Christine Bescherer und Andreas Fest

Mathematik und informatische Bildung. Programmieren mit Scratch.

Das Projekt dileg-SL (Projektlaufzeit: 2016–2019) sowie die Publikation beim kopaed-Verlag wurden gefördert von der Deutsche Telekom Stiftung.

Auf der kopaed-Seite zum Buch gibt es einen digitalen Anhang zum Download:
https://kopaed.de/dateien/Junge_1106_df_Online-Anhang.pdf

Zitationsempfehlung:
Bescherer, Christine/Fest, Andreas (2019): Mathematik und informatische Bildung. Programmieren mit Scratch. In: Junge, Thorsten/Niesyto, Horst (Hrsg.): Digitale Medien in der Grundschullehrerbildung. Erfahrungen aus dem Projekt dileg-SL. Schriftenreihe Medienpädagogik interdisziplinär, Band 12. München: Verlag kopaed, S. 117-130.

Erschienen in:

Thorsten Junge & Horst Niesyto (Hrsg.):
Digitale Medien in der Grundschullehrerbildung

Erfahrungen aus dem Projekt dileg-SL

kopaed
medienpädagogik interdisziplinär 12

Christine Bescherer und Andreas Fest

Mathematik und informatische Bildung

Programmieren mit Scratch

Die Nutzung des Programmierens zur Entwicklung mathematischer Vorstellungen ist nicht neu, aber trotzdem noch nicht im Grundschulunterricht in Deutschland etabliert. Dieses Konzept wird anhand der didaktischen Begründung eines Beispiels für eine solche Lernumgebung vorgestellt. Ergänzt wird dies durch die Beschreibung der Design-based-Research-Entwicklung eines Seminarkonzepts für Studierende im Lehramt Grundschule zum Programmieren im Mathematikunterricht.

Einführung

Das Teilprojekt „Mathematik und informatische Bildung“ wurde im Rahmen des von der Deutsche Telekom Stiftung finanzierten Projekts „Digitales Lernen Grundschule – Stuttgart/Ludwigsburg“ durchgeführt. Hauptbestandteil des Teilprojekts war die Konzeption und zweimalige Durchführung der Veranstaltung „Computer im Mathematikunterricht (primar)“ an der Pädagogischen Hochschule Ludwigsburg. Die beiden Hochschulseminare richteten sich vorrangig an Studierende des Lehramts Grundschule bzw. Sonderpädagogik im Fach Mathematik im Bereich „fachdidaktische Vertiefung“ sowie an Studierende im Erweiterungsfach Medienpädagogik. Eines der wichtigen Ziele dieser Veranstaltungen war es, mit den Studierenden Lernumgebungen zu entwickeln, in denen Schülerinnen und Schüler der 3. Klasse mathematische Vorstellungen mit Hilfe von Programmieren (weiter)entwickeln können. Dabei sollte neben den mathematischen Erkenntnissen auch die Entwicklung informatischer und algorithmischer Grundkompetenzen der Schülerinnen und Schüler – und der

Studierenden – unterstützt werden. Die Lernumgebungen wurden in Unterrichtsversuchen an unserer Kooperationsschule erprobt.
Informatische und algorithmische Kompetenzen sind notwendig, um mit der fast kompletten Durchdringung unseres Alltags mit Informatiksystemen umgehen zu können (GI, 2019). Andererseits bietet die Informatik besondere Vorgehensweisen, um Probleme zu bearbeiten wie Modularisierung, Wiederverwendung oder systematisches Testen (vgl. ‚Computational Thinking', Grover & Pea 2013).
Durch die kooperative Durchführung der Seminare im Dozenten-Team (in beiden Durchgängen arbeiteten Kolleginnen und Kollegen aus der Mathematik- bzw. Informatikdidaktik sowie der Medienpädagogik zusammen) konnten die Studierenden jeweils auf eine sehr hohe Expertise zurückgreifen. Diese Interdisziplinarität zusammen mit den Erprobungsphasen in einer 3. Klasse der Kooperationsschule erwiesen sich als die Hauptpunkte für den Erfolg.

Programmieren zum Mathematiklernen

In der Mathematikdidaktik gibt es eine lange Tradition, in der in konstruktivistischen Lehr-/Lernszenarien die Entwicklung mathematischer Vorstellungen oder mentaler Modelle durch eigenes Programmieren in kindgerechten Sprachen wie Logo oder Scratch gefördert werden (u.a. Papert 1975; Löthe 1985, Noss, Healy, Hoyles 1997 oder Benton, Hoyles, Kalas, & Noss 2017). Dieser Tradition liegt die Thematik zugrunde, wie mathematische Vorstellungen oder Bedeutungen aus der Verbindung von „altem" Wissen und „neuen" Fragestellungen – mit oder ohne entsprechende Experimentierumgebungen – konstruiert werden können. Woher weiß ein Kind (oder auch Erwachsener), ob seine mathematischen Vorstellungen/ Bedeutungszuschreibungen korrekt sind? Und wie kann es – und eventuell auch die Lehrkraft – nachvollziehen, welche Änderung in den mathematischen (Fehl-)Vorstellungen dann eine bessere Vorstellung hervorbringt?
In der Idee des Programmierens oder neuerdings „Coding" werden vor allem drei Aspekte des Mathematiklernens realisiert:
Wenn Kinder selbst programmieren/Code schreiben, dann

