Algorithmisches Denken und Programmieren im Grundschullehramtsstudium

Luisa Greifenstein, Ute Heuer & Gordon Fraser

Informatische Inhalte finden zunehmend Eingang in die Lehrpläne der Grundschulen. Insbesondere Grundschullehrkräfte empfinden jedoch ihre fehlende informatische Fachkompetenz als Herausforderung. Nach dem TPACK-Modell (Mishra & Koehler, 2006) erfordert das Unterrichten informatischer Inhalte jedoch sowohl technologisches Fachwissen (Technological Content Knowledge) als auch pädagogisches Fachwissen (Pedagogical Content Knowledge): Im Informatikunterricht sollte die Lehrkraft sowohl die Hintergründe der Technologien (TCK) als auch die Lernprozesse in Bezug auf informatische Inhalte und Ziele (PCK) kennen und reflektieren können. Um diesbezüglich Synergien zu nutzen und ein breitgefächertes Fachwissen bei Lehramtsstudierenden zu fördern, bietet sich eine Zusammenarbeit von Fachwissenschaft und Fachdidaktik an. So werden an der Universität Passau am Lehrstuhl für Software Engineering II Werkzeuge zur Unterstützung des Programmierens entwickelt und die Werkzeuge mit schulischer Zielgruppe in Zusammenarbeit mit der Didaktik der Informatik verwendet, evaluiert und weitergedacht. Im Projekt „primary::programming“ liegt der Fokus auf angehenden Grundschullehrkräften. Hier werden u.a. im Seminar „Algorithmisches Denken“ Studierende des Grundschullehramts gefördert, immer mit Blick auf fachliche und fachdidaktische Denkweisen. Um Kriterien für einen Lehrerfolg im Themenbereich „Algorithmisches Denken in der Grundschule“ zu ermitteln, wurden die Studierenden regelmäßig schriftlich befragt. Es zeigt sich, dass die Studierenden insbesondere den hohen Praxisbezug schätzen, sich Unterstützung beim Durchdenken der theoretischen Inhalte wünschen und ihren Lerngewinn in der zukünftigen Arbeit mit den neuen Inhalten in der Grundschule sehen.

1. Einführung

1.1. Programmieren in der Grundschule

Der informatischen Bildung im Primarbereich kommt international sowie auch im deutschsprachigen Raum eine zunehmende Bedeutung zu (Heintz, Mannila und Färnqvist, 2016). So gibt es seit drei Jahren die Empfehlungen der Gesellschaft für Informatik „Kompetenzen für informatische Bildung im Primarbereich“ (Best, Borowski, Büttner, Freudenberg, Fricke et al., 2019). Mittlerweile sind informatische Kompetenzen zudem in der Hälfte der deutschen Grundschullehrpläne explizit als verpflichtender Bestandteil integriert (Nenner & Bergner, 2022). Allerdings bleibt bei neuen Inhalten generell oft die Frage offen, inwieweit Lehrkräfte mit dem neuen (Informatik) Lehrplan arbeiten (Wolff & Martens, 2020). Das wird im Grundschulkontext dadurch verschärft, dass insbesondere Grundschullehrkräfte ihr fehlendes Fachwissen als Schwierigkeit wahrnehmen (Greifenstein, Graßl und Fraser, 2021; Sentance und Cszimadia, 2017). Weiter haben jeweils etwa zwei Drittel der befragten Grundschullehrkräfte und Studierenden laut eigenen Aussagen keine Programmiererfahrung (Brämer et al., 2020). Außerdem haben Grundschullehramtsstudierende vergleichsweise häufig Fehlvorstellungen wie beispielsweise in Bezug auf den Aufbau des Internets (Dengel & Heuer, 2017) und sie verknüpfen mit Informatik abgesehen vom Programmieren fälschlicherweise die Nutzung digitaler Medien (Döbeli Honegger & Hielscher, 2017). Zudem verfügen sowohl Grundschullehramtsstudierende als auch Grundschullehrkräfte über ein relativ geringes individuelles Interesse an der Informatik (Brämer et al., 2020).

Zum einen aufgrund dieser Forschungsergebnisse bezüglich kognitiver und affektiver Faktoren und zum anderen wegen der beschriebenen zunehmenden Integration informatischer Inhalte in den Grundschulkontext sollten entsprechende Inhalte in das Studium des Lehramts an Grundschulen aufgenommen und dabei geeignete Kompetenzerwartungen angezielt werden. Veranstaltungen mit informatischen Inhalten für Grundschullehramtsstudierende sollten (1) einen Praxisfokus aufweisen, u.a. um das Interesse zu erhöhen (Brämer, Rehfeldt, Bauer & Köster, 2020), (2) ausreichend theoretische Inputs geben (Haselmeier, 2019) und (3) gleichzeitig die Studierenden nicht fachlich überfordern (Döbeli Honegger & Hielscher, 2017).

1.2. Veranstaltungen für (angehende) Grundschullehrkräfte

Es gibt vereinzelt Veranstaltungen, die dem begegnen und untersucht wurden: Dabei konnten affektive Aspekte wie das Interesse (Brämer et al., 2020) oder das Selbst- und Fachkonzept (Haselmeier, 2019) gesteigert werden. Auch kognitive und didaktische Aspekte wie Fach- und Vermittlungskompetenzen (Nenner, Damnik & Bergner, 2021) und die Bewältigung von Alltagssituationen (Schmitz, 2022) werden im Kontext der Aus- und Weiterbildung (angehender) Grundschullehrkräfte beforscht.

Während die genannten Seminare eine Einführung in verschiedene Themenbereiche der Informatik geben, behandelt das im Folgenden beschriebene

Seminar „Algorithmisches Denken“ des Projekts „primary::programming“1 ausschließlich Zugänge zum Programmieren in der Grundschule. Dadurch steht ausreichend Zeit zur Verfügung, die zum Verzahnen vielfältiger Theorie- und Praxiselemente genutzt wird. So kann immer wieder Interesse geweckt und Lerngewinn ermöglicht werden.

2. Aufbau des Seminars

2.1. Rahmenbedingungen

Das Seminar richtet sich überwiegend an Grundschullehramtsstudierende – unabhängig von ihrer Fächerkombination oder Semesterzahl. Es können Leistungspunkte für den freien Wahlbereich (Lehramt Grundschule), für das Zertifikat „Information and Media Literacy” oder das Erweiterungsfach Medienpädagogik erworben werden. Das Seminarkonzept wurde von dem interdisziplinären Autorenteam dieses Beitrags gemeinsam entwickelt. In das Konzept konnten so Ideen aus den Bereichen Grundschulpädagogik, Medieninformatik, Didaktik der Informatik und Software Engineering einfließen.

