BAYER, Lukas
Universität Koblenz

Integration von Computational Thinking und Programmierung in den Mathematikunterricht

Bereits Ende der 60er-Jahre proklamierten Feuerzeig & Papert (2011) das Programmieren als eine Möglichkeit in der Mathematik experimentell zu arbeiten und hoben die Bedeutung der Programmiersprache als formalen Sprache zur präzisen Formulierung von Lösungswegen hervor. In der aktuellen Forschung wird Programmieren dabei als eine Möglichkeit gesehen, sogenanntes „Computational Thinking“ (kurz: CT) zu fördern (Lye & Koh, 2014). Dabei sehen Kallia et al. (2021) starke Ähnlichkeiten von CT und mathematischem Denken und ziehen Parallelen von CT im Kontext von Mathematik zur mathematischen Modellierung. Derzeit gibt es jedoch nur wenige Untersuchungen und Ansätze, die die Integration von Programmierung in den Mathematikunterricht über verschiedene Themengebiete und Klassen hinweg betrachtet.

In diesem Posterbeitrag wird daher ein Forschungsprojekt vorgestellt, welches den Einsatz von Programmierung als Werkzeug im Mathematikunterricht entlang eines Schuljahres in einem Educational Design Research Prozess (vgl. McKenney & Reeves, 2018) untersucht. Mehrere Interventionen sollen entwickelt und über den Verlauf eines Schuljahres evaluiert werden. Dabei steht nicht das Lernen des Programmierens sondern der mathematische Unterrichtsgegenstand im Vordergrund.

Bei der Evaluation der Materialien soll der Einfluss auf Selbstwirksamkeitserwartung, Einstellung zum Fach Mathematik, sowie Förderung von CT in einem Mixed-Method-Ansatz näher betrachtet werden. Das Projekt befindet sich aktuell am Anfang, erste Ergebnisse sind 2025 zu erwarten.

Literaturverzeichnis

Feurzeig, W. & Papert, S. (2011). Programming-languages as a conceptual framework for teaching mathematics. *Interactive Learning Environments, 19*(5), 487–501. https://doi.org/10.1080/10494820903520040

Kallia, M., Van Borkulo, S., Drijvers, P., Barendsen, E. & Tolboom, J. (2021). Characterising computational thinking in mathematics education: a literature-informed Delphi study. *Research in Mathematics Education, 23*(2), 159–187. https://doi.org/10.1080/14794802.2020.1852104

Lye, S. Y. & Koh, J. H. L. (2014). Review on teaching and learning of computational thinking through programming: What is next for K-12? *Computers in Human Behavior, 41*, 51–61. https://doi.org/10.1016/j.chb.2014.09.012

McKenney, S. & Reeves, T. C. (2018). *Conducting educational design research*. Routledge. https://doi.org/10.4324/9781315105642

In: P. Ebers, F. Rösken, B. Barzel, A. Büchter, F. Schacht & P. Scherer (Hrsg.),
Beiträge zum Mathematikunterricht 2024.
57. Jahrestagung der Gesellschaft für Didaktik der Mathematik. WTM.
https://doi.org/10.37626/GA9783959872782.0

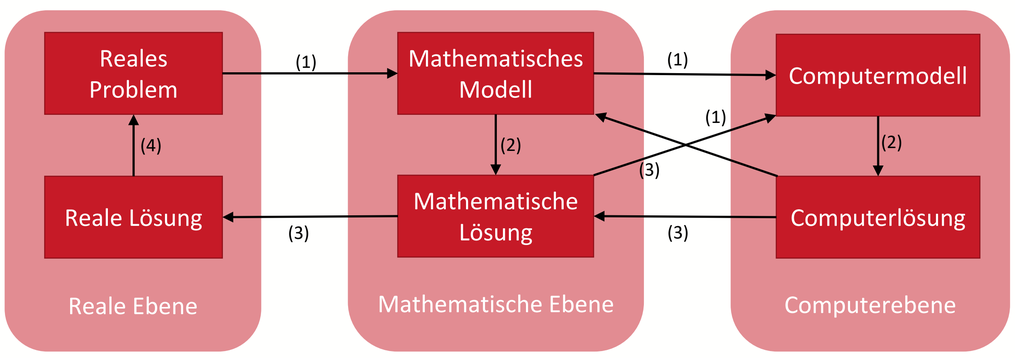
Bayer, Lukas (bayer@uni-koblenz.de)

universität koblenz
Mathematik / Naturwissenschaften

Integration von Computational Thinking und Programmierung in den Mathematikunterricht

Einführung

- Programmieren ermöglicht *mathematisches Experimentieren* und Lösungswege in einer formalen Sprache präzise zu formulieren [2].
- Programmieren fördert Computational Thinking [4], welches Ähnlichkeiten zu mathematischem Denken aufweist (Figur 1)
- Nur wenige Untersuchungen, die sich den Einfluss der Integration von Programmierung in den mathematischen Regelunterricht über verschiedene Themengebieten und Klassen hinweg betrachten (z.B. [1]).



