Klaus-Tycho FÖRSTER, Hildesheim/Zürich

Die Programmiersprache Scratch in der Sekundarstufe I

Algorithmen sind eine fundamentale Idee für die mathematische und informatische Ausbildung in allen Altersstufen und allen Schulformen. Dieses ist seit langem ein zentraler Schwerpunkt didaktischer Untersuchungen und wird in der Literatur aus unterschiedlichsten Blickwinkeln ausführlich behandelt. Im Alltag des 21. Jahrhunderts beschränkt sich die Relevanz nicht nur auf den Unterricht dieser beiden Fächer: Algorithmen sind „*fächerübergreifend und alltagsrelevant*“ (Schmidt-Thieme 2005).

Programmieren ist hierzu insbesondere hilfreich für die Überprüfung der Korrektheit von Algorithmen und ihrer formalen Präzisierung. Ebenso ist Programmierung förderlich für die Kritikfähigkeit am Ergebnis oder nach Weigand 1989 „*das Wechselspiel zwischen dem Erstellen eines Programms und dem Interpretieren der vom Computer gelieferten Ergebnisse*“. Somit ist Programmieren wichtig für die Überprüfung und Bewertung komplexerer Modellierungen, sowie als Abschluss des Modellierungsverfahrens.

Die GDM forderte daher schon 1981, dass alle Mathematiklehrer die Algorithmen ihres Unterrichts in einer Programmiersprache realisieren können sollen (vergl. Weigand 1989). Aber während nach Oldenburg 2011 in den 80ern noch viele Schulbücher kleine Programme enthielten, kritisiert er, dass *„gegenwärtig im Mathematikunterricht fast nicht programmiert* [wird]“. Nicht nur nach Kortenkamp 2005 ist diese Entwicklung bedauerlich: *„Soll im Mathematikunterricht programmiert werden? Die kurze Antwort, ein uneingeschränktes ‚ja!‘ … ".*

1. Integration von Algorithmen und Programmierung in den MU

Dieser etablierten Forderung steht in Niedersachsen durch curriculare Vorgaben in Mathematik entgegen, dass Algorithmen fast überhaupt nicht und Programmieren eigentlich gar nicht in den Lehrplänen und Schulbüchern enthalten sind, sehr wahrscheinlich auch nicht in der nächsten Iteration des Kerncurriculums Mathematik. Zwar könnte das Fach Informatik hierbei ein guter Partner sein, in Niedersachsen ist Informatikunterricht vor Klasse 10 jedoch leider fast nicht existent. Zumindest in Niedersachsen trägt damit das Fach Mathematik weiterhin die zentrale Verantwortung für die Ausbildung und Vermittlung von algorithmischem Denken!

Daher wird eine Programmiersprache benötigt, die fast unmittelbar einsetzbar und in allen Stufen verwendbar sein muss. Über die letzten Jahrzehnte gab es eine große Anzahl von verschiedenen neuen Ansätzen für die schulische Programmierung. Jedoch erfüllen die etablierten Varianten zumindest

einen zuvor genannten Punkte nicht, sei es nun durch textbasierte Syntax wie in *Java* oder nur punktuelle Einsetzbarkeit wie *Turtle-Grafik* in *Logo* oder *Kara*. Bei einer zeitlichen Reduktion ausgewählter mathematischer Inhalte in den Curricula könnte Raum geschaffen werden, jedoch wird von allen Seiten um jeden Inhaltspunkt gefochten – mit guten Argumenten.

In dieser Situation bietet sich die 2007 veröffentlichte visuelle Programmiersprache *Scratch* an, die schon länger national und international in der Informatikdidaktik etabliert ist und auch in der Mathematikdidaktik genutzt wird, siehe etwa Romeike 2011 oder Wörler 2012. Scratch ist kostenfrei erhältlich für Windows/MacOS/Linux und hat diverse Erweiterungen für komplexere Projekte, wie etwa *BYOB*/*Snap* (snap.berkeley.edu), womit es auch zur Einführung in die Programmierung an Universitäten verwendet wird, siehe dazu etwa Förster 2011. Befehle in Scratch sind selbsterklärend, Syntaxfehler sind nicht möglich, Fehler (gerade in geometrischen Algorithmen) können schnell (ein)gesehen werden und auch Grundschüler ohne Programmiererfahrung haben sofort erste eigene Erfolgserlebnisse. Scratch und seine Erweiterungen sind daher sinnvoll einsetzbar in allen Klassenstufen, von der Grundschule bis zum Abitur – und darüber hinaus.

