1: Steigert die Intervention das Verständnis von Sequenzen und Schleifen in der Blockprogrammierung?
- Forschungsfrage 2: Steigert die Intervention das Interesse an der Blockprogrammierung in Schüler:innen oder Lehrkräften?

4 Intervention: Vierstufige Einführung in die Blockprogrammierung

Bei der Ausgestaltung der individuellen Lernschritte wurde besonderes Augenmerk auf das Lernziel, die Orientierung an den Bedürfnissen der Zielgruppe sowie eine leicht verständliche Handhabung der Materialien gelegt.

Lernzielorientierung: Das Lernziel der Lerneinheit besteht darin, den Schüler:innen das Verständnis und die kreative Anwendung von Sequenzen und Schleifen mithilfe von Blockprogrammierung zu vermitteln. Jede Lernstufe in dieser Lerneinheit zielt darauf ab, das Ziel zu erreichen, indem sie eine vereinfachte Darstellung der Blockprogrammierung nutzt. Auf diese Weise erlernen die Schüler:innen schrittweise und kontinuierlich eine konsistente Repräsentation der Programmierung.

Zielgruppenorientierung: In den Lernstufen werden die Lernenden mittels einer vielseitigen Methodik auf verschiedenen Ebenen angesprochen. Zudem legt das Unterrichtsdesign einen besonderen Fokus auf Individualisierbarkeit, Einbeziehung von Präferenztypen und Geschlechtsneutralität.

doi.org/10.35468/6203-13

Fokus auf *User Experience*: Zudem wurde die *unplugged* Lerneinheit so konzipiert, dass sie auch von Lehrkräften mit begrenzter Programmiererfahrung problemlos umgesetzt werden kann. Zum einen bedarf es keiner computertechnischen Unterstützung durch die Lehrkraft, zum anderen sind die Materialien handlich und vielseitig einsetzbar.

Die Intervention integriert verschiedene didaktische Methoden, darunter Bastelarbeiten, Bewegungsspiele, Brettspiele und Roboterprogrammierung, um die Lernenden schrittweise mit der Blockprogrammierung vertraut zu machen, wie in Abbildung 1 zusammengefasst und in Tabelle 1 detailliert. Im Rahmen der Bastelarbeit und der Bewegungsspiele erhalten die Lernenden Möglichkeiten, sich mit der Programmierumgebung vertraut zu machen und ihre anfänglichen Programmierkenntnisse in bewegungsorientierten Spielen zu erproben. Auf dieser Basis werden die erstellten Programmierbausteine auf Brettspiele übertragen, um den Schülern und Schülerinnen die Planung und Umsetzung der Bewegungen von Spielfiguren strategisch zu vermitteln. Im letzten Schritt werden die Fertigkeiten vertieft, indem optisch ähnliche Programmierblöcke in Kombination mit realen Robotersystemen eingesetzt werden. Die Besonderheit dieser Methodik liegt in ihrer kohärenten Struktur, die schrittweise die Integration neuer Funktionalitäten ermöglicht. Im Folgenden werden die spezifischen Lernstufen detailliert erläutert.

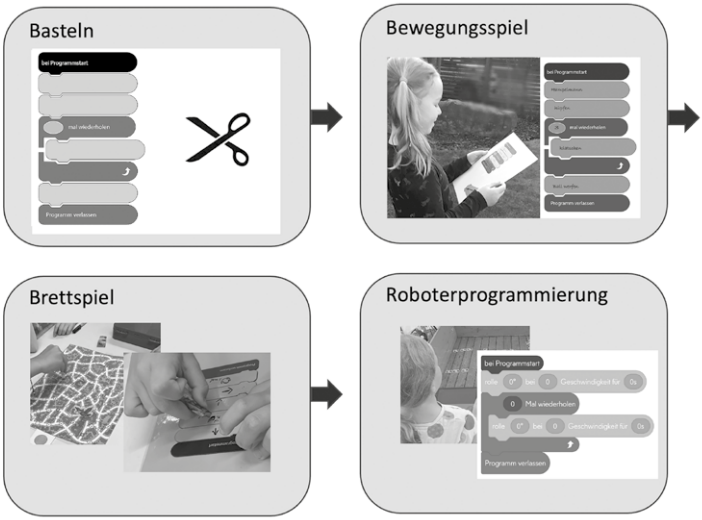

Abb. 1: Übersicht über die einzelnen Lernstufen: Basteln, Bewegungsspiel, Brettspiel, Programmierung (eigene Darstellung)

doi.org/10.35468/6203-13

Tab. 1: Lernziele, zeitlicher Ablauf und weiterführende Links (eigene Darstellung)