Aufgrund der COVID-19-Pandemie wurde das Seminar im Sommersemester 2020 als asynchrones Online-Seminar erstellt. Das Seminar wurde anhand des Studierenden-Feedbacks und eigener Erfahrungen stetig überarbeitet. So wurde in einem Semester jeweils zwei Studierenden pro Einheit aufgetragen, die benötige Zeit pro Aufgabe zu notieren, um eine passendere Zeitangabe machen zu können, da zuvor von den Studierenden rückgemeldet wurde, dass die Zeitangaben nicht realistisch sind. Die größten Änderungen jedoch bestanden darin, dass auf die Lernplattform ILIAS gewechselt wurde und die letzten Lerneinheiten, in denen es um den Physical-Zugang geht, in Präsenz stattfinden. Die nachfolgenden Ausführungen und die Auswertung des qualitativen Feedbacks beschreiben den aktuellen Stand des Seminars (Sommersemester 2022, n=8). Die Auswertung der quantitativen Selbsteinschätzungen bezieht bis auf den ersten Durchgang zusätzlich auf die vorherigen Semester (n=64).

Das Seminar richtet sich überwiegend an Grundschullehramtsstudierende – unabhängig von ihrer Fächerkombination oder Semesterzahl. Ideen aus den Bereichen Grundschulpädagogik, Medieninformatik, Didaktik der Informatik und Software Engineering sind darin eingeflossen.

2.2. Ablauf

Das Seminar besteht aus 15 Lerneinheiten, davon werden zwölf Einheiten asynchron über ILIAS bearbeitet (wobei die zwölfte Einheit durch einen Unterrichtsbesuch ersetzt werden kann) und drei Einheiten in Präsenz. Der Inhalt der Lerneinheiten wird im Abschnitt „4 Inhalte des Seminars“ näher beschrieben. Im Laufe des Seminars entsteht in Anlehnung an die „Informatikbox” (Haselmeier, 2019) eine „Programmierbox”. Darin sammeln die Studierenden Materialien und Praxisbeispiele samt eigener Bewertungen.

Im Folgenden wird bezogen auf den Aufbau des Seminars die Umsetzung wichtiger Faktoren des multidimensionalen Bedingungsmodells des Lehrerfolgs (Rindermann, 2003) beleuchtet:

Strukturierung: Ein einheitlicher Aufbau, dem jede Lerneinheit in ILIAS folgt, ermöglicht Studierenden, ihre zur Verfügung stehende Lernzeit effektiv zu nutzen. So gibt es für jede Lerneinheit einen Ordner, der wiederum jeweils in einen Überblick mit Lernzielen, drei Bücher („Organisatorisches“, „Theorie“ und „Praxis“) und eine Checkliste eingeteilt ist. Der Ablauf ist jeweils ähnlich: (1) Lernziele und ggf. organisatorische Infos durchlesen (2) Feedback zur letzten Lerneinheit durchlesen und ggf. Abgaben überarbeiten, (3) Theorie-Inhalte durchdenken und eigene wichtige Erkenntnisse im eigenen Theorie-Dokument mitnotieren, (4) Praxis-Inhalte ausprobieren, Materialien bewerten und teilweise selbst erstellen 6) Bearbeitungen mithilfe einer Checkliste durchgehen und auf der Lernplattform abgeben und 7) als relevant erachtete Materialien in der Programmierbox aufbewahren.

Verarbeitungstiefe: Es wird versucht, die Studierenden kognitiv zu aktivieren, indem beispielsweise kleine Reflexions-Aufgaben eingebaut werden. Zusätzlich ermöglichen freiwillig zu erledigende (Teil-)Aufgaben die Vertiefung von Inhalten, die die Studierenden besonders interessieren. Am Ende des Theorie-Teils können die Studierende das Gelernte mittels Selbstkontrollfragen überprüfen.

Kooperativität/Klima: Um das Lernen möglichst flexibel zu gestalten, werden die Studierenden in die Frage nach der Kooperation mit einbezogen: Zu Beginn des Semesters wählen die Studierenden zwischen „Zusammenarbeit“ (einzelne Aufgaben werden gemeinsam bearbeitet und Kontaktmöglichkeit bei Fragen), „Ansprechteam“ (Kontaktmöglichkeit bei Fragen) oder „Einzelarbeit“ (individuelle Bearbeitung). Aus den ersten zwei Varianten werden jeweils Gruppen in ILIAS erstellt, in denen sich die Studierenden austauschen oder Kontaktdaten

1https://www.uni-passau.de/bereiche/presse/pressemeldungen/meldung/lehrerbildung-im-digitalen-zeitalter-freistaat-und-bund-foerdern-zukunftsweisendes-konzept-der-universitaet-passau, zuletzt aufgerufen am 10.11.2022

austauschen können. Allgemein steht ein Forum für Fragen offen und zu Beginn der Veranstaltung stellen sich alle Teilnehmenden im Forum vor, indem sie ihre Programmiererfahrungen und Erwartungen an das Seminar beschreiben. Auch wird Peer-Feedback genutzt, um einzelne Abgaben gemeinsam zu verbessern.

Feedback zu den Lerneinheiten: Direkt vor der Veröffentlichung der jeweils nächsten Lerneinheit bekommen alle Studierenden Feedback zu ihren Abgaben. Darin wird zunächst notiert, ob alles abgegeben wurde und dann meist zu einer Aufgabe ein elaboriertes Feedback gegeben, in dem Fehlvorstellungen diskutiert und Verbesserungstipps gegeben werden.

3. Inhalte des Seminars

3.1. Themenbereich 1 (Lerneinheiten 1 bis 3): Der Unplugged-Zugang

Programmierenlernen kann durch das Arbeiten mit analogen Materialien unterstützt werden. Wenn man dabei auf Mobile Devices und Computer verzichtet, bezeichnet man diesen Zugang oftmals als „unplugged". Daraus ergeben sich Vorteile wie ein fehlender Wartungsaufwand, geringere Anschaffungskosten und insbesondere ein handlungsorientierter Einstieg in das Programmierenlernen.