- be***greif***en sie die Abstraktionen, die den Kern der Mathematik darstellen
- modellieren sie ***dynamisch*** mathematische Konzepte und Beziehungen
- gewinnen sie ***Selbstvertrauen*** in ihre Fähigkeiten und ihr Wirken als Mathematiklernende (in Anlehnung an Gadanidis & Silver 2017, S. 1)

Wie sieht so eine Lernumgebung - oder in diesem Kontext auch „Mikrowelt" (Kynigos 2007) genannt - aus?

In den Seminaren wurde ein Planungsraster vorgegeben, das den Studierenden ein möglichst weitgehendes Gerüst (im Sinne von Scaffolding wie z.B. im Cognitive Apprenticeship-Ansatzes vgl. Collins, Brown & Holum, 1991) bei der Entwicklung und Planung der Unterrichtseinheiten bieten sollte. Dieses Planungsraster wird im Folgenden genutzt, um anhand eines Beispiels für eine prototypische Lernumgebung die mathematik- und informatikdidaktischen Begründungen darzustellen.
Das gesamte Raster im Überblick einschließlich (hochschul)didaktischer Begründungen findet sich im digitalen Anhang.

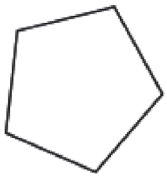
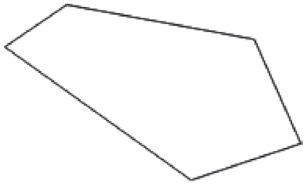
Thema: Untersuchung regelmäßiger Vielecke

Mathematische Erkundungen/Entdeckungen: *Die Schülerinnen und Schüler sollen durch die Untersuchung der regelmäßigen Vielecke ein Verständnis für die geometrischen Eigenschaften sowie die „Bestimmungsstücke" dieser ebenen Figuren bekommen. Mathematisch bedeutet dies, dass die Seitenlängen alle gleich sind und die Drehwinkel (Außenwinkel) jeweils 360°/Anzahl der Ecken betragen. Damit reichen zwei „Bestimmungsstücke" (Eckenanzahl und Seitenlänge), um ein konkretes Vieleck zu konstruieren.*
Ähnlich wie die meisten Personen beim Begriff Viereck sofort an ein Quadrat denken, ist es auch bei den anderen Vielecken. Mit den Benennungen Fünfeck, Zehneck usw. werden automatisch die regelmäßigen Varianten verbunden. Dies macht es schwer, die Besonderheiten dieser Figuren im Gegensatz zu den nicht-regelmäßigen, aber vielleicht z.B. trotzdem symmetrischen Vielecken zu verinnerlichen.
Ein anderes Lehr-/Lernszenario um die Besonderheit regelmäßiger Vielecke zu erkennen, wäre ein Einstieg mit vielen verschiedenen unregelmäßigen, symmetrischen und regelmäßigen Vielecken, die z.B. als reale Figuren aus Karton vorliegen. Die Schülerinnen und Schüler müssten dann die Figuren in verschiedene Kategorien sortieren und die Gemeinsamkeiten bzw. Unterschiede beschreiben. Dies ist für Grundschülerinnen und -schüler sehr komplex, da es zum einen ein hohes Abstraktionsvermögen[1] erfordert und ihnen andererseits die Erfahrung mit allgemeinen Vielecken meist noch völlig fehlt.
Durch das Programmieren regelmäßiger Vielecke werden die Gemeinsamkeiten aller dieser Figuren (gleiche Länge der Seiten und gleiche Winkel an allen