Lernstufe	Lernziel	Dauer (Min.)	Weiterführende Links
Einführung und Basteln	Kenntnisse und Vorstellungen sowie mögliche Zukunftsperspektiven bezüglich Roboter artikulieren. Kennenlernen von Sequenzen und Schleifen	45	Einführung: Der Roboter Check (Checker Welt, 2023): Online unter: https://youtu.be/DANJgT73XVU?si=xE8TZ33eY-y-w40n (Abrufdatum: 03.04.2024). Kopiervorlage: Online unter: https://cloud.ph-karlsruhe.de/index.php/s/RLgMn5fMikg3zWT (Abrufdatum: 03.04.2024).
Bewegungsspiele	Die Schüler sollen in der Lage sein, einfache Sequenzen zu erstellen und grundlegende Schleifenkonzepte zu verstehen und anzuwenden.	45	Erweiterte Einsatzmöglichkeit: Online unter: https://cloud.ph-karlsruhe.de/index.php/s/nqbswJCPqaWJQiX (Abrufdatum: 03.04.2024)
Brettspiele	Vertiefung des Lernzieles aus „Bewegungsspiel“	45	Zusammenfassendes Video: Online unter: https://youtu.be/JyyRLY6zMmM (Abrufdatum: 03.04.2024)
Roboterprogrammierung	Die Schüler sollen in der Lage sein, einfache Sequenzen und Schleifen mithilfe spielerischer Roboterübungen zu verstehen und anzuwenden.	90	Beispielaufgaben: Online unter: https://cloud.ph-karlsruhe.de/index.php/s/nCtx8R2DSfBw2A2 (Abrufdatum: 03.04.2024).

4.1 Einführung und Bastelarbeit

Zu Beginn der Unterrichtseinheit werden die Schülerinnen und Schüler in die Welt der Robotik eingeführt. Zunächst wird gemeinsam in der Gruppe diskutiert, welche Roboter den Lernenden bereits bekannt sind oder wo sie sich Unterstützung durch Roboter vorstellen können. Im Anschluss wird eine haptische Programmierumgebung erstellt. Ein Auszug ist in Abbildung 2 dargestellt, in Tabelle 1 befindet sich ein Link zur Kopiervorlage. Die visuelle Programmierumgebung orientiert sich an der später verwendeten Program-

doi.org/10.35468/6203-13

mierumgebung und umfasst Blöcke für Programmstart und -ende, Sequenzen, Schleifen und Bedingungen. Die Blöcke enthalten teilweise vorgegebene Texte und können teilweise beschriftet werden.

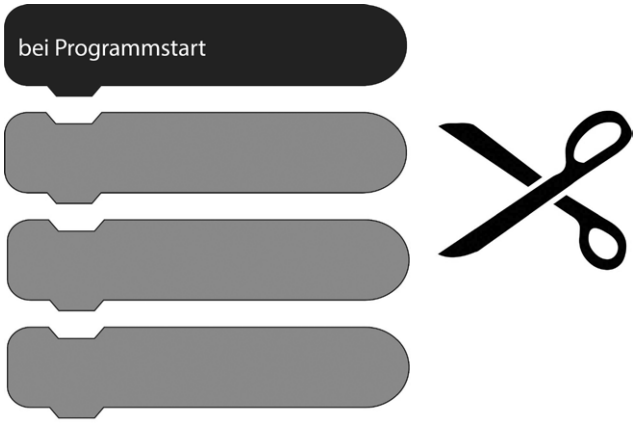

Abb. 2: Bastelarbeit: die Programmierumgebung wird von den Lernenden erstellt (eigene Darstellung)