2. Bewährte Ansätze: Turtle-Geometrie & Konstruktionsbeschreibung

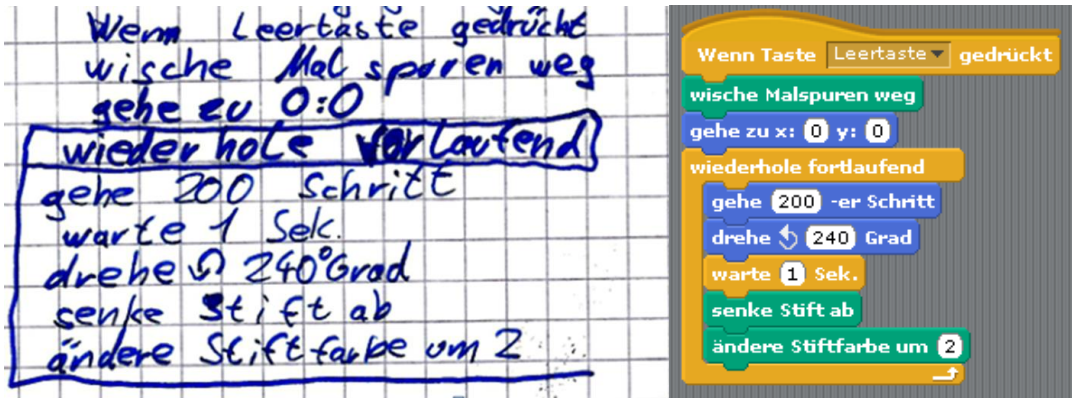
Nach Vollrath 1991 betonte schon Holland 1974, dass „*die klassische Konstruktionsbeschreibung nichts anderes ist als die Angabe eines Algorithmus*". Und Holland führt weiter aus, dass *„erst die Algorithmisierung der Konstruktion dazu zwingt, jeden einzelnen Konstruktionsschritt auf seine Durchführbarkeit zu überprüfen"*. Da eine vertiefte Behandlung von Variablen in Niedersachsen erst in Klassenstufe 7 erfolgt und die Einführung von Variablen in der Informatikdidaktik als einer der schwierigeren Aspekte gilt, bietet es sich naheliegender Weise an, die bewährten Aspekte der Turtle-Grafik aus Logo als begleitenden Einstieg in die mathematische Programmierung mit Scratch zu nutzen. Denn nach Hromkovič 2012 ist mit Logo eine *„hohe Interaktion und gegenseitige Befruchtung mit dem Geometrieunterricht* […] *leicht zu erreichen* […] *und prägt die Entwicklung algorithmischen Denkens"*. Selbst scharfe Kritiker der Turtle-Grafik/Logo-Philosophie räumen ein, dass die Ausbildung des Winkelbegriffs unterstützt wird, etwa bei der Konstruktion von regelmäßigen Vielecken – in Niedersachsen ein Unterrichtsinhalt der Klassenstufe 6.

3. Schulversuch: Erprobung im Unterricht der Klassenstufe 6

Uns interessierte daher vor allem, ob sich die bewährten Konzepte aus der Turtle-Grafik/Logo in Scratch umsetzen lassen und ob die gewünschte Förderung algorithmischen Denkens und formaler Präzision erreicht wird. Da-

zu wurde Scratch begleitend in einer Unterrichtseinheit zu Winkeln und Parkettierungen in einer 6. Klasse im Mathematikunterricht eingesetzt (15 Mädchen, 13 Jungen). Im Rahmen von zwei Doppelstunden wurde zuerst Scratch eingeführt und dann über Konstruktionsbeschreibung von Dreiecken zu Vielecken und abschließend zu Parketten weiterverwendet. Diese Vorgehensweise orientiert sich an einem Mix aus bewährten Unterrichtskonzepten von Logo und den curricularen Vorgaben in Niedersachsen.

Beim Übergang vom Dreieck zum regelmäßigen Vieleck bewegten sich die Schüler auf eine höhere Stufe des algorithmischen Verständnisses, da der Algorithmus nicht nur mehr auf eine spezifische Aufgabe beschränkt war, sondern als Konzept einer selbstdesignten „Black Box“ genutzt werden konnte. Dieser Übergang fiel nicht unbedingt leicht – oder wie es ein Schüler beschrieb, dass es beim ersten Mal schwer war, „*aber wenn man den Bogen raus hatte ging es sehr leicht*“. Als die S. eine Woche später unangekündigt gebeten wurden, ihre Vorgehensweisen noch einmal darzulegen, wählten mehrere als Sprachform eine Art Struktogramm, welche intuitiv auch beim Codieren mit den Blöcken in Scratch vorliegt.