Durch „hands-on"-Aktivitäten wird eine starke Handlungsorientierung ermöglicht. Beispielsweise können Bausteine der blockbasierten Programmiersprache Scratch (https://scratch.mit.edu/) auf Papier ausgedruckt oder wie im Projekt „primary::programming" aus Holz gefräst werden (Abbildung 1). Durch das Kombinieren der Blöcke und das Testen der entstehenden Skripte durch das Verschieben einer Figur auf einem Spielfeld wird ein haptischer Zugang ermöglicht (Abbildung 2). Gleichzeitig werden die Kinder dadurch, dass (im Gegensatz zur digitalen Programmierumgebung) nur eine Auswahl an Blöcken zur Verfügung gestellt wird, nicht überfordert. Die Studierenden lernen mit „Informatik an Grundschulen" und „Informatik entdecken – mit und ohne Computer" (Günther, 2017) weitere Projekte kennen, die sich mit der Aktivierung der Kinder durch unplugged-Materialien beschäftigen. Eine vermehrt kognitive Aktivierung wird mit den Informatik-Biber-Aufgaben ermöglicht. Durch diese wird algorithmisches Denken – definiert als Denkprozesse wie Abstraktion, Dekomposition, Generalisierung und Evaluation (Bell & Duncan, 2018; Selby & Woollard, 2014) – ermöglicht.

Ob algorithmische Denkprozesse bereits in der Grundschule angebahnt werden sollen, wird teilweise kontrovers diskutiert (https://beat.doebe.li/bibliothek/f00114.html). Tatsächlich kennen Kinder Algorithmen bereits aus ihrer Lebenswelt wie beispielsweise Spielanleitungen oder Backrezepte (Haider & Knoth, 2021).

Besonders affektive Faktoren sprechen für eine frühe Einführung, zum Beispiel um Vorurteilen entgegenzuwirken und das Selbstkonzept aller Kinder bereits frühzeitig zu stärken (Haselmeier, 2019). In einigen Bundesländern wurden informatische Inhalte bereits im Lehrplan der Grundschule verankert (Nenner & Bergner, 2022) und auch die Gesellschaft für Informatik hat „Kompetenzen für informatische Bildung im Primarbereich" (Best et al., 2019) verfasst.

3.2. Themenbereich 2 (Lerneinheiten 4 bis 12): Der softwarebasierte Zugang

Beim softwarebasierten Zugang werden digitale Endgeräte verwendet, um Artefakte zu bearbeiten und zu erstellen. Durch die digitale Programmierumgebung bekommen die Lernen-