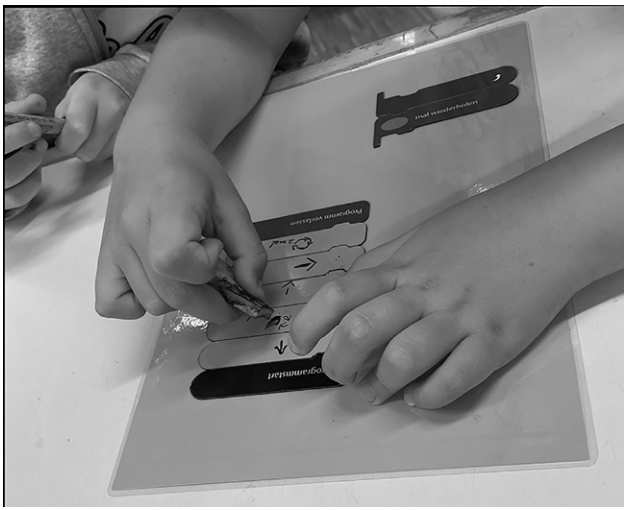

Abb. 3: Die Programmierblöcke werden beschriftet und angeordnet (eigene Darstellung)

doi.org/10.35468/6203-13

4.2 Bewegungsspiele

Im nächsten Schritt wird zur aktiven Förderung der Programmierfähigkeiten die eigens entwickelte Programmierumgebung für Bewegungsspiele eingesetzt. Die Schülerinnen und Schüler gestalten in kooperativer Zusammenarbeit Bewegungsabläufe. Vorab wird eine Einführung in die Konzeption eines Programms gegeben, das mit einem Start- und Endblock beginnt und endet. Im Rahmen dieser Aufgabe nutzen die Schülerinnen und Schüler Programmierblöcke, um Aktionen wie „hüpfen" oder „klatschen" zu definieren (siehe Abb. 4). Dabei wird ihnen erläutert, dass jede Aktion nur einmalig ausgeführt wird. Anschließend erfolgt die Einführung des Schleifenblocks, der verwendet wird, um eine Aktion mehrfach ausführen zu lassen. Die Schüler:innen haben in Gruppen die Möglichkeit, eigenständig Programme zu konzipieren. Zur Erhöhung des Unterhaltungsfaktors ist es erlaubt, eine Vielzahl von Bewegungen einzubeziehen, wie beispielsweise Hampelmann, Liegestütze oder Tanzschritte. Nach einer festgelegten Programmierzeit präsentieren die Teams ihre choreografierten Bewegungen vor dem Lehrenden oder der Klasse. Ein Teammitglied liest das erstellte Programm vor, während ein anderes Teammitglied die entsprechende Bewegung ausführt. In einer darauffolgenden, progressiven Lehrsequenz wäre es möglich, schrittweise Bedingungen und Funktionen in ähnlicher Weise zu integrieren.

Abb. 4: Bewegungen werden geplant, programmiert und ausgeführt (eigene Darstellung)

doi.org/10.35468/6203-13

Abbildung 5 zeigt beispielhaft Anordnungen der Programmierbausteine mit zunehmender Komplexität. Ein Beispiel für Bewegungsspiele und die Vermittlung von komplexen Konstrukten ist in Tabelle 1 verlinkt.

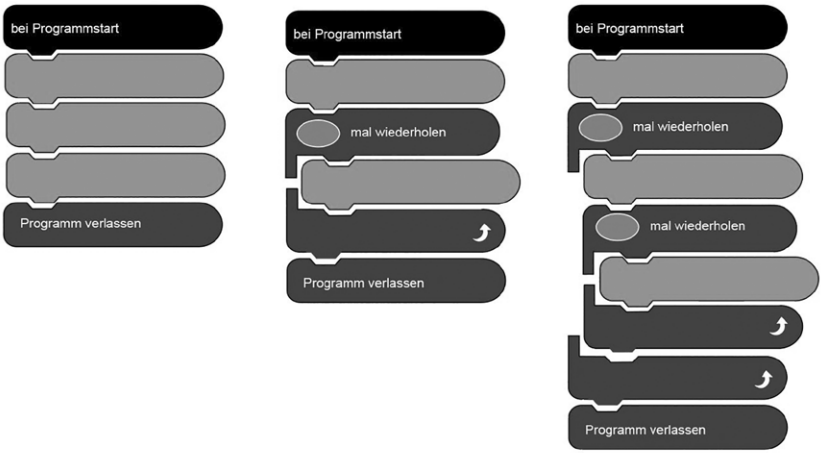

Abb. 5: Programmierung von Sequenzen, einfache und geschachtelte Schleife (eigene Darstellung)

4.3 Brettspiele

Um das Verständnis von Programmierkonstrukten weiter zu vertiefen, wird im nächsten Schritt ein Brettspiel durchgeführt. Ein solches Brettspiel und die benötigten Programmierblöcke sind in Abbildung 6 beispielhaft dargestellt. Als Rahmengeschichte dient eine einfache Schatzsuche, welche im Folgenden ausführlich erläutert wird, um einen Einblick in die Spielidee zu geben.

