Gerhard Brandhofer

Coding und Robotik im Unterricht

Summary: *Dieser Artikel setzt sich mit der aktuellen Situation zu Coding und Robotik im Schulunterricht in Österreich auseinander. Es soll die Frage beantwortet werden, wie die Behandlung von Coding und Robotik im Unterricht zu legitimieren ist, um schließlich einige Lernumgebungen für den Einstieg in diese Themen vorzustellen und Verbindungen zu Game Based Learning, Making und Computational Thinking darzustellen. Im Rahmen dieser Vorstellung werden Lernumgebungen für die Altersstufen vom Kindergartenalter bis zur Sekundarstufe II berücksichtigt. Eine Kategorisierung entsprechend informatischer Kriterien soll die Mächtigkeit der einzelnen Programme verdeutlichen und die Vergleichbarkeit ermöglichen.*

Einleitung

In der 2016 erstellten Dagstuhl-Erklärung wurden die Anforderungen an Bildungssysteme in einer digital vernetzten Welt festgehalten. Es sei Aufgabe aller Fächer, fachliche Bezüge zur digitalen Bildung herzustellen, daneben müsse aber auch ein eigenständiger Lernbereich eingerichtet werden (Gesellschaft für Informatik, 2016, S. 1). Diese Ansicht deckt sich mit meiner Forderung für ein garantiertes Zeitgefäß im Unterricht der Sekundarstufe für Medienbildung und Informatik (Brandhofer, 2014). Mit dem Ziel, dass Schüler/innen die Fähigkeit erlangen sollen, mit digitalen Systemen selbstbestimmt umzugehen, wird in der Dagstuhl-Erklärung eine umfassende Betrachtungsweise zugrunde gelegt. Sie beinhaltet die technologische, die gesellschaftlich-kulturelle und die anwendungsbezogene Perspektive zu digitalen Medien: „Dies erfordert, sie zu verstehen, zu erklären, im Hinblick auf Wechselwirkungen mit dem Individuum und der Gesellschaft zu bewerten sowie ihre Einflussmöglichkeiten zu sehen und nicht nur ihre Nutzungsmöglichkeiten zu kennen“ (Gesellschaft für Informatik, 2016, S. 2).

Im österreichischen Bildungswesen ist Programmieren als Teilbereich der Informatik vergleichsweise schlecht verbreitet. Ein flächendeckender, verpflichtender Lehrinhalt ist im Pflichtschulwesen bisher nicht gegeben, wenn, dann werden diese Inhalte im Rahmen von schulautonomen Schwerpunktsetzungen berücksichtigt. Eine Dokumentenanalyse (Curricula und Schwerpunktbeschreibungen) und darauf bezogene ergänzende Interviews mit Schulleitern und Schulleiterinnen sowie Informatiklehrenden von acht Schwerpunktschulen in Niederösterreich ergab, dass etwa 88 % der Inhalte dem Bereich Computer Literacy zuordenbar sind und 12 % für medienpädagogische Themen reserviert sind. Coding kommt in keinem der gesichteten Papiere vor, wird aber zumindest an drei der genannten Schulen dennoch praktiziert (Scratch, Lego Mindstorms). Computational Thinking ist explizit weder Inhalt der Papiere noch des Unterrichts. Zu erwähnen ist, dass die vier hier verwendeten Kategorien weder trennscharf noch umfassend sind.

Zur Legitimation

Warum sollte Programmieren Teil der Curricula der Primarstufe und Sekundarstufe sein? Zur Legitimation des Lernens mit digitalen Medien und über digitale Medien wurden an anderer Stelle mehrere Legitimationsansätze ausführlich behandelt: „Methodenvielfalt, Wechselwirkung, Arbeitswelt, Lebenswelt und Handlungsreflexion sind fünf Ansätze zur Legitimierung des Einsatzes digitaler Medien im Unterricht“ (Brandhofer, 2015a, S. 83). Es sollte unbestritten sein, dass Lernen mit digitalen Medien und über digitale Medien