1 Zur Entwicklung abstrakter Begriffe in der Geometrie vgl. Franke & Reinhold 2016, S. 115 ff

Ecken) sehr plastisch durch den Schleifenbefehl („Wiederhole x-mal") fassbar.[2] *Weiter müssen die Schülerinnen und Schüler Vorstellungen zur Orientierung in der Ebene nutzen bzw. erst entwickeln, z.B. wie weit vorwärts gegangen bzw. nach rechts oder links gedreht werden muss.*
Außerdem können sehr gute Schülerinnen und Schüler hier schon eine erste Vorstellung zum Variablen-Begriff bekommen, indem sie z.B. nur die Länge der Seiten ihrer Vielecke variieren. In dem Programm für die konkrete Figur bleibt alles gleich, nur die Angabe der Länge ändert sich. Durch Probieren kann man leicht erkennen, dass die Variation des Winkels sehr überraschende Effekte verursacht.
Eine mögliche Weiterarbeit ist der Auftrag: „Lasse den Computer Muster aus mehreren regelmäßigen Figuren zeichnen." Oder für sehr gute Schülerinnen und Schüler auch: „Schreibe ein Programm, in dem du nur die Anzahl der Ecken und die Länge der Seite eingeben musst und der Computer zeichnet das richtige Vieleck."
Bei dem Musterauftrag kommt noch eine gewisse erschwerte Orientierung in der Ebene hinzu, da Bewegungen mit und ohne „Strich" unterschieden werden müssen.

Auftrag: *Regelmäßige Vielecke (Drei-, Vier-, Fünf-, Sechs-, Zehnecke usw.) bestehen nur aus gleich langen Seiten.*
Beispiele:

Das ist ein regelmäßiges Fünfeck.	*Das ist auch ein Fünfeck, aber nicht regelmäßig.*
	

Lass den Computer verschieden große, regelmäßige Drei-, Vier-, Fünf-, Sechs-, Zehnecke zeichnen. Beschreibe was bei den Programmen für den Computer immer gleich ist und wie sich die Programme unterscheiden (müssen).

2 Natürlich können auch regelmäßige Vielecke ohne Schleife gezeichnet werden, durch einfache Mehrfachausführung derselben Bewegungen. Dies erlaubt z.B. schwachen Schülerinnen und Schülern einen Erfolg. Aber das Ziel der Unterrichtseiheit besteht darin, dass eine Schleifenstruktur genutzt wird.

Tipp: Beginne mit verschiedenen regelmäßigen Vierecken, also Quadraten.

Vorgehen:
Die Schülerinnen und Schüler müssen erkennen, dass die „Computerfigur" (Scratch) oder der „Igel" (Logo) genaue Anweisungen brauchen, um die regelmäßigen Vielecke zeichnen zu können. Welche Anweisungen das sind, können die Kinder selbst entdecken.
Deshalb wird zuerst das „Laufe ein Quadrat"-Spiel gespielt. Auf dem Schulhof oder im Klassenzimmer wird ein Quadrat auf den Boden gezeichnet bzw. mit Kreppband geklebt. Dann werden die drei Rollen „Läufer/in", „Begleiter/

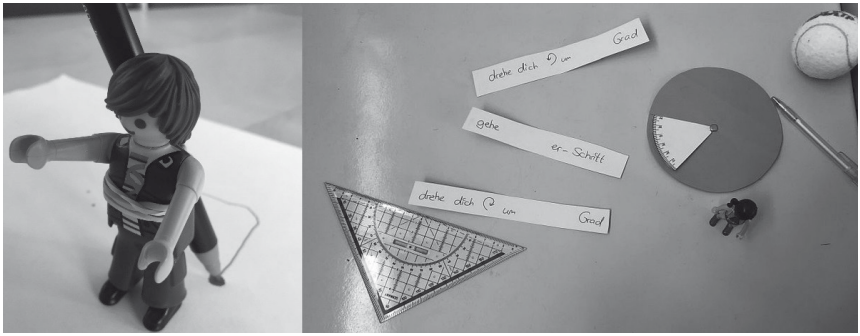


Abb. 1: Materialien für den Unplugged-Einstieg

in" und „Anweiser/in" verteilt. (Es ist wichtig, dass jedes Gruppenmitglied im Rahmen der Einführung jede der Rollen übernimmt, da dabei jeweils verschiedene Sichtweisen auf die spätere Programmierung ermöglicht werden.) Dem/der Läufer/in werden die Augen verbunden und die Anweisungen „Vorwärts x Schritte" (Dabei muss der/die Begleiter/in darauf achten, dass geradeaus gelaufen wird und keine Hindernisse im Weg sind.) und „Nach rechts/links drehen". (Hier wird erst mal die Gradzahl, um die gedreht werden soll, noch nicht thematisiert, dies kommt erst später.) Nachdem alle Kinder der Gruppe alle Rollen durchlebt haben, wird anschließend das Zeichnen eines Quadrats durch eine „Playmobil-Figur" im Klassenzimmer durchgeführt.
Dazu wird einer Playmobil-Figur ein Stift am Rücken befestigt (s. Abb. 1 links). Analog zu den Anweisungen vom Schulhof sollen jetzt die Anweisungen für die Figur gesammelt werden. Hier kommen jetzt die Angaben zum Winkel, um den gedreht werden muss, hinzu. Dazu eignen sich Winkelscheiben wie sie in Abb. 1 (rechts) gezeigt werden:

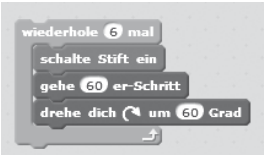
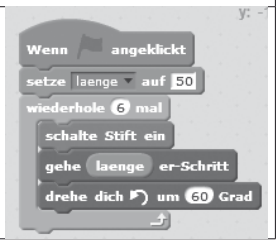
Nachdem alle Schülerinnen und Schüler Erfahrungen mit dem Geben bzw. dem Befolgen der Anweisungen gemacht haben, werden diese Anweisungsfolgen mit vorbereiteten Papierstreifen gelegt.