Zusammenfassung:

Auf dem Spielfeld werden Schätze in Form von kleinen Steinen oder Figuren platziert. Das Ziel des Spiels ist es, möglichst viele Schätze zu sammeln. Gespielt wird in mehreren Runden, in denen die Bewegungen der Spielfiguren im Voraus geplant und in der Programmierumgebung festgelegt werden. Die Anzahl der Programmierblöcke, die in jeder Runde verwendet werden dürfen, wird entweder von der Lehrkraft vorgegeben oder durch Würfeln bestimmt.

Vorbereitung:

1. Brett und Figuren auswählen, Würfel bereitlegen.
2. Schätze verteilen.

doi.org/10.35468/6203-13

3. Programmierblöcke verteilen (jede:r bekommt Programm, Start/Ende und eine Schleife).
4. Alle Sequenzen in die Mitte legen.

Abb. 6: Brettspiel Wadenbeisser (Moses Verlag 2023) (eigene Darstellung)

Ablauf:

1. Würfeln, um die Anzahl der Sequenzen in der aktuellen Runde zu bestimmen.
2. Anzahl der Sequenzen aus der Mitte nehmen.
3. Alle Teams programmieren ihren Bewegungsablauf (vgl. Abb. 3).
4. Wenn alle Bewegungsabläufe fertig sind, werden die Figuren nacheinander auf dem Spielfeld bewegt.
5. Wenn eine Figur auf dem Weg auf das Feld mit einem Schatz kommt, dürfen die Spieler den Schatz nehmen.
6. Das Spiel endet, wenn alle Schätze vom Spielfeld genommen wurden.

Es können weitere Spielideen gemäß den Präferenzen der Teilnehmenden in Betracht gezogen werden. Eine Möglichkeit wäre, Spielende ähnlich wie bei *Mensch-Ärgere-Dich-Nicht* aus dem Spiel zu entfernen oder kooperativ gegen die Zeit oder einen imaginären Gegner spielen zu lassen. Dabei ist zu betonen, dass zahlreiche Spiele geeignet sind, um Programmierkonzepte zu vermitteln, bei denen die Spielaktion im Voraus geplant und anschließend ausgeführt

doi.org/10.35468/6203-13

wird. Die Auswahl altersgerechter Spiele, welche den Vorlieben der Teilnehmenden entsprechen, kann eine motivierende Wirkung haben. Bei der Auswahl eines Spiels empfiehlt es sich in der Regel, vorhandene Spiele daraufhin zu prüfen, ob sie für die Integration von Programmieraspekten geeignet sind. Besonders vorteilhaft ist die Gestaltung eines Spielfelds, das es ermöglicht, dass die Spielfiguren in bestimmte Richtungen (nach vorne, hinten, rechts, links) bewegt werden können. Beispiele hierfür sind Spiele wie *Wadenbeiser* (Moses Verlag 2023) oder *Robo-Ralley* (Garfield 2023).

4.4 Roboter Programmierung

Für die Einführung in die Roboterprogrammierung werden Schüler:innen zunächst im Klassenverbund mit der Entwicklungsplattform des jeweiligen Roboters sowie der zugehörigen Blockprogrammierung vertraut gemacht. Dieser Prozess erfolgt durch eine praxisorientierte Präsentation, wobei die Betonung auf der Analogie zur selbst erstellten Umgebung liegt, und gleichzeitig relevante Unterschiede erläutert werden. In den vorliegenden Beispielen wurde der Roboter *Sphero* verwendet, wie von *Sphero Edu* (2021) bestätigt. Eine Gegenüberstellung der analogen und digitalen Programmierumgebungen ist in Abbildung 7 zu finden. Im Anschluss daran haben die Schüler:innen die Möglichkeit, sich in Gruppenarbeit mit dem Robotersystem vertraut zu machen. Im Lernraum wurden verschiedene Aufgaben vorbereitet, um die Schüler:innen in die Lage zu versetzen, ihre erworbenen Fähigkeiten in der Praxis anzuwenden. Die Roboter können beispielsweise so programmiert werden, dass sie vorgegebene Wege befahren, wie in Abbildung 8 gezeigt. In Tabelle 1 ist eine detaillierte Übersicht über beispielhafte Aufgaben verlinkt.