Figur 1: Computational Thinking in mathematischen Kontexten nach [3] bestehend aus dem Prozess des Erstellen eines Modells (1), dem Arbeiten innerhalb des Modells (2), dem Übersetzen von Resultaten in den vorherigen Kontext (3) und dem Verifizieren der Lösung am realen Problem (4)

Forschungsfrage

Wie lässt sich Programmieren als Werkzeug in das bestehende Curriculum der Sekundarstufe I integrieren und welchen Einfluss hat der Werkzeugeinsatz auf Computational Thinking, Selbstwirksamkeitserwartung und Einstellungen zum Fach Mathematik?

Methoden

- Iterativer Educational Design Research Prozess (vgl. [5]) mit zwei größeren Evaluationsphasen (Schuljahr 2024/25 und Schuljahr 2025/26), um Effekt durch Integration in zwei aufeinanderfolgenden Klassenstufen zu evaluieren
- Evaluation über Mixed-Method-Ansatz (qualitative Interviews und Beobachtungen, sowie Kompetenztests zu Computational Thinking und Selbstwirksamkeitserwartung)

Anforderungen & Zielsetzung

- **Programmierung als Werkzeug** begleitend im Regelunterricht
- **„Low Threshhold, High Ceiling“**-Erfahrung

Mögliche Probleme aus Sicht der Lehrkräfte

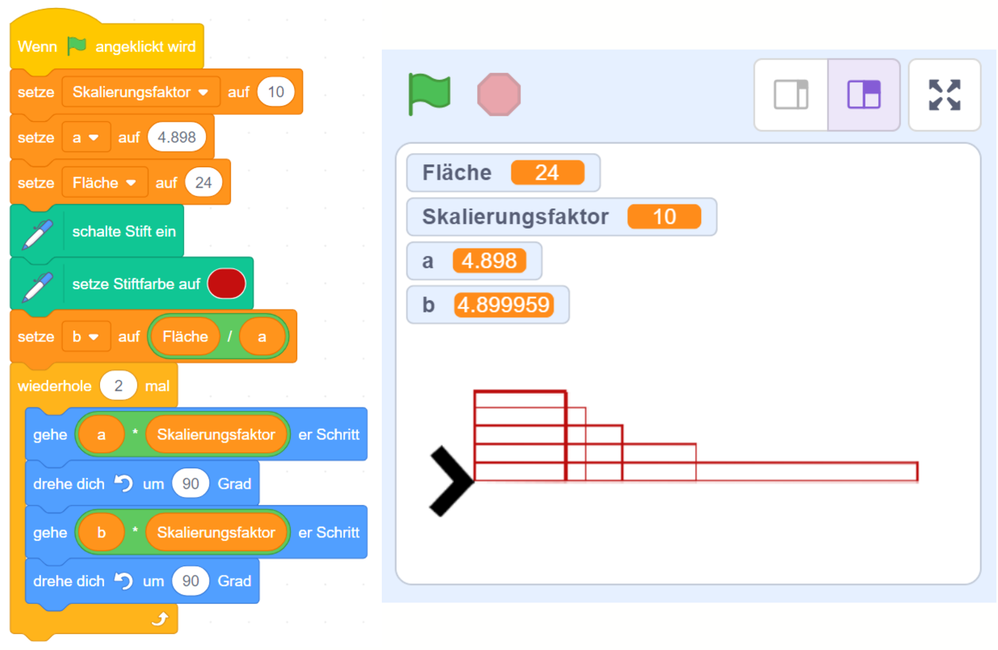
- **Technische Ausstattung** der Schulen und der Familien
- **Eigene Kompetenz** als Lehrkraft
- **Zeit** (Forderung nach Entlastung)