Programmierumgebung in Logo/Scratch:
Je nach verwendeter Programmiersprache werden nach dem unplugged-Einstieg die notwendigen Anweisungen auf Papierstreifen geschrieben und zusammengelegt. Für schwache Schülerinnen und Schüler sind die Streifen schon vorbereitet und sie müssen die Anzahl der Schritte, bzw. die Gradzahl der Drehung eintragen und sie dann in der richtigen Reihenfolge zusammenfügen. Sobald dies geschehen ist, also ohne Zwischenkorrektur durch die Lehrkraft, sollen die Kinder diese Befehle am Computer ausprobieren. Sie müssen dann noch darauf hingewiesen werden, wie sie systematisch variieren können und am Ende das „richtige" Programm aufschreiben bzw. aus den Streifen legen und mit Klebeband zusammenkleben sollen.
Die Kinder können nun auf sehr unterschiedlichen Niveaus arbeiten. Einerseits ist es schon möglich, die regelmäßigen Vielecke „stückweise" zu erzeugen, z.B. vw 100 re 120 vw 100 re 120 vw 100[3]*. Dies ist aber nicht das Ziel, denn es soll die Figur im Ganzen verstanden werden. Dazu müssen die Kinder zum Erstellen von „eigenen Programmen" angeregt werden.*
Als weitere Materialien neben der Befehlsliste (s.u.) werden Vorlagen mit allen Programmierschritten, aber ohne konkrete Werte bereitgehalten, die die Schülerinnen und Schüler für die Dokumentation ihrer „korrekten" Programme verwenden können.

Befehlsliste[4]

Was soll der Computer tun?	*Logo*	*Scratch*
Zeichne einen geraden Strich der Länge … .	*vorwärts … / vw …* *Bsp. vw 100*	
Drehe dich um … Grad nach rechts.	*rechts … / re …* *Bsp. rechts 90*	

3 Programmiersprache Logo: vw 100" … vorwärts um 100 Schritte und „re 120" … nach rechts drehen um 120°
4 Jede Schüler/innengruppe bekommt die Liste selbstverständlich nur in einer der beiden Programmiersprachen.

Drehe dich um … Grad nach links.	*links … / li …* *Bsp. links 45*	
Wische alle Striche weg.	*bild*	
Bewege dich … vorwärts ohne Strich.	*stifthoch vorwärts …* *stiftab / sh vw … sa* *Bsp. sh vw 75 sa*	
Wiederhole … mal.	*wiederhole […] / wh […]* *Bsp. wh 6 [vw 60 re 60]*	
Wähle eine andere Farbe.	*farbe [… … …] (Werte für rot, grün und blau zwischen 0 und 255 einsetzen)* *Bsp. farbe [100 200 50]*	 *(Durch Klicken auf die gewünschte Farbe irgendwo auf dem Bildschirm, ändert sich die Farbe.)*
Schreibe ein Programm	pr name :Eingabe … ende Bsp. pr figur :s wh 6 [vw :s li 60] ende	

Zeitplanung bei 3-4 Schüler/innen pro (studentische) Lehrkraft[5]*:*
Unplugged-Einstieg: 20 min
Übergang zum Programmieren: 10 min
Programmieren: 45 min
Vergleichen der Ergebnisse, d.h. der erstellten Programme und Diskussion der Übereinstimmungen und Unterschiede: 15 min

5 Das Lehr-/Lernszenario ist hier für die Durchführung durch mehrere Lehramtsstudierende in einer Klasse geplant. Wenn eine – erfahrene – Lehrkraft dieses Szenario alleine durchführt, müssen die Zeiten und Hilfsmaterialien entsprechend angepasst werden.