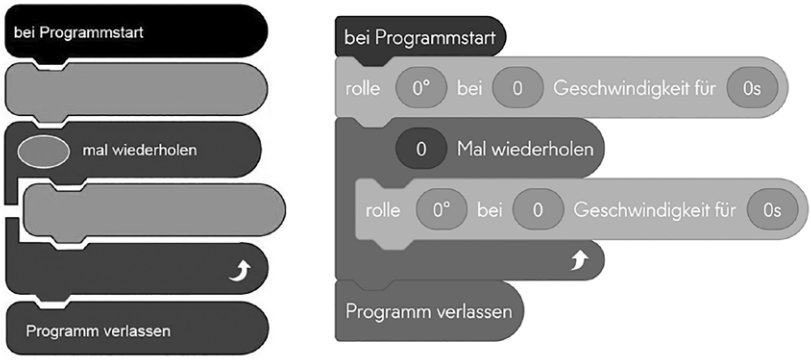

Abb. 7: Unplugged Programmierung im Vergleich zur Programmierumgebung von Sphero Edu (eigene Darstellung)

doi.org/10.35468/6203-13

Abb. 8: Beispiel für die Umsetzung einer Programmieraufgabe (eigene Darstellung)

5 Zusammenfassung und Fazit

Zur effektiven Einführung in die Programmierung schlagen wir eine altersgerechte Unterrichtseinheit vor. Diese beinhaltet eine aufbauende Sequenz von Lernstufen, mit dem übergeordneten Ziel, Grundschüler:innen systematisch in die Blockprogrammierung einzuführen. Die methodische Struktur umfasst die Stufen Basteln, Bewegungsspiele, Brettspiele und Roboterprogrammierung. Die Schüler:innen erstellen zunächst eine visuell der realen Programmierumgebung ähnelnde Umgebung, erleben sie durch physische Aktivitäten spielerisch und integrieren dann selbst erstellte Programmierblöcke in Brettspiele. Die abschließende Stufe konzentriert sich auf die eigenständige Programmierung von Robotern. Dieser praxisnahe Ansatz fordert das vertiefte Verständnis für die Roboterprogrammierung und ermöglicht den Schüler:innen eine aktive Auseinandersetzung mit den technologischen Herausforderungen.
Die Blockprogrammierung bildet einen essenziellen Bestandteil zahlreicher hervorragend ausgearbeiteten Programme und Projekte (bspw. Scratch 2023; Alice 2023; code.org 2023; Open Roberta 2023; DelighteX GmbH 2023). Jedoch ist das Verständnis der Blockprogrammierung von entscheidender Bedeutung, um Zugang zu diesen Ressourcen zu erhalten. Im Spannungsfeld der rasanten digitalen Entwicklung, des meist geringen Zeitbudgets der Lehrkräfte, die häufig keine umfassende Informatikausbildung genossen haben, kann jedoch die Integration von Blockprogrammierungssoftware herausfordernd sein. Lehrkräfte müssen sich möglicherweise zunächst mit neuen Softwarelösungen vertraut machen, während die Grundschüler:innen durch die Funktionsweise von Endgeräten und Programmen abgelenkt werden könn-

doi.org/10.35468/6203-13

ten. Vor diesem Hintergrund empfehlen wir die vorgestellte Methodik. Diese konzentriert sich simultan auf grundlegende Lernziele (z. B. Blockprogrammierung mit Sequenzen, Schleifen, Bedingungen), lässt sich an die Zielgruppe anpassen und kann im *unplugged* Teil ohne digitale Kompetenzen angewandt werden. Falls Ressourcen fehlen, um den digitalen Aspekt umzusetzen, erlangen die Lernenden dennoch einen substanziellen Nutzen durch eine fundierte Einführung in die Blockprogrammierung. Somit stellt diese Einführung eine solide Basis dar, um künftigen Aufgaben im Bereich der *plugged* Blockprogrammierung kompetent zu begegnen.