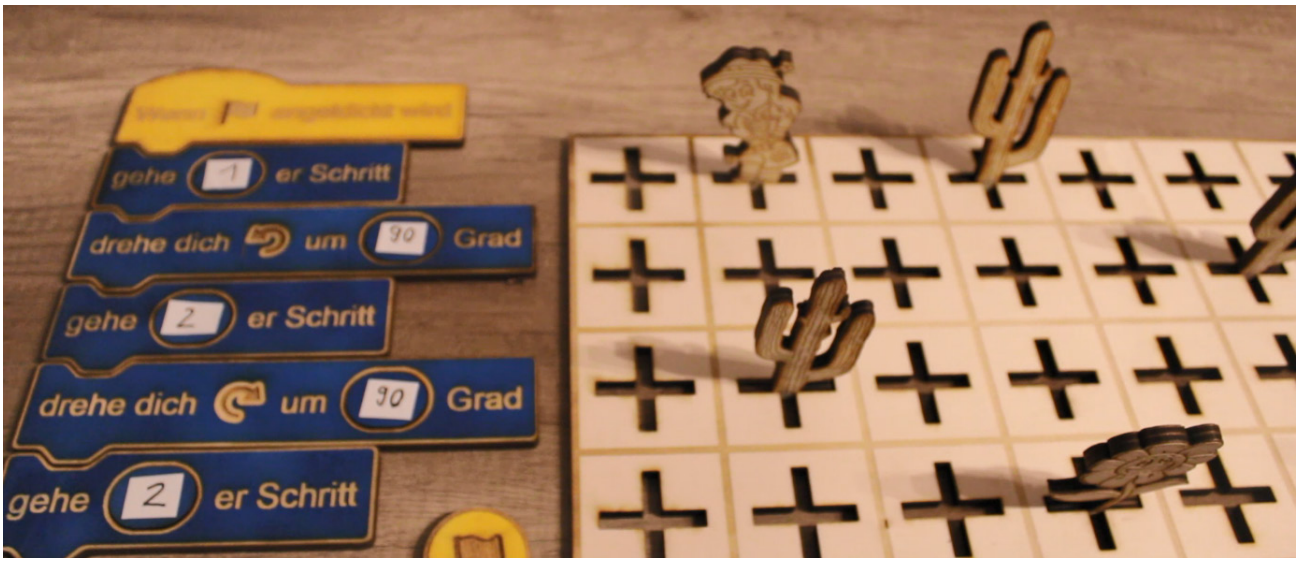

Abbildung 1: Holzspielfeld, -figuren und -blöcke zum Programmieren

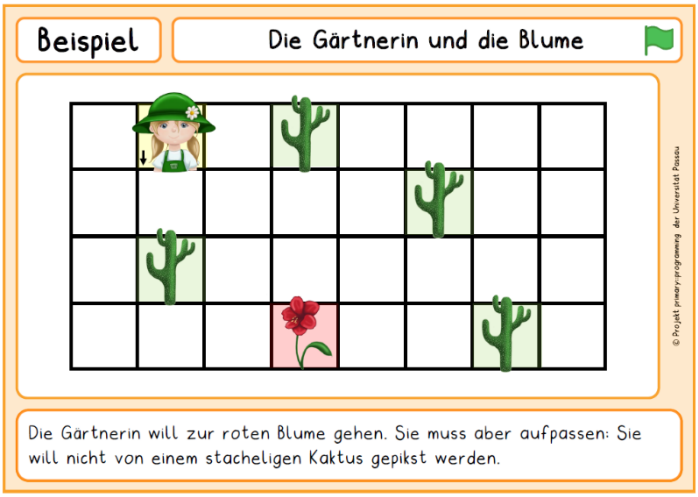

Abbildung 2: Beispielaufgabe des Schatzsuche-Materials

den direktes Feedback. Allerdings muss bei der Vielzahl an Programmiersprachen eine sinnvolle Wahl getroffen werden. Dafür bietet sich das Konzept „Low Floor – Wide Walls – High Ceiling“ an: Einführungsprogrammiersprachen sollen einen leichten Einstieg, verschiedene Zugangsweisen und Artefakte und bei Bedarf auch anspruchsvollere Programme ermöglichen (Weintrop & Grover, 2020). Diese Kriterien und auch das Kriterium der sozialen Dimensionen (Weintrop & Grover, 2020) werden in der blockbasierten Programmiersprache Scratch umgesetzt: Scratch ermöglicht über seine digitale Umgebung einen freundlichen Austausch zu den Programmen, die häufig auch aktuelle Themen einbeziehen (Graßl & Fraser, 2022). Der Fokus im softwarebasierten Themenbereich des Seminars liegt daher auf Scratch. So bearbeiten die Studierenden anfangs selbst grundschulgerechte Aufgaben des „Programmierzirkus“ (https://www.edu.sot.tum.de/ddi/fuer-lehrkraefte/grundschule/) und optional der Programmierlernumgebung „Programmieren mit der Maus” (https://programmieren.wdrmaus.de/). Weitere Aufgabentypen können im Rahmen des Use-Modify-Create-Lehrkonzepts unterschieden werden, so kann vorgegebener Code beispielsweise getestet (= USE) oder verbessert werden (= MODIFY) oder ein ganz eigenes Programm erstellt werden (= MODIFY) (Geldreich, Talbot & Hubwieser, 2019; Lee, Martin, Denner, Coulter, Allan et al., 2011). Im Rahmen einer Lerneinheit wählen die Studierenden aus diesen Aufgabentypen einen aus, erstellen eine passende Aufgabe und überarbeiten ihre Aufgabe in einer späteren Einheit mithilfe eines Peer-Feedbacks.
Beim Erstellen von Aufgaben ist insbesondere auch auf die Lernprozesse dahinter zu achten. Eine Lernzieltaxonomie mit verschiedenen Stufen, bei der ganz grundlegend zwischen dem Lesen und Erstellen von Code unterschieden wird, kann hierbei helfen (Fuller, Johnson, Ahoniemi, Cukierman, Hernán-Losada et al., 2007). Um die Kinder während der Lernprozesse zu unterstützen, muss den Kindern u.a. bei Fehlern in ihren Programmen weitergeholfen werden. Dieser Debugging-Prozess kann Lehrkräfte vor Herausforderungen stellen (Michaeli & Romeike, 2019), was durch teilweise fehlendes Fachwissen von Grundschullehrkräften (Greifenstein et al., 2021; Sentance et al., 2017) nochmal verstärkt werden kann. Deswegen wurde ein strukturiertes Vorgehen vorgestellt (McManus, 2020) und das Analysetool „LitterBox“ zur Unterstützung vorgestellt und ausprobiert. LitterBox wird am Lehrstuhl für Software Engineering II der Universität Passau entwickelt und generiert Hinweise zum Code (Abbildung 3), die neben dem Ort der Fehlerursache eine Erklärung des Fehlers und eine Hilfestellung zur weiteren Vorgehensweise enthalten (Fraser, Heuer, Körber, Obermüller & Wasmeier, 2021). Im Rahmen einer Studie in der Vorlesung von Dr. Stefanie Winkler der Didaktik der Mathematik der Universität Passau wurde festgestellt, dass diese Hinweise helfen, um effektiv und effizient Schülerprogramme zu verbessern (Greifenstein, Obermüller, Wasmeier, Heuer und Fraser, 2021). Zum Debugging-Prozess gehört neben der Verbesserung von Programmen auch das Geben von Feedback. Zum Einüben der Kriterien guten Feedbacks (u.a. Narciss, 2008) und zur Qualitätssicherung der in Lerneinheit 5 erstellten Aufgaben und der in Lerneinheit 7 bearbeiteten Wiki-Seiten geben sich die Studierenden Peer-Feedback. Auch weitere Kriterien, die sich speziell für die Bewertung von Code eignen, wie die Funktionalität, Komplexität und Code-Qualität, werden erarbeitet.
Während die vorherigen Erläuterungen sich auf das blockbasierte Programmieren beziehen, wird in der letzten Lerneinheit dieses Themenbereichs der Übergang vom blockbasierten zum textbasierten Programmieren thematisiert (Strong und North, 2021). Alternativ kann statt dieser Lerneinheit auch an einem Programmierunterricht teilgenommen werden. Die Studierenden haben für den Unterrichtsbesuch wechselnde Wahlmöglichkeiten: Zur Auswahl stehen häufig Workshops des Projekts primary::programming wie „STITCH IT“ (Programmieren von Stickmustern) oder „Ozobot“ (Programmieren mithilfe von Filzstiften und Tablet) oder das CoderDojo des Lehrstuhl für Software Engineering II.

3.3. Themenbereich 3: Der Physical-Zugang

Da sich der physical-Zugang mit der Programmierung von Sensoren und Aktoren von programmierbaren Gegenständen beschäftigt, finden (seit Wintersemester 2021/2022) die letzten Ein-

Abbildung 3: LitterBox-Meldung für das Fehlermuster „fehlende Wiederholung“

heiten an einem Präsenztag in Gruppen von ca. 9 Studierenden statt. Die Verortung des physical-Zugangs erfolgt anhand des „Perspektivrahmen Sachunterricht“ (Gesellschaft für Didaktik des Sachunterrichts, 2013).
Als besonders beliebt, sowohl in der Forschung als auch im Unterricht, hat sich der Roboter „Ozobot Evo“ erwiesen. Die Besonderheit dieses Roboters besteht darin, dass er bereits vorprogrammiert ist, sodass er einer Linie folgen und mit Filzstiften gemalte Farbcodes erkennen, lesen und auf diese Codes reagieren kann. Anhand einiger Arbeitsblätter (https://storage.googleapis.com/ozobot-lesson-library/ozoblockly-training-k-1/ozoblockly-training-k-1.pdf) werden Linienverfolgung und Farbcodes erkundet (Körber, Bailey, Greifenstein, Fraser, Sabitzer & Rottenhofer, 2021) und typische Probleme diskutiert (Greifenstein, Graßl, Heuer und Fraser, 2022). Während sich der Ozobot-Roboter besonders für das Programmieren ohne digitales Endgerät eignet, ermöglicht der „Codey Rocky“-Roboter unkompliziert das digitale blockbasierte Programmieren. Durch verschiedene Aktoren und Sensoren wird das Kriterium der Wide Walls ermöglicht.
Neben dem Einsatz von Robotern, können auch andere programmierbare Geräte im physical-Zugang eingesetzt werden. So werden mithilfe der Programmierumgebung TurtleStitch (https://www.turtlestitch.org/run) werden zuerst Muster programmiert und schließlich auf die Stickmaschine übertragen und auf eine Tasche gestickt. Anschließend werden Verbesserungs- und Erweiterungsvorschläge für die Unterrichtsmaterialien zum Programmieren von Stickmustern des Projekts primary::programming diskutiert.