Name	*Anzahl der Ecken*	*Anzahl der Seiten*	*Gradzahl um die gedreht wird*	*Anzahl der Ecken * Winkel*
Dreieck	*3*	*3*	*120°*	*3*120° = 360°*
Viereck (Quadrat)	*4*	*4*	*90*	*4*90° = 360°*
Fünfeck	*5*	*5*	*72*	*5*72° = 360°*
Sechseck	*6*	*6*	*60*	*6*60° = 360°*
…	*…*	*…*	*…*	*…*
Zehneck	*10*	*10*	*36*	*10*36° = 360°*

***Abschluss*:**
Die Schülerinnen und Schüler sollen die Programme, die korrekte regelmäßige Vielecke ergeben, festhalten. Dies kann entweder durch speichern unter einem geeigneten Namen, aufschreiben oder durch zusammenkleben der vorgefertigten Streifen in der richtigen Reihenfolge geschehen. Zum Abschluss werden dann die Programme in der Gruppe oder in der gesamten Klasse verglichen. Dabei soll den Kindern bewusst werden, dass bei allen regelmäßigen Vielecken mit einer bestimmen Eckenzahl immer dieselbe Gradzahl zur Drehung nach rechts oder links benutzt werden muss. Die Anzahl der Schritte ist auch immer gleich und ergibt die Größe des Vielecks. Diese Erkenntnisse werden an der Tafel oder über Beamer in einer Tabelle festgehalten:

Bewertung/Rückmeldung/Anerkennung:
Alle Kinder dürfen am Ende ihre Programme und ihre Ideen vorstellen und bekommen so die Rückmeldungen, ob ihre Vorstellungen korrekt sind oder nicht. Wenn Kinder große Schwierigkeiten mit dem Erstellen der Programme hatten, so können sie ihre „schrittweise" Lösung vorstellen und sie mit den Lösungen der anderen vergleichen. Eine geeignete Rückfrage wäre in diesem Fall: „Was kannst du mit deinem Programm noch machen, dass es so wird wie das von XY?"

***Computational Thinking/Informatisches Denken*:**
Zum einen sollen die Kinder durch das systematische Probieren und Dokumentieren der richtigen Lösungen die typischen Aktivitäten „Debugging und systematische Fehlersuche" selbst durchführen. Weiter müssen sie zur Lösung der Aufgabe auch folgende Aspekte des Computational Thinkings (Grover & Pea, 2013) durchführen:

- Abstrahieren und Verallgemeinern von Mustern (Erkennen, was an den Vieleck-Programmen gleich und was unterschiedlich ist.)
- Symbolische Darstellungen und Datenrepräsentationen (Die Programme an sich sind symbolische Repräsentationen und bei der Verwendung von Variablen befinden sie sich auf einer sehr abstrakten symbolischen Ebene.)
- Algorithmische Vorstellungen zu Abläufen und ihrer Steuerung (Die Programme sind Realisierungen dieser algorithmischen Vorstellungen.)
- Strukturierte Zerlegung von Problemen/Modularisierung (Falls die Kinder die Muster aus den regelmäßigen Vielecken erstellen, arbeiten sie konkret mit verschiedenen Modulen.)

Je nach Zeit soll am Ende noch mal besprochen werden, was wir eigentlich genau gemacht haben. Hier besteht dann auch die Gelegenheit, z.B. „das systematische Testen, das man beim Programmieren immer machen muss" oder die „Wiederverwendung von schon fertigen Programmen (Modularisierung)", zu thematisieren.

Reflexion:
In diesem Szenario interessiert besonders, wie weit die Schülerinnen und Schüler abstrahieren können und zwar sowohl auf der Programmierebene, wie auch bei den mathematischen Vorstellungen. Können sie also z.B. von einem Sechseck auf ein Zwölfeck schließen, ohne dass sie dies konkret programmiert haben.

Die didaktischen Begründungen[6] stammen sowohl aus der Mathematik-, der Informatik- wie auch der Hochschuldidaktik und sollen hier exemplarisch aufzeigen, wie das Programmieren im Mathematikunterricht der Grundschule tatsächlich zu neuen mathematischen Erkenntnissen beitragen kann und gleichzeitig das informatische Denken erlebbar machen kann. Außerdem soll deutlich gemacht werden, wie zukünftige Lehrende solche Lehr-/Lernszenarien planen können.
Programmieren in der Grundschule muss nicht mit der Entwicklung mathematischer Vorstellungen verknüpft sein, so gibt es viele Ansätze z.B. die Nutzung und Bedienung von Sensoren bei Calliope mini (https://calliope.cc/) im Sachunterricht oder die Konzepte der „Zauberschule Informatik" der RWTH Aachen (http://schuelerlabor.informatik.rwth-aachen.de/module/zauberschule). Dort werden mathematische Grundlagen der Informatik wie z.B. Binärzahlen oder Markow-Ketten in Form von Spielen oder „Zaubereien" für Grundschülerinnen und -schüler erfahrbar gemacht.