6 Erfahrungen und Ausblick

Die Materialien wurden an mehreren Grundschulen getestet. Die gewonnenen Erfahrungen zeigen, dass der Ablauf für die Einführung in die Programmierung in der Grundschule geeignet ist. Die Lehrkräfte äußerten positive Bewertungen zu den Materialien und bestätigten, dass sie die *unplugged* Teile auch ohne tiefgehende Informatikkenntnisse eigenständig durchführen können. Nach einer kurzen Einführung waren die Schüler:innen in der Lage, eigenständig Bewegungsabläufe zu programmieren, indem sie Sequenzblöcke beschrifteten und in eine definierte Reihenfolge brachten. Die anschließende Einheit des Bewegungsspiels motivierte die Lernenden dazu, das Brettspiel zu spielen, da sie erneut auf die gleichen Programmiermaterialien zurückgreifen konnten und somit keine weitere Erklärung erforderlich war. Der Übergang zur digitalen Programmierumgebung hat den Schülerinnen und Schülern keinerlei Schwierigkeiten bereitet. Sie haben schnell gelernt, wie sie die Roboter programmieren können. Aufgrund dieser positiven Erfahrung ergeben sich vielfältige Forschungsfragen. Ein besonderes Interesse besteht darin, die Auswirkungen der Materialien auf die Kompetenzen und Motivation von Lehrkräften und Lernenden zu untersuchen. Eine konkrete Fragestellung ist etwa, inwiefern die Intervention das Verständnis von Schleifen verbessert hat. Nach einer Einführung in die Blockprogrammierung können Produkte wie *Scratch* (2023) oder *Alice* (2023) auch für fächerübergreifenden Unterricht genutzt werden. Die Kinder können zum Beispiel Geschichten entwickeln und dafür Figuren gestalten (Kunst), Dialoge schreiben (Deutsch), Figuren über Landkarten bewegen (Sachunterricht), Parcours entwerfen und programmieren (Sport) oder geometrische Figuren programmieren (Mathematik). Dabei wird die Programmierung als Instrument zur Förderung kreativer und problemlösungsorientierter Fähigkeiten eingesetzt.

doi.org/10.35468/6203-13

Literatur

Alice (2023): Alice. Online unter: https://www.alice.org/ (Abrufdatum: 05.01.2023).

B-Bot (2023): B-Bot. Online unter: https://www.b-bot.de (Abrufdatum: 05.01.2023).

Barradas, R., Lencastre, J., Soares, S. & Valente, A. (2021): The Code.org Platform in the Developing of Computational Thinking with Elementary School Students. In: B. Csapó & J. Uhomoibhi (Hrsg.): CSEDU´20: Computer Supportet Education. 13th International Conference. Wiesbaden: Springer, 118–145.

Bell, T. & Vahrenhold, J. (2018): CS unplugged – how is it used, and does it work? In: Adventures between lower bounds and higher altitudes. In: H.-J. Böckenbauer, D. Komm & W. Unger (Hrsg.): Adventures Between Lower Bounds and Higher Altitudes. Wiesbaden: Springer, 497–521.

Bentz, A. & Standl, B. (2022): Identification of pupils' preferences of patterners and dramatists in secondary school computer science education'. In: Discover Education, 1(11).

Bentz, A. & Standl, B. (2023a): Beyond Gender Differences: Can Scratch Programs Indicate Students' Preferences? In: S. Sentence & M. Grillenberger (Hrsg.): WiPSCE´23: 18. Workshop on Primary and Secondary Computing Education; 27. bis 29. September 2023 Cambridge (England). New York: ACM, Article 8, 1–10. https://doi.org/10.1145/3605468.3609771

Bentz, A. & Standl, B. (2023b): Novice Programmers Conceptions of Loops in K-12 Education in Consideration of Interest and Ability. In: S. Sentence & M. Grillenberger (Hrsg.): WiPSCE´23: 18. Workshop on Primary and Secondary Computing Education; 27. bis 29. September 2023 Cambridge (England). New York: ACM, Article 9, 1–9. https://doi.org/10.1145/3605468.3605506

Bildungsportal NRW (2023): Informatik an Grundschulen. Online unter: https://www.schulministerium.nrw/informatik-grundschulen (Abrufdatum: 05.01.2023).

Brackmann, C. P., Moreno-Leon, J., Roman-Gonzalez, M., Casali, A., Robles, G. & Barone, D. (2017): Development of computational thinking skills through unplugged activities in primary school. In: J. Tenenberg, D. Chinn, J. Sheard & L. Malmi (Hrsg.): ICER´17: 13. International Computing Education Research Workshop; 18. bis 20. August 2017 Tacoma (USA). New York: ACM, 65–72.

