Heinrich C. Mayr, Martin Pinzger (Hrsg.): INFORMATIK 2016,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2016 1197

FROM SKETCH TO SCRATCH - schrittweise zu „computational thinking“ geführt werden

Alois Bachinger[1] und Anton J. Knierzinger[2]

Abstract: Grundsätzlich ist es nicht notwendig, dass sich Kinder im Kindergarten und in der Grundschule mit Informatik beschäftigen. Was wir im Workshop aber zeigen wollen ist, dass es sehr sinnvoll ist, sich mit Strukturen des allgemeinen Problemlösens und mit technischen Aspekten der kindlichen Spielwelt und des Lernalltags zu beschäftigen. Die Gründe dafür liegen nicht nur in der Omnipräsenz von digitaler Technik, der auch Kinder in diesem Alter ausgesetzt sind. Vielmehr vergrößert die Entwicklung von algorithmischen Denken die persönlichen und methodischen Kompetenzen. Nicht zuletzt aber ermöglicht die Beschäftigung mit Codieren die Umsetzung didaktischer Modelle, die kreativen, und motivierenden Unterricht ermöglichen. Der Weg von der Problemanalyse (Sketch) bis zur formalen Beschreibung eines Algorithmus (in Scratch) kann für Kinder lustig sein und ihre Kreativität anregen.

Ausgehend von der Bedeutung die Algorithmen in unserer Gesellschaft erreicht haben, soll unser 7-Stufen Modell zum Unterricht von Coding im Grundschulalter präsentiert und die Werkzeuge Faktoren, die Kinder in diesem Alter für dieses Thema begeistern können, gezeigt und diskutiert.

Keywords: Kindergarten, Grundschule, Codieren, algorithmisches Denken, Didaktik

1 Einleitung

Begeisterung für die Beschäftigung mit einem Inhalt oder einer Methode ist ein Grundelement erfolgreichen Lernens. Wir verfolgen seit 1980 die Integration von Informationstechnik in der Schule. Die Frage, warum wir in Deutschland und Österreich zu wenig Nachwuchs in technischen Berufen auf allen Ebenen haben, hängt sehr stark mit der Einstellung der Schüler zu Technik und ihrer Begeisterung dafür zusammen. Die Entwicklung von „Computational Thinking“ ist ein Weg, Schüler zu begeistern.

2 Was ist „computational thinking“?

Die Bezeichnungen für arbeiten mit Computern in der Schule hat sich im Lauf der Zeit immer wieder verändert. Ursprünglich war es Informatik, späten dann Informatische Grundbildung, usw. Für unsere Arbeit passt wohl am besten: computational thinking. Diesen Begriff beschreibt Jeanette M. Wing [Wi08] als: „… it represents a universally

[1] Pädagogische Hochschule Linz, Salesianumweg 3, 4020 Linz, alois.bachinger@ph-linz.at
[2] Pädagogische Hochschule Linz, Figulystr. 1, 4020 Linz, anton.knierzinger@ph-linz.at

applicable attitude and skill set everyone, not just computer scientists, would be eager to learn and use“. Es geht also sowohl um Einstellungen als auch um Fertigkeiten und zwar solche, die sich an die Allgemeinheit und nicht nur an Spezialisten wenden. Computational thinking heißt viel mehr als Computer programmieren zu können. Es braucht dazu Denken auf verschiedenen Ebenen der Abstraktion. Dazu gehören:

- Konzepte zu entwickeln statt programmieren zu lernen,
- kreative Prozesse statt erlernte Fähigkeiten anwenden,
- in der Art von Menschen nicht in der von Computern zu denken,
- dabei mathematische und technische Kompetenzen zu entwickeln und
- Ideen, nicht Werkzeuge in den Vordergrund zu stellen.

3 Didaktische Begründung

Aus unseren Erfahrungen und Untersuchungen, aber auch aus Literaturhinweisen, kann abgeleitet werden, dass computational thinking sowohl die Methodenkompetenzen (Umgang mit IT und digitalen Medien, Sprachkompetenz) wie auch die persönlichen Kompetenzen (Kommunikationsfähigkeit und Problemlösekompetenz, Selbstbewusstsein und Selbsteinschätzung) erweitert.

4 Faktoren für Begeisterung

Resnicks 4P`s [Re16] gibt folgende Faktoren für kreatives Lernen an:

- **projects**: Das Arbeiten soll zielorientiert, fächerübergreifend, praxisorientiert und vor allem kreativ sein.
- **peers:** Lernen floriert als soziale Aktivität, wenn Leute Ideen austauschen, an gemeinsamen Zielen arbeiten und ihre Ergebnisse zusammenlegen.
- **passion:** Wer an Projekten arbeitet, die ihm sinnvoll erscheinen, Spass machen, und herausfordernd sind, der arbeitet länger, härter und effektiver.
- **play:** Lernen soll Spass machen, experimentieren erlauben, zu seinen Grenzen führen und immer wieder probieren erlauben.

In den Aufgabenstellungen sollen sich Situationen mit spielerischem Charakter mit Bezügen zu Alltag, Technik und Phantasiewelten mischen und zu ergänzen.