6 Weitere didaktische Begründungen finden sich in dem Planungsraster im digitalen Anhang.

Durchführungen der beiden Seminare

Die Seminare „Computer im Mathematikunterricht“ fanden im Wintersemester 2017/2018 sowie im Sommersemester 2018 im wöchentlichen Rhythmus statt. Während in der Pilotphase 17 Studierende teilnahmen, wirkten in der Durchführungsphase acht Studierende mit. Dies ist aber nicht auf gesunkenes Interesse der Studierenden zurückzuführen sondern auf terminliche Überschneidungen mit anderen Veranstaltungen. (Angemeldet hatten sich 18 Studierenden, aus unterschiedlichen Gründen reduzierte sich die Teilnehmer/innenzahl dann auf 8.)
Durch die kooperative Durchführung der Seminare im Dozenten-Team konnte für die Aspekte Mathematikdidaktik (Bescherer), Medienpädagogik (Junge) und Informatikdidaktik (Fest) jeweils auf eine sehr hohe Expertise zurückgegriffen werden, was sich als äußerst erfolgreich erwies.
Die teilprojektspezifische Begleitforschung[7] folgte dem Design-based-Research-Ansatz zur Entwicklung des Seminarkonzepts kombiniert mit der Verfeinerung von Wirkmodellen (Wachsmuth & Hense 2016). Den letzteren Ansatz wählten wir, da durch die geringen Fallzahlen sowohl der Studierenden als auch bei den Schülerinnen und Schülern im Rahmen einer solchen Entwicklungs- und Machbarkeitsstudie im Feld kein experimentelles Design zum Nachweis der Wirksamkeit leistbar ist. Die erwartete Wirkung bezieht sich vor allem auf den Zuwachs an Fähigkeiten im Umgang mit den Elementen des Computational Thinkings sowie z.B. auch Änderungen in der computerspezifischen Selbstwirksamkeitserwartung. Solche Neuerungen, insbesondere wenn sie mit technischem Aufwand verbunden sind, können nur gelingen, wenn die Protagonisten entsprechende Haltungen bzw. Einstellungen dazu entwickeln. Unser Ziel war hier, dass den Studierenden bewusst wird, welche Möglichkeiten für den konkreten Einsatz die Nutzung von Programmieren zum Mathematiklernen bietet und welcher Aufwand damit verbunden ist.

Ergebnisse

Die Evaluation der Pilotphase (WiSe 2017/18) erfolgte über eine Vorher-Nachher-Befragung[8] der Studierenden zur Computernutzerselbstwirksamkeit CUSE (Spannagel & Bescherer 2009) sowie zur Selbstwirksamkeit in Bezug auf den Einsatz von Computern im Unterricht SECU (Dinse da Salas, Rohlfs & Spanna-

7 Zur Evaluation des Gesamtprojekts siehe das Kapitel Evaluation im Projekt „Digitales Lernen Grundschule“ in diesem Band.

8 14 Personen hatten sowohl die Vor- wie auch die Nachher-Befragung ausgefüllt. In der zweiten Durchführung der Veranstaltung haben wir aufgrund der geringen Teilnehmer/innen-Zahl auf den Einsatz des Fragebogens verzichtet.

gel 2016), die Analyse der Lernumgebungen sowie Screenvideos der Programmierphasen der Kinder und eine Abschlussdiskussion mit den Studierenden am Ende des Semesters.

In den Befragungen zur Selbstwirksamkeitserwartung konnte statistisch kein Unterschied zwischen den Mittelwerten der Vorher- und Nachher-Ergebnisse festgestellt werden. Bei Betrachtung der Mittelwerte nahm die Computernutzerselbstwirksamkeit sogar leicht ab (Mittelwert $CUSE_{pre}$ 57,00 / Mittelwert $CUSE_{post}$ 55,46)[9]. Da die Situation, in der die 17 Studierenden mit 16 Schülerinnen und Schüler die Lernumgebungen erprobt hatten, sehr weit von einem normalen Mathematikunterricht entfernt war, ist der gleichbleibende Wert des SECU-Mittelwerts nicht überraschend. Enttäuschend dagegen ist der Abfall der Mittelwerte der drei Fragen im CUSE-Fragebogen, die sich auf das Lernen mit Computern beziehen:

- Item 5: *Das Verwenden von Computern macht Lernen interessanter.* $Mittelwert_{pre}$ = 4,64 / $Mittelwert_{post}$ = 3,86[10]
- Item 8: *Computer sind gute Hilfsmittel beim Lernen.* $Mittelwert_{pre}$ = 4,86 / $Mittelwert_{post}$ = 4,38
- Item 10: *Einige Computerprogramme machen Lernen eindeutig einfacher.* $Mittelwert_{pre}$ = 5,29 / $Mittelwert_{post}$ = 4,43