Buhnova, B. & Happe, L. (2020): Girl-friendly computer science classroom: Czechitas experience report. In: A. Jansen, I. Malavolta, H. Muccini, I. Ozkaya & O. Zimmermann (Hrsg.): ESCA ´20: the proceedings of 14th european conference on software architecture, gender and diversity in SA track, 14–18.

Cernochová, M., Selcuk, H. & Svoboda, M. (2020): The Role of Algorithmic Thinking Development in the Learning of Elementary School Pupils aged 10–13. London: Routledge.

Checker Welt (o. J.): Der Roboter Check. Online unter: https://youtu.be/DANJgT73XVU?si=xE8T-Z33eY-y-w40n (Abrufdatum: 05.01.2024).

code.org (2023): Code.org. Online unter: https://code.org/ (Abrufdatum: 05.01.2023).

Computer Science Education Research Group der University of Canterbury, Neuseeland (2023): Cs unplugged. Online unter: https://www.csunplugged.org/de/ (Abrufdatum: 05.01.2023).

DelighteX GmbH (2023): ‚Cospaces edu'. Online unter: https://www.cospaces.io/ (Abrufdatum: 05.01.2023).

Denning, P. J. & Tedre, M. (2022): Computational thinking: A disciplinary perspective. In: Informatics in Education, 20(3), 361–390.

Dr. Scratch (2023): Dr. scratch. Online unter: http://www.drscratch.org/ (Abrufdatum: 05.01.2023).

Gomes, A. & Mendes, A. J. (2007): Learning to program – difficulties and solutions. Academic Conference Paper. In: ICEE´07: International Conference on Engineering and Education; 3. bis 7. September 2007 Coimbra (Portugal). New York: ACM, 3–7.

Grabarczyk, P., Freiesleben, A., Bastrup, A. & Brabrand, C. (2022): Computing Educational Programmes with more Women are more about People less about Things. In: Annual Conference on Innovation and Technology in Computer Science Education, ITiCSE, 1, 172–178.

doi.org/10.35468/6203-13

Grover, S. & Basu, S. (2017): Measuring student learning in introductory block-based programming: Examining misconceptions of loops, variables, and Boolean logic. In: R. Davoli & M. Goldweber (Hrsg.): ITiCSE´17: Proceedings of the Conference on Integrating Technology into Computer Science Education, 267–272.

Gulz, A. (2002): Spatially Oriented and Person Oriented Thinking – Implications for User Interface Design. In: A. Tatnall (Hrsg.): Education and Information Technologies. Wiesbaden: Springer, 67–80.

Gulz, A. (2003): Thinking styles and socially enriched learning material: Differential effects on motivation and memory performance. In: Lund University Cognitive Studies, 102.

Happe, L. & Marquardt, K. (2023): Rockstartit: Authentic and inclusive interdisciplinary software engineering courses. In: J. Grundy (Hrsg.): ICSE´23: Proceedings of the ACM/IEEE 45th International Conference on Software Engineering: Workshops Proceedings, New York.

Kanaki, K. & Kalogiannakis, M. (2023): Algorithmic thinking in early childhood. In: L. Gómez Chovez, A. López & I. Candel Torres (Hrsg.): EDULEARN´21: 13. International Conference on Education and New Learning Technologies; 5.bis 6. Juli 2021. IATED: Valentia, 66–71.

Kubo (2023): Kubo. Online unter: https://kubo.education/ (Abrufdatum: 05.01.2023).

Marquardt, K., Wagner, I. & Happe, L. (2023), Engaging girls in computer science: Do single-gender interdisciplinary classes help? In: J. Grundy (Hrsg.): ICSE´23: 45. International Conference on Software Engineering; 14.bis 20. Mai 2023 Melbourne. New York: ACM, 128–140

Michaelis, J. & Weintrop, D. (2022): Interest development theory in computing education: A framework and toolkit for researchers and designers. In: ACM Transactions on Computing Education, 22, 1–27.

Mladenovic, M., Krpan, D. & Mladenovic, S. (2016): Introducing programming to elementary students novices by using game development in python and scratch. In: L. Gómez Chovez, A. López & I. Candel Torres (Hrsg.): EDULEARN´16: 8. International Conference on Education and New Learning Technologies; 4.bis 6. Juli 2016 Barcelona. IATED: Valentia, 1622–1629.