4. Auswertung

4.1. Qualitative Auswertung des Feedbacks

Jeweils nach Abschluss der drei Themenbereiche unplugged-, softwarebasierter und physical-Zugang evaluierten die Studierenden das Seminar über eine Umfrage in ILIAS: Nach den ersten beiden Zugängen (folglich nach 3 und nach 12 Lerneinheiten) wurde nur Frage (4), nach dem letzten Zugang (folglich nach dem Präsenztag) zusätzlich die Fragen (1) bis (3) gestellt, mit dem Zusatz „Die folgenden Fragen beziehen sich auf alle Lerneinheiten, also das gesamte Seminar.“

(1) „Was hat Ihnen an diesem Seminar gefallen? Was sollte beibehalten werden?“
(2) „Was hat Ihnen an diesem Seminar nicht gefallen? Was sollte geändert werden?“
(3) „Gibt es Aufgaben oder Materialien, die Sie weglassen würden oder die Sie sich (vermehrt) gewünscht hätten? Wenn ja, welche?“
(4) „Möchten Sie sonst noch etwas bzgl. des Seminars mitteilen?“

Die Antworten des letzten Durchgangs (Sommersemester 2022, n = 8) wurden qualitativ ausgewertet, indem Ober- und Unterkategorien gebildet wurden und ihnen die Aussagen zugeordnet wurden. Durch die Kodierung der Antworten ergaben sich vier Oberkategorien, die nicht 1:1 den Fragen der Umfrage zugeordnet werden, da die Studierenden teilweise fragenübergreifend geantwortet haben und um positive und negative Aspekte einer Kategorie gemeinsam diskutieren zu können.
Die Ober- und Unterkategorien werden im Folgenden jeweils geordnet nach ihrer Häufigkeit diskutiert, wobei die Oberkategorien in der Überschrift stehen und die Unterkategorien kursiv geschrieben sind. Die Studierenden werden – um Anonymität zu wahren – S1 bis S8 genannt. Abbildung 4 zeigt die Oberkategorien und bei wie vielen Studierenden diese in den Antworten kodiert wurden. Wenn Studierende verschiedene Unterkategorien erwähnt haben, wurden diese summiert und in Abbildung 4 als „# Nennungen“ notiert. Doppelte Nennungen einer Unterkategorie wurden nicht mehrfach einbezogen.

4.1.1. Strukturierung

Mit zwölf positiven Nennungen von sechs Studierenden und sechs negativen Nennungen von vier Studierenden (Abbildung 4), wurde die Strukturierung jeweils am häufigsten erwähnt. Das ist auch darauf zurückzuführen, dass hierunter vergleichsweise viele Unterkategorien fallen, die im Folgenden vorgestellt werden.

Bezüglich der Strukturierung wurde am häufigsten der *Aufbau* und auch die *Gestaltung* positiv erwähnt. Dabei wurde besonders die Einheitlichkeit hervorgehoben: „Der Aufbau generell ist gut. Durch eine gewisse Routine, die das Seminar gibt, sehr cool zu bearbeiten“ (S3). S6 fand es trotzdem „*abwechslungsreich* und vielseitig“.

Die *klare Aufgabenformulierung* wurde teilweise negativ erwähnt, allerdings jeweils nur in Bezug auf eine Aufgabe bzw. einzelne Aufgaben im Allgemeinen: Bei einem neuen Aufgabentyp in Einheit 3 hätten sich S5 und S7 ein explizites Beispiel gewünscht. Das Feedback von S1 bleibt dagegen vage: „[…] die Aufgaben im Seminar waren gut und meistens klar formuliert. […] Vereinzelte Aufgaben waren für mich nicht ganz klar verständlich.“

	positiv # Studierende	positiv # Nennungen	negativ # Studierende	negativ # Nennungen	∑
Strukturierung	6	13	4	6	29
Theorie / Praxis	6	11	4	6	27
Affektive Bewertung	6	9	0	0	15
Lerngewinn	4	8	1	1	14
∑	22	41	9	13	

Abbildung 4: Oberkategorien und deren Nennung durch die acht Studierenden

Da das Feedback der Studierenden summativ und folglich nicht zeitnah gegeben wird, wird sich ggf. nicht an explizite Aufgaben erinnert. Es könnte die Möglichkeit gegeben werden, dass Studierende bei Verständnisschwierigkeiten direkt im ILIAS-Buch einen Kommentar einfügen, zum einen für (Peer-)Feedback und zum anderen zur Überarbeitung für den nächsten Durchgang.

Bezüglich des *Feedbacks* an die Studierenden gab es – je nach Feedbacktyp – unterschiedliche Meinungen: Während „das wöchentliche Feedback einen immer wieder Motivation [gab]“ (S2), wurde das Peer-Feedback in Einheit 8 teilweise als unpraktisch empfunden: „Schwierig ist es, wenn man Aufgaben beantworten soll, die von anderen Studierenden abhängen. Wenn deren Ergebnisse fehlen bremst es einen selbst aus und man kommt nicht weiter“ (S6). Aus diesem Grund sollte zukünftig zwischen Abgabe und Peer-Feedback zu dieser Abgabe mindestens eine Lerneinheit zwischengeschaltet sein, damit tatsächlich bereits alle Abgaben eingetroffen sind, was aufgrund von Krankheit oder Versäumnis sonst eventuell nicht der Fall ist.

Die *wöchentlichen Abgaben* wurden als „schöne Idee“ (S3) bezeichnet, wobei einzelne Studierende durchblicken lassen, dass die Bearbeitung „teilweise allerdings zeitlich sehr aufwändig“ war (S7) bzw. „Manchmal etwas mehr Zeit investiert werden [musste], was aber in Ordnung war“ (S2). Das könnte u.a. am Theorie-Dokument liegen, wie S1 erklärt: „Alleine zusätzlich unsere Gedanken und Aha-Momente aufzuschreiben ist aufwendig, da viele neue Informationen dazu kommen und erst wieder unterbrochen werden muss um alles zu notieren...“ Zwar soll dadurch die Verarbeitung der Informationen angeregt werden, vielleicht kann es aber wahlweise um ein anderes Format ersetzt werden.

4.1.2 Theorie- und Praxis-Inhalte

Die *Aufteilung in Theorie und Praxis* wurde mehrfach und ausschließlich positiv hervorgehoben. So wurde sie beispielsweise als „sehr gerecht / angemessen“ (S2), „klar[e]“ (S4) und als „Gute Mischung aus theoretischem und eigenem praktischem Arbeiten“ (S8) beschrieben.

Generell wurden die *Materialien* größtenteils beispielsweise als „sehr passend“ (S7) und „hilfreich“ (S8) beschrieben. Nur bezüglich der Beispiele merkt S3 an, dass „Weniger Beispiele. 1-2 konkrete Beispiele“ sinnvoll wären.