Dieser Befund ist umso bedauerlicher, da die computerbezogene Selbstwirksamkeitserwartung ein sehr wichtiger Prädiktor zur Nutzung von digitalen Medien/Werkzeugen im Unterricht ist (Kreijns, Van Acker, Vermeulen, & Van Buuren 2013). In der Diskussion mit Kolleginnen und Kollegen auf der Tagung der Didaktik der Mathematik 2018 in Paderborn wurde jedoch rückgemeldet, dass solche Ergebnisse bei Lehramtsstudierenden nach der ersten Auseinandersetzung mit dem Thema Computer beim Mathematiklernen durchaus üblich sind. (Wenn solche Ergebnisse i.A. auch nicht veröffentlicht werden.) Eine mögliche Erklärung dafür ist, dass die Studierenden erstmals erkennen, welche Kenntnisse und digitalisierungsbezogene Kompetenzen für einen erfolgreichen Einsatz von Computern im Unterricht (tatsächlich) notwendig sind. Diese Einschätzung kann auch durch eine Aussage im Projektjournal einer Teilnehmerin aus dem Sommersemester 2018 belegt werden: *„…, dass wir als angehende Lehrkräfte im Bereich Technologie und Medien viel zu wenig aufgeklärt werden. Es ist wichtig die Schülerinnen und Schüler mit Medien vertraut zu machen und den richtigen Umgang damit zu vermitteln, weswegen ich es unumgänglich finde,*

9 Fragebogen mit 12 Items, 6-teilig likertskaliert mit Werten 1 bis 6.
10 Minimum 1 und Maximum 6

sich in dieser Hinsicht als Lehrperson zu informieren." (Aussage einer Studentin aus dem Sommersemester 2018)

Die Studierenden hatten im ersten Durchgang in ihren Planungsrastern große Schwierigkeiten, die mathematischen Grundlagen ihrer Lernumgebungen darzustellen. Dagegen wandten sie viel Mühe für die Gestaltung des „unplugged-Einstiegs" und der grafischen Oberfläche der Lernumgebungen auf. Deshalb wurden in der zweiten Durchführung die mathematischen Inhalte auf „Muster aus regelmäßigen Vielecken" und „Orientierung in der Ebene" eingeschränkt.
Die Analyse der Screenvideos zeigt, dass zwar immer wieder das für das Computational Thinking typische „Testen" und „Debugging" zu erkennen sind. Beides bezieht sich jedoch vor allem auf den Umgang mit Scratch an sich (beim Ziehen und Zusammenklicken der Blöcke sowie Eingeben von Parametern). Eine systematische Überprüfung der mathematischen Vorstellungen und entsprechende Anpassung des Programmes sind praktisch nicht erkennbar.
In der Reflexionsrunde wurde für die schwache Auseinandersetzung mit den mathematischen Inhalten – sowohl der Studierenden wie auch der Schülerinnen und Schüler – vor allem der hohe Zeitaufwand für die Einarbeitung in Scratch und auch die Faszination mit den Möglichkeiten von Scratch genannt. Als sehr positiv an dem Seminar an sich bewerteten die Studierenden den „unplugged-Einstieg" sowie die Möglichkeit, ihre Lernumgebung ganz frei zu wählen und „was eigenes zu machen".
Insgesamt sind in den beiden Seminaren Lernumgebungen zu folgenden Themen entstanden:

- kleinstes gemeinsames Vielfaches
- Simulation von Ampelschaltungen
- Weg der Müllabfuhr (Eulerkreis)
- Orientierung auf der Ebene (Himmelsrichtungen/Koordinatensystem/rechts, links)
- Kombinatorik
- Muster aus Quadraten erstellen
- Talervervielfältigung (Funktionsmaschine).

Die Schülerinnen und Schüler (9 Mädchen und 9 Jungen) arbeiteten wie zu erwarten auf sehr unterschiedlichem Niveau. Von den 18 Kindern haben 8 Eltern, die beide im Ausland geboren wurden und nur 3 haben Eltern, die beide in Deutschland geboren wurden. Außerdem sind noch einige Inklusionskinder in der Klasse.
Die Erprobungen an der Kooperationsschule erfolgten zu einem gemeinsamen Termin, sodass die Studierenden nicht mit der gesamten Klasse sondern in

Kleingruppen arbeiteten (Pilotphase: zwei Studierende arbeiteten mit jeweils zwei SuS; Durchführungsphase: zwei Studierende betreuten vier bis fünf SuS). Durch die enge Betreuung der Kleingruppen durch die Studierenden konnten jedoch alle Schülerinnen und Schüler aktiv mitarbeiten. Da keine Gruppe eine abschließende Phase zur Festigung der gewonnenen mathematischen Erkenntnisse durchführte, kann zum Erreichen der mathematikdidaktischen Ziele nichts gesagt werden. Für eine weitere Durchführung eines solchen Seminars muss noch stärker auf das Erreichen der mathematischen Ziele geachtet werden. So kann eine Veranstaltung z.B. zum „Entdeckendem Mathematikunterricht", die nicht nur auf die Nutzung von Programmiertätigkeiten beim Mathematiklernen ausgerichtet ist, eventuell die Erwartungen der Studierenden eher auf die Mathematik lenken.
Für eine statistisch fundierte Untersuchung der Wirksamkeit des Seminarkonzepts war leider die Teilnehmer/innenzahl auf Seiten der Studierenden zu gering. Auf der Ebene der Schülerinnen und Schüler war die Interventionszeit von zwei mal 90 min zu kurz für messbare Effekte.
Trotz des hohen Zeitaufwands bewerteten die Studierenden ihre Erfahrungen in den Seminaren als positiv und wichtig für ihre spätere Tätigkeit als Lehrkräfte. Eine Studentin formuliert dies in ihrem Projektjournal wie folgt:

> „Zum ersten Mal gezielt Medien in einer dritten Klasse einzusetzen war für mich im Vorfeld sehr spannend. Ich war besonders angespannt, da ich bisher lediglich mit Tablets in einer Grundschulklasse gearbeitet habe. Die Tatsache, dass ich diese Form des Unterrichtens nun in Begleitung von erfahrenen Dozenten erleben konnte, gibt mir für die Zukunft ein sicheres Gefühl beim Einsatz von Medien wie Computer in der Schule. Des Weiteren erachte ich es als Erfolgserlebnis, dass ich nun ein wenig mit dem Verfahren des Programmierens vertraut geworden bin. Nachdem die ersten Programmierungen geglückt sind, war auch das eine positive Erfahrung für mich." (Aussage einer Studentin aus dem Wintersemester 2017/2018)

Literaturverzeichnis

Benton, Laura/Hoyles, Celia/Kalas, Ivan/Noss, Richard (2017). Bridging Primary Programming and Mathematics: some findings of design research in England. In: Digital Experiences in Mathematics Education, 3, pp. 115-138.

Collins, Allan/Brown, John Seely/Holum, Ann (1991). Cognitive apprenticeship: Making thinking visible. In: American educator, 15(3), pp. 6-11.

Dinse de Salas, Simone/Rohlfs, Carsten/Spannagel, Christian (2016). Coaching tea-

chers in using technology. In: Proceedings of EdMedia 2016-World Conference on Educational Media and Technology Vancouver, BC, Canada: Association for the Advancement of Computing in Education (AACE). (S. 927-934). https://www.learntechlib.org/p/173060/ [Zugriff: 31.07.2019].

Franke, Marianne/Reinhold, Simone (2016). Didaktik der Geometrie in der Grundschule (3. Auflage). Berlin, Heidelberg: Springer Spektrum.

Gadanidis, George/Silver, Bronna (2017). Computer Coding in the K–8 Mathematics Curriculum?. What Works. https://oere.oise.utoronto.ca/wp-content/uploads/2018/02/Computer_Coding_K8_en.pdf [Zugriff: 31.07.2019].

GI – Gesellschaft für Informatik (Hrsg.). (2019). Kompetenzen für informatische Bildung im Primarbereich. https://gi.de/fileadmin/GI/Hauptseite/Service/Publikationen/Empfehlungen/GI-Empfehlung_2019_Kompetenzen_fuer_informatische_Bildung_im _Primarbereich__Web_.pdf [Zugriff: 31.07.2019].

Grover, Shuchi/Pea, Roy (2013). Computational Thinking in K–12 A Review of the State of the Field. In: Educational Researcher, 42. Jg., Nr. 1, pp. 38-43.

Kreijns, Karel/Van Acker, Frederik/Vermeulen, Marjan/Van Buuren, Hans (2013). What stimulates teachers to integrate ICT in their pedagogical practices? The use of digital learning materials in education. In: Computers in human behavior, 29(1), pp. 217-225.

Kynigos, Chronis (2007). Half-baked logo microworlds as boundary objects in integrated design. In: Informatics in Education - An International Journal, Vol 6_2, pp. 335-359.

Löthe, Herbert (1985). Logo - eine Sprache zum Kommunizieren über Mathematik. In: Beiträge zum Mathematikunterricht, Franzbecker Verlag, S. 199-202.

Noss, Richard/Healy, Lulu/Hoyles, Celia (1997). The construction of mathematical meanings: Connecting the visual with the symbolic. Educational studies in mathematics, 33(2), pp. 203-233.

Papert, Seymour (1975). Teaching Children Thinking. In: Journal of Structural Language 4, pp. 219-229.

Spannagel, Christian/Bescherer, Christine (2009). Computerbezogene Selbstwirksamkeitserwartung in Lehrveranstaltungen mit Computernutzung. In: Notes on Educational Informatics, Section A: Concepts and Techniques, 5(1). pp. 23-43.

Wachsmuth, Elisabeth/Hense, Jan (2016). Wirkmodelle zur Unterstützung der Evaluation komplexer Hochschulprojekte. In: Qualität in der Wissenschaft. 3+4/2016. Bielefeld: Universitäts-Verlag Webler, S. 80-87.