Moreno-Leon, J., Robles, G. & Roman-Gonzalez, M. (2015): Dr. Scratch: automatic analysis of scratch projects to assess and foster computational thinking. In: RED. Revista de Educacion a Distancia, 15(46), 1–23. Online unter: https://www.um.es/ead/red/46/moreno_robles.pdf (Abrufdatum: 05.01.2023).

Moreno-Leon, J., Robles, G. & Roman-Gonzalez, M. (2020): Towards Data-Driven Learning Paths to Develop Computational Thinking with Scratch. In: IEEE Transactions on Emerging Topics in Computing, 8(1), 193–205.

Moses Verlag (2023): Jagd nach der titelstory. Online unter: https://www.moses-verlag.de/ (Abrufdatum: 05.01.2023).

Munasinghe, B., Bell, T. & Robins, A. (2023): Unplugged activities as a catalyst when teaching introductory programming. In: Journal of Pedagogical Research, 7(2), 56–71. https://doi.org/10.33902/JPR.202318546

Open Roberta (2023): Open Roberta. Online unter: https://www.open-roberta.org/ (Abrufdatum: 05.01.2023).

Resnick, M. (2017): Lifelong kindergarten: Cultivating creativity through projects, passion, peers, and play. Cambridge: MIT press.

Richard Garfield (2023): Robo rally. Online unter: https://de.wikipedia.org/wiki/Robo_Rally (Abrufdatum: 05.01.2023).

Rusk, N., Resnick, M., Berg, R. & Pezalla-Granlund, M. (2008): New pathways into robotics: Strategies for broadening participation. In: Journal of Science Education and Technology, 17(1), 59–69.

Schubert, S. Schwill A. (2011): Didaktik der Informatik. Heidelberg: Spektrum Akademischer Verlag.

Scratch (2023): Scratch. Online unter: www.scratch.com (Abrufdatum: 05.01.2023).

Simon, A. & Geldreich, K. (2017): Gender differences in scratch programs of primary school children. In: E. Barendsen & P. Hubwieser (Hrsg.): WiPSCE´17: 12. Workshop on Primary and Secondary Computing Education; 8. bis 10. November 2017 Nimjegen. New York: ACM, 57–64.

Sphero Edu (2021): Sphero edu. Online unter: https://edu.sphero.com/ (Abrufdatum: 05.01.2023).
Standl, B. (2017): Solving Everyday Challenges in a Computational Way of Thinking. In: V. Dagiene & A. Hellas (Hrsg.): Informatics in Schools: Focus on Learning Programming. Basel: Springer International Publishing, 180–191.
steamon.eu (2022): Stem on. Online unter: http://www.steamon.eu (Abrufdatum: 05.01.2023).
Troiano, G. M., Snodgrass, S., Argimak, E., Robles, G., Smith, G., Cassidy, M., Tucker-Raymond, E., Puttick, G. & Harteveld, C. (2019): Is My game ok Dr. scratch?: Exploring programming and computational thinking development via metrics in student-designed serious games for STEM. In: J. A. Fails (Hrsg.): IDC´19: 18. ACM International Conference on Interaction Design and Children; 12. bis 15. Juni 2019 Boise (USA). New York: ACM, 208–219.
Tsarava, K., Moeller, K. & Ninaus, M. (2018): Training Computational Thinking through board games: The case of Crabs Turtles Keywords. In: International Journal of Serious Games, 5(2), 25–44.
Vanicek, J., Dobias, V. & Simandl, V. (2022): Understanding loops: What are the misconceptions of lower-secondary pupils? In: Informatics in Education, 22(3), 525–544.
Wing, J. (2006): Computational thinking. In: Communications of the ACM, 49, 33–35.

Autorin

Bentz, Anette, Dr.
Pädagogische Hochschule Karlsruhe
Institut für Informatik und digitale Bildung
Bismarckstraße 10, 76133 Karlsruhe
Anette.Bentz@ph-karlsruhe.de
Arbeits- und Forschungsschwerpunkte:
Entwicklung und Erforschung innovativer Ansätze
zur Vermittlung von Computational Thinking,
Verküpfung von Informatikausbildung
mit Entrepreneurship Education,
Gestaltung und Evaluation von Lernumgebungen,
die unternehmerisches Denken und Handeln
mit praxisnahen Grundlagen der Informatik verbinden

Diese Arbeit wird unterstützt von der Vector Stiftung.

doi.org/10.35468/6203-13