Vergleich von nachträglicher schriftlicher Beschreibung und Scratch-Code

Somit wurde selbständig zur Beschreibung des Vorgehens oft nicht die mathematische Umgangssprache gewählt, sondern die präzise und eindeutige Sprache der Programmierung – der Vorteil der Sprachform für Algorithmen lag anscheinend für die Schüler auf der Hand.

Im Rahmen der Parketterstellung war vor allem die präzise Vorgehensweise und das Konzept der Modularisierung relevant. Ausgehend von der einzelnen Figur konnte zunächst eine Aneinanderreihung und dann daraus ein endliches Parkett erstellt werden. Hierzu waren z.T. Programme mit bis zu 30 Befehlen nötig – bei denen ein einzelner Fehler das Ergebnis zunichtemachten konnte – die von den S. erfreulich korrekt programmiert wurden.

Nach unseren Erfahrungen eignet sich Scratch daher sehr gut als begleitender Einschub in den Geometrieunterricht der Klassenstufe 6. Wir planen

weitere Untersuchungen in höheren Klassenstufen zum begleitenden Einsatz von Scratch. Als Anschluss bieten sich diverse Anknüpfungspunkte an: Etwa Kongruenzsätze, Zufallsexperimente, Bisektion, das Heron-Verfahren oder der euklidische Algorithmus, wobei sich hier schon z.T. die Scratch-Erweiterung BYOB/Snap besser nutzen lässt. Eine weitere Möglichkeit ist die parallele bzw. zeitlich vorgezogene vertiefte Behandlung von Variablen sowohl aus der Sicht des Programmierens als auch aus der „klassischen" Sichtweise im Mathematikunterricht. So führt etwa Serafini 2011 aus, dass Kinder, die das Konzept der Variable über die Programmierung kennenlernen, später im Mathematikunterricht von diesen Vorerfahrungen profitieren könnten. Ein erhöhter Zeitaufwand ist hier sogar sicherlich gerechtfertigt, „*da Variablen Mittel der Verallgemeinerung sind und somit die Grundlage der formalen Betrachtung und Beschreibung von mathematischen Strukturen bilden*" (Reimann 2012).

Literatur

Förster, K.-T. (2011): Neue Möglichkeiten durch die Programmiersprache Scratch: Algorithmen und Programmierung für alle Fächer. In: BzMU 2011, 263-266.

Holland, G. (1974): Die Bedeutung von Konstruktionsaufgaben für den Geometrieunterricht.In: Der Mathematikunterricht, 20(1974), Heft 1, 71-86.

Hromkovič, J. (2012): Einführung in die Programmierung mit LOGO: Lehrbuch für Unterricht und Selbststudium. Wiesbaden: Springer Vieweg.

Kortenkamp, U. (2005): Strukturieren mit Algorithmen. In: Kortenkamp et. al. (Hrsg.): Informatische Ideen im Mathematikunterricht. Hildesheim: Franzbecker, 77-85.

Oldenburg, R. (2011): Mathematische Algorithmen im Unterricht: Mathematik aktiv erleben durch Programmieren. Wiesbaden: Vieweg+Teubner Verlag.

Reimann, K. (2012): Verschiedene Stufen in der historischen Entwicklung der Algebra. In: BzMU 2012, 685-688.

Romeike, R. (2011): Logos Erben – Konstruktionistische Ansätze für Mathematikunterricht und Mathematiklehrerausbildung. In: BzMU 2011, 695-698.

Serafini, G. (2011): Teaching Programming at Primary Schools: Visions, Experiences, and Long-Term Research Prospects. ISSEP 2011, 143-154.

Schmidt-Thieme, B. (2005): Algorithmen – fächerübergreifend und alltagsrelevant? In: Engel, Joachim u. a. (Hrsg.): Strukturieren - Modellieren - Kommunizieren. Leitbilder mathematischer und informatorischer Aktivitäten. Hildesheim, 177-188.

Vollrath, H.-J. (1991): Lokales Ordnen an geometrischen Konstruktionen. In: H. Postel, A. Kirsch, W. Blum (Hrsg.): Mathematik lehren und lernen, Festschrift für Heinz Griesel, Hannover, 217-228.

Weigand, H.-G. (1989): Algorithmen und Computer im Mathematikunterricht. In: BzMU 1989, 386-389.

Wörler, J. (2012): Simulieren (fast) ohne Realitätsbezug - oder: Für die Freiheit der Variation. In: Mathematik lehren, Nr. 174(2012), 41-44.