Thematische Ideen aus Sicht der Lehrkräfte

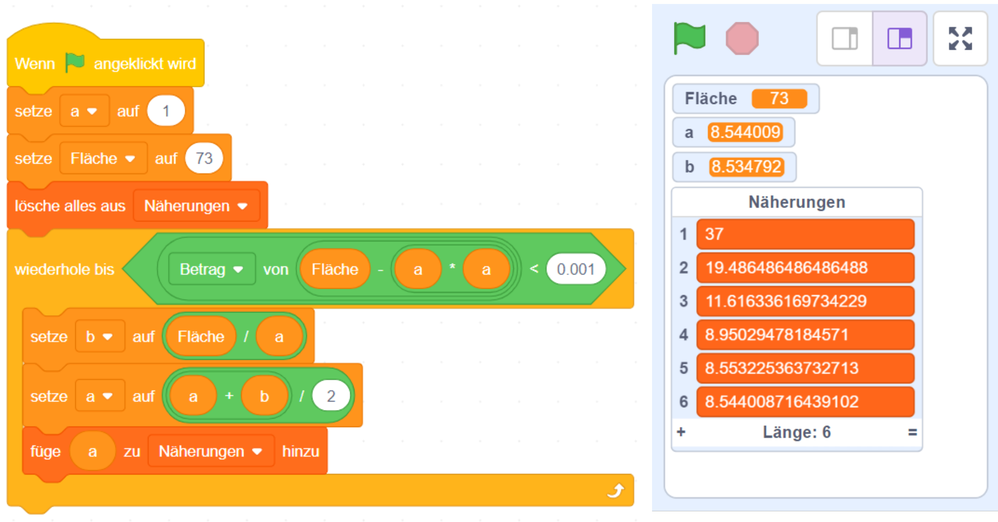
- **Numerische Algorithmen** der Analysis (z.B. Newton-Verfahren)
- **Kombinatorik, Stochastik, Statistik** (z.B. Monte-Carlo-Simulationen)
- **Variablen**- und Parameterkonzept
- Algorithmen der **Bruchrechnung**

Selbstentwickeln von Lernumgebung am Beispiel des Heron-Verfahrens

- Algorithmus als fundamentale Idee der Mathematik [6] und numerische Mathematik bieten eine natürliche Beziehung zur Programmierung. Das Heron-Verfahren bietet sich daher für den Einsatz von Programmierung im Mathematikunterricht an.
- Schülerinnen und Schüler entwickeln ihre eigene Lernumgebung, um die Idee des Algorithmus zu erkunden (Figur 2), bevor sie ihn umsetzen (Figur 3)



Figur 2: Ein Scratch-Programm (links) mit, welches ein Rechteck mit vorgegebenem Flächeninhalt und vorgegebener Seite a erzeugt und Seite b berechnet. Durch Variation von a lässt sich ein Quadrat annähern (rechts). (Quelle: https://scratch.mit.edu)



Figur 3: Das Heron-Verfahren implementiert in Scratch (links) mit Abbruchbedingung anhand der Abweichung der vorgegebenen Fläche von a^2. Der Output für die Wurzel von 73 ist rechts im Bild dargestellt. (Quelle: https://scratch.mit.edu)

Ausblick

- Projekt befindet sich noch am Anfang, erste Resultate sind im Frühjahr 2025 zu erwarten

2023/2	2024/1	2024/2	2025/1	2025/2	2026/1
Anforderungsanalyse					
	(Weiter-)Entwicklung von Material				
		Evaluation im Schuljahr 2024/25		Evaluation im Schuljahr 2025/26	

Kontakt
Lukas Bayer, Universität Koblenz, bayer@uni-koblenz.de

Quellenverzeichnis

[1] Boylan, M., Demack, S., Wolstenholme, C., Reidy, J. & Reaney, S. (2018). *ScratchMaths: evaluation report and executive summary. Project Report.*
[2] Feurzeig, W., Papert, S. A. & Lawler, B. (2011). *Programming-languages as a conceptual framework for teaching mathematics.*
[3] Kallia, M., van Borkulo, S. P., Drijvers, P., Barendsen, E. & Tolboom, J. (2021). *Characterising computational thinking in mathematics education: a literature-informed Delphi study*
[4] Lye, S. Y., Koh, J. H. L. (2014). *Review on teaching and learning of computational thinking through programming: What is next for K-12?*
[5] McKenney, S. & Reeves, T. C. (2012). *Conducting Educational Design Research.*
[6] Schweiger, F. (1992). *Fundamentale Ideen. Eine geistesgeschichtliche Studie zur Mathematikdidaktik.*