5 7 Stufen im Programmierunterricht für Anfänger

Dieses Konzept stellt 7 Stufen im Unterricht von Problemlösen und Codieren dar. Von Alltagssituationen (Realität) aus wird über dingliche Darstellung und Mischformen mit steigender Abstraktion eine rein maschinelle Repräsentation (Virtualität) von Algorithmen erreicht. Die Ausprägung der einzelnen Stufen muss an Vorwissen, Alter und Lernziele angepasst werden, ist aber zumindest in den Grundelementen auch im Kindergartenalter möglich. Das Konzept stellt so einen Leitfaden für Lehrende dar. Verwendet werden vor allem Algorithmen, die Bewegung von Objekten codieren. In den Aufgabenstellungen sollen sich Situationen mit spielerischem Charakter mit Bezügen zu Alltag, Technik und Phantasiewelten mischen.

Die, in den folgenden sieben Stufen vorgeschlagene, Vorgangsweise steigert das notwendige Abstraktionsniveau stufenweise. Die Bandbreite der Einsatzszenarien reicht von 5 Jahren unter Berücksichtigung der entwicklungspsychologischen Veränderungen bis ins Erwachsenenalter. Bei jüngeren Kindern wird man sich stärker auf die ersten Stufen konzentrieren und nicht alle Stufen durchlaufen. Bei Erwachsenen, etwa in der Lehrerausbildung, genügt es die ersten Stufen kurz zu streifen.

real:
Algorithmen werden im Alltag gefunden. Sie werden durch reale Dinge dargestellt (z.B. Bewegungen von Menschen, Objekten)

blended:
Algorithmen werden sowohl real wie auch softwaremäßig repräsentiert

virtuell:
Der Algorithmus, sein Ablauf und seine Wirkung laufen ausschließlich am Computer ab.

Zur didaktischen Umsetzung werden sieben Stufen vorgeschlagen:

Stufe 1: Algorithmen im Alltag.

Algorithmen sind Ketten von Anweisungen, Algorithmen entdecken, in Sprache beschreiben lernen, einzelne Schritte definieren können.

Stufe 2: Algorithmen als Anweisungen.

Lernende geben sich gegenseitig einfache Befehle zur Bewegungssteuerung, Bewegungen werden als Befehlsketten dokumentiert, Darstellung von Algorithmen als Symbole auf Würfeln und Befehlskarten.

Stufe 3: Steuerung eines tastenprogrammierbaren Roboters.

Als Roboter wird BeeBot, eine kleine, mit Tasten programmierbare „Biene“ verwendet. Die Eingabe des Algorithmus zu seiner Bewegungssteuerung erfolgt noch direkt am Gerät, zuerst step-by-step dann als Befehlsfolge. Schwierigkeitsstufen:

1: Freies Agieren in der Ebene ohne mathematisches Bezugssystem
2: Einführung eines Rasters (Koordinatensystems)

Stufe 4: Algorithmen in maschineller Darstellung.

Erste Umsetzung in maschinelle Darstellung erfolgt mit Darstellung der Bewegung des Roboters (Biene) am PC durch die Beebot-App, RunMarco-App und CargoBot-App.

Stufe 5: Algorithmen in einer einfachen Programmiersprache.

Verwendet wird vorerst Scratch junior. Als Aufgabenstellungen werden Spielsituationen und das Erzählen von Geschichten vorgeschlagen. Die Schüler werden animiert, selbst neue Problemstellungen zu erfinden und diese zu lösen.

Stufe 6: Konstruktion und Programmierung von Robotern.

Ausgehend von einem über Computersteuerung programmierbaren, fertigen Roboter (z.B. Auto), wird zu Roboterbaukästen (z.B. Cubelets) übergegangen. Einführung von Sensoren, Aktoren und Controls. Die Programmierung erfolgt über vorhanden Tools. Vorstellung von einfachen Programmstrukturen (Alternative, Iteration, Module)

Stufe 7: Programmieren in Scratch.

Scratch ist eine vollwertige Programmiersprache, die Einführung in das Programmieren in allen Altersstufen bis hin auf Universitätsniveau und auf allen Abstraktionsstufen ermöglicht. Mit ihr können auch die gebauten Roboter gesteuert werden.

6 Erfahrungsberichte

Dieses Konzept wurde bisher in zahlreichen, unterschiedlichen Settings erfolgreich durchgeführt. Erwähnt seien: KinderUNI, Talente Oberösterreich, Lange Nacht der Forschung, Studiengänge der Grundschullehrerausbildung, Lehrerfortbildungsveranstaltungen für Primar- und Sekundarstufe, Pädagogische Veranstaltungen (Fachmessen Interpädagogica und Bildung online), Fachtagungen und auch in der Erwachsenenbildung (z.B. Sprachkompetenzentwicklung von Asylwerbern).

Literaturverzeichnis

[KG15] Knierzinger, A.; Gradinarova, B.: Learning to Code, Coding to Learn - The world of algorithms in higher education. In: (Elmas, M.,et.al., Eds.): ICQH Proceedings Book, University of Sakarya, icqh.net, Stand 30.12.2015

[Re16] Resnick, M.: Give P's a Chance: Projects, Peers, Passion, Play, http://web.media.mit.edu/~mres/papers/constructionism-2014.pdf, Stand: 13.5.2016

[Wi08] Wing,J., Computational Thinking and thinking about computing, http://rsta.royalsocietypublishing.org/content/366/1881/3717, Stand: 13.5.16