Durch den „großen *Praxisbezug* war alles sehr anschaulich und gut vorstellbar“ (S2) und auch die Ergebnisse von Brämer et al. (2020) konnten bestätigt werden, dass durch einen Praxisfokus besonders affektive Faktoren positiv beeinflusst werden können, wie beispielsweise S6 erklärt: „Der Praxistag hat viel Spaß gemacht und man konnte sehr viel eigenständig ausprobieren und einiges kennenlernen.“

Zusammenfassung: Die Studierenden schätzen einen einheitlichen Aufbau und wünschen sich klare Aufgabenformulierungen mit Beispielen. Zudem sollte die Möglichkeit eines zeitnahen Feedbacks (sowohl bei Verständnisschwierigkeiten als auch beim Peer-Feedback) gegeben werden.

Allerdings hätten sich einzelne Studierende noch *mehr Praxis*, teilweise auch in Präsenz gewünscht: „Gerade in Zeiten von Corona natürlich etwas schwierig umzusetzen und im programmiert ist die Theorie sehr wichtig. Allerdings denke ich gerade für die GS ist Praxis sehr wichtig. Deshalb hätte ich mir (nach dem Seminar) noch mehr ‚Praxis‘ mit Robotern oder anderen Geräten vorgestellt“ (S1). Das zeigt wiederum die Bedeutung des Praxistags

in Präsenz, da S1 daran nicht teilnehmen konnte. Zudem wurde zwei Mal „Vielleicht noch ein[…] Praxistag am Anfang des Seminars“ (S2) vorgeschlagen, „damit das Seminar ‚runder‘ wird und man alle zu Beginn kennen lernt“ (S 8).
Es wurden zusätzliche einzelne Verbesserungsvorschläge genannt, die ggf. in den nächsten Durchführungen geeignet umgesetzt werden:
„Eventuell könnten auch *mehr Selbsttestfragen* eingebaut werden. Diese waren sehr hilfreich und haben dazu geführt, dass man über das Gelesene nachdenkt bzw. es ein zweites Mal liest“ (S7).
Auch wurde ein Flipped Classroom-Konzept für den Präsenztag vorgeschlagen: „Bei der Präsenzsitzung hätte ich den Theorieteil am Anfang als Online Angebot besser empfunden. In der Präsenzsitzung dann […] vielleicht die verschiedenen Zugänge der Roboter intensiver Ausprobieren.“ (S7)

4.1.3 Affektive Bewertung

Besonders in Frage (4) zu sonstigen Kommentaren zum Seminar wurde das Seminar häufig positiv affektiv bewertet (Abbildung 4). So hat das Seminar u.a. S2 „sehr Spaß gemacht, vor allem das Programmieren in Scratch“, u.a. S6 „*gut gefallen*“ und S5 ist „sehr *zufrieden* und freu mich jede Woche :)“.
Diese positiven Bewertungen könnten zum einen damit zusammenhängen, dass die Studierenden die Seminare für den freien Bereich selbst auswählen und daher Themenbereiche wählen, die sie interessieren.
Zum anderen könnten die Bewertungen wiederum mit dem Verhältnis von Theorie und Praxis zusammenhängen, da der Praxisbezug das Interesse erhöhen kann (Brämer, Rehfeldt, Bauer & Köster, 2020) und die Beschränkung auf grundlegende Theorie-Inhalte gleichzeitig nicht überfordern (Döbeli Honegger & Hielscher, 2017).

Zusammenfassung: Die Studierenden wünschen sich einen auffallend großen Praxisbezug, z.B. in der Form von Präsenztagen zum Erproben von Materialien. Die Theorie-Teile dagegen scheinen asynchron zu funktionieren, solange ein Durchdenken mit z.B. Selbstkontrollfragen ermöglicht wird.

4.1.4 Lerngewinn

Einzelne Studierende haben explizit angesprochen, wo sie ihren persönlichen Lerngewinn sehen (Abbildung 4).
S1 schreibt „Auch die *Sichtweise* auf das Programmieren in der GS hat sich für mich geändert. […] Trotzdem muss man sich auch als Lehrkraft erstmal in diese Aufgaben hineindenken. Gerade die letzte Einheit fand ich persönlich *nicht einfach*. […] Trotzdem war das Seminar für mich eine große *Bereicherung* im Bezug auf die Fächer in der GS!“
S8 hat „das Gefühl neue, relevante Unterrichtsaspekte zu entdecken. ich würde zukünftig gern selbst mit den vorgestellten Programmen arbeiten […] ich denke, dass ich jetzt deutlich besser darauf vorbereitet bin, Programmieren in der GS einzuführen.“

Zusammenfassung: Die Studierenden waren dem Seminar gegenüber positiv eingestellt. Eine Ausdifferenzierung und mögliche Gründe hierfür gehen aus den anderen Ergebnissen hervor.

4.2 Inwieweit wurde Lehrerfolg wahrgenommen?

Die qualitativen Auswertungen lassen zwar individuelle Erkenntnisse zu, um den Lerngewinn bei allen Studierenden zu untersuchen, werden im Folgenden die quantitativen Daten der Selbsteinschätzung ausgewertet (n=64).
Aufseiten der Studierenden lässt sich der Lehrerfolg mit dem geweckten Interesse, dem Lerngewinn und dem Kompetenzerwerb abfragen (Rindermann, 2003). Die Studierenden schätzten dazu wiederum nach Abschluss jedes Themenbereichs ihre Veränderungen innerhalb einer fünfstufigen Likert-Skala von „trifft gar nicht zu” (= 0) bis „trifft völlig zu” (= 4) selbst ein. Es zeigte sich ein insgesamt recht positives Bild (Abbildung 5, Abbildung 6 und Abbildung 7).
Es gibt allerdings Unterschiede hinsichtlich der drei Themenbereiche (unplugged-, softwarebasierter und physical-Zugang) und der drei Teilaspekte des Lehrerfolgs (Interessenssteigerung, Lerngewinn und Kompetenzerwerb).
Hinsichtlich des Kompetenzerwerbs sind keine signifikanten Unterschiede zwischen den Themenbereichen festzustellen und auch die Mittelwerte ähneln sich stark (unplugged: ø=3,58; softwarebasiert: ø= 3,45; physical: ø=3,56). Bei der Interessenssteigerung und dem Kompetenzgewinn sind dagegen signifikante Unterschiede mit alpha = 0.05 zu verzeichnen. Die Einheiten des physical-Zugang steigerten das Interesse signifikant mehr als die des unplugged-Zugangs ($p=0.017$, $rpb=0.18$) und des softwarebasierten Zugangs ($p<0.001$, $rpb=0.27$). Ähnliches zeigt sich beim eingeschätzten Lerngewinn. Der physical-Zugang ermöglichte

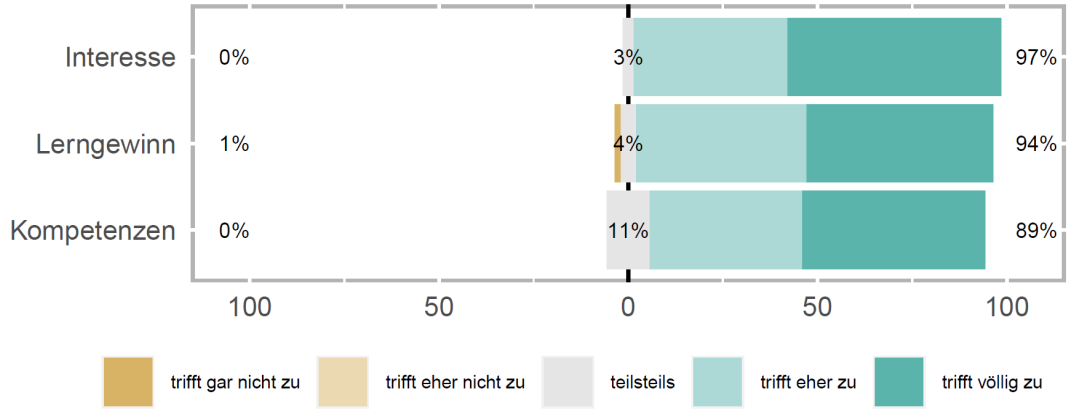

Abbildung 5: Selbsteinschätzung zum „unplugged“-Zugang

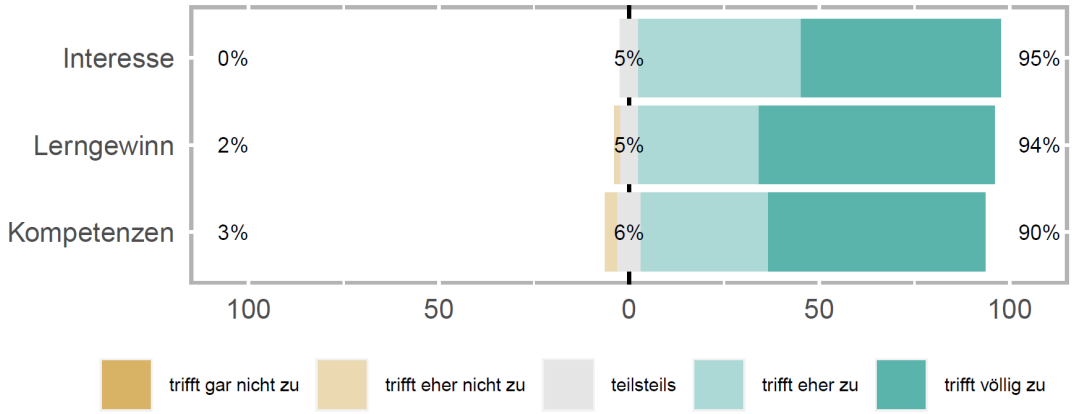

Abbildung 6: Selbsteinschätzung zum softwarebasierten Zugang

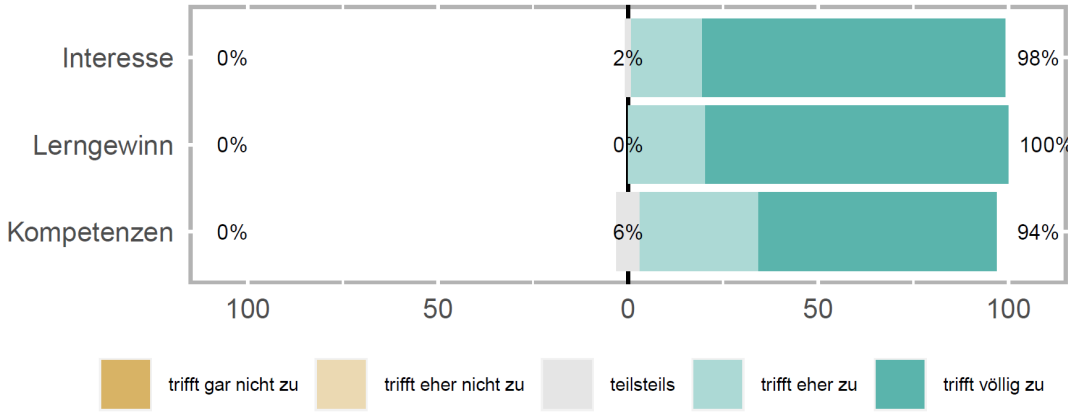

Abbildung 7: Selbsteinschätzung zum „physical“-Zugang

diesen signifikant mehr als der unplugged-Zugang ($p<0.001$, $r_{pb}=0.27$) und der softwarebasierte Zugang ($p=0.019$, $r_{pb}=0.18$).

Die Ergebnisse könnten darauf zurückzuführen sein, dass der physical-Zugang von Anfang an eine praktische Umsetzung fokussiert, was wiederum Interesse erhöhen kann (Brämer, Rehfeldt, Bauer & Köster, 2020). Da die Studierenden zu ihrer Schulzeit vermutlich weniger mit Robotern oder Microcontrollern, sondern hauptsächlich softwarebasiert mit beispielsweise Robot Carol oder Scratch programmiert haben, konnte sich beim Programmieren von Lernrobotern ein erhöhter Lerngewinn einstellen.

Zusammenfassung: Die Studierenden sehen ihren Lerngewinn v.a. im zukünftigen Arbeiten mit den neuen Inhalten in der Grundschule.

5. Fazit und Ausblick

In diesem Beitrag haben wir ein Seminarkonzept vorgestellt, welches Theorie- und Praxiselemente verzahnt. Im Rahmen der Durchführung des Seminars lernten die Grundschullehramtsstudierenden theoretische Inhalte und praxisorientierte Materialien für den unplugged-, den softwarebasierten und den physical-Zugang kennen. Dabei arbeiteten sie u.a. mit verschiedenen Programmierumgebungen, mit einem statischen Analysewerkzeug und mit Lernrobotern. Eine erste Evaluation zeigt, dass die Studierenden den strukturierten Aufbau schätzen

und sich noch mehr Praxisbezug und Beispiele wünschen. Zudem konnte das Seminar sowohl Kompetenzen fördern, ermöglichte einen Lerngewinn und konnte Interesse wecken, wobei die beiden letzteren insbesondere im physical-Zugang ermöglicht wurden. Es ist geplant, einzelne Teile des Seminars weiterhin unter Berücksichtigung des Feedbacks der Studierenden zu ergänzen bzw. zu überarbeiten.

Literaturangaben

Bell, T. & Duncan, C. (2018). Teaching computing in primary schools. Computer Science Education: Perspectives on Teaching and Learning in School, 131.

Best, A., Borowski, C., Büttner, K., Freudenberg, R., Fricke, M., Haselmeier, K. et al. (2019). Kompetenzen für informatische Bildung im Primarbereich. LOG IN, 38(1), 1-36.

Brämer, M., Rehfeldt, D., Bauer, C. & Köster, H. (2020). Vorerfahrungen, Interessen und Selbstwirksamkeitserwartungen von Grundschullehramtsstudierenden und-lehrkräften bezüglich informatischer Inhalte. Didaktik der Physik-Beiträge zur DPG-Frühjahrstagung, 1.

Dengel, A. & Heuer, U. (2017). Aufbau des Internets: Vorstellungsbilder angehender Lehrkräfte. Informatische Bildung zum Verstehen und Gestalten der digitalen Welt.

Döbeli Honegger, B. & Hielscher, M. (2017). Vom lehrplan zur lehrerinnenbildung-erste erfahrungen mit obligatorischer informatikdidaktik für angehende schweizer primarlehrerinnen. Informatische Bildung zum Verstehen und Gestalten der digitalen Welt.

Fraser, G., Heuer, U., Körber, N., Obermüller, F. & Wasmeier, E. (2021). Litterbox: A linter for scratch programs. IEEE/ACM 43rd International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET), 183-188.

Fuller, U., Johnson, C. G., Ahoniemi, T., Cukierman, D., Hernán-Losada, I. et al. (2007). Developing a computer science-specific learning taxonomy. ACM SIGCSE Bulletin, 39(4), 152-170.

Geldreich, K., Talbot, M. & Hubwieser, P. (2019). Aufgabe ist nicht gleich Aufgabe–Vielfältige Aufgabentypen bewusst in Scratch einsetzen. Informatik für alle.

Gesellschaft für Didaktik des Sachunterrichts. (2013). Perspektivrahmen Sachunterricht. Julius Klinkhardt.

Günther, C. (2017). Informatik entdecken–mit und ohne Computer. Informatische Bildung zum Verstehen und Gestalten der digitalen Welt.

Graßl, I., & Fraser, G. (2022). Scratch as Social Network: Topic Modeling and Sentiment Analysis in Scratch Projects. arXiv preprint arXiv:2204.05902.

Greifenstein, L., Graßl, I. & Fraser, G. (2021). Challenging but Full of Opportunities: Teachers' Perspectives on Programming in Primary Schools. 21st Koli Calling International Conference on Computing Education Research, 1-10.

Greifenstein, L., Graßl, I., Heuer, U. & Fraser, G. (2022). Common Problems and Effects of Feedback on Fun When Programming Ozobots in Primary School. The 17th Workshop in Primary and Secondary Computing Education, 1-10.

Greifenstein, L., Obermüller, F., Wasmeier, E., Heuer, U. & Fraser, G. (2021). Effects of Hints on Debugging Scratch Programs: An Empirical Study with Primary School Teachers in Training. The 16th Workshop in Primary and Secondary Computing Education, 1-10.

Haider, M., & Knoth, S. (2021). Algorithmen und logisches Denken. Digitale Medien im Sachunterricht der GS in Theorie und Praxis: didaktisch sinnvolle Einsatzmöglichkeiten und Materialien für alle Themenbereiche, 82-86.

Haselmeier, K. (2019). Informatik in der Grundschule–Stellschraube Lehrerbildung. Informatik für alle.

Heintz, F., Mannila, L., & Färnqvist, T. (2016, October). A review of models for introducing computational thinking, computer science and computing in K-12 education. IEEE Frontiers in Education conference (FIE), 1-9.

Körber, N., Bailey, L., Greifenstein, L., Fraser, G., Sabitzer, B. & Rottenhofer, M. (2021). An Experience of Introducing Primary School Children to Programming using Ozobots (Practical Report). The 16th Workshop in Primary and Secondary Computing Education, 1-6.

Lee, I., Martin, F., Denner, J., Coulter, B., Allan, W. et al. (2011). Computational thinking for youth in practice. Acm Inroads, 2(1), 32-37.

McManus, S. (2020): Debugging In Scratch. Hello World, 13, 70–73.

Michaeli, T., & Romeike, R. (2019). Current status and perspectives of debugging in the k12 classroom: A qualitative study. In 2019 ieee global engineering education conference (educon) (S.. 1030-1038). IEEE.

Mishra, P., & Koehler, M. J. (2006). Technological pedagogical content knowledge: A framework for teacher knowledge. Teachers college record, 108(6), 1017-1054.

Narciss, S. (2008). Feedback strategies for interactive learning tasks. In Handbook of research on educational communications and technology (S. 125-143). Routledge.

Nenner, C. & Bergner, N. (2022). Informatics Education in German Primary School Curricula. International Conference on Informatics in Schools: Situation, Evolution, and Perspectives, 3-14.

Nenner, C., Damnik, G. & Bergner, N. (2021). Weil informatische Bildung nicht erst in der Sekundarstufe beginnen darf: Integration informatischer Bildung ins Grundschullehramtsstudium. INFOS 2021 – 19. GI-Fachtagung Informatik und Schule, 1-10.

Rindermann, H. (2003). Lehrevaluation an Hochschulen: Schlussfolgerungen aus Forschung und Anwendung für Hochschulunterricht und seine Evaluation. Zeitschrift für Evaluation, 2(2), 233-256.

Schmitz, D. (2022). Informatische Anknüpfungspunkte im Alltag von Lehrkräften. 1. Nachwuchskonferenz der Didaktik der Informatik, 31-33.

Sentance, S. & Csizmadia, A. (2017). Computing in the curriculum: Challenges and strategies from a teacher's perspective. Education and Information Technologies, 22(2), 469-495.

Selby, C. & Woollard, J. (2014). Refining an understanding of computational thinking.

Strong, G. & North, B. (2021). Pytch—an environment for bridging block and text programming styles (Work in progress). The 16th Workshop in Primary and Secondary Computing Education, 1-4.

Weintrop, D., & Grover, S. (2020). JavaScript, python, scratch, or something else? Navigating the bustling world of introductory programming languages. Computer science in K-12: An A-to-Z handbook on teaching programming, 99-112.

Wolff, T. B., Hellmig, L. & Martens, A. (2020). STATE OF THE ART IN CURRICULUM RESEARCH FROM THE PERSPECTIVE OF GERMAN COMPUTER SCIENCE TEACHERS. ICERI2020 Proceedings, 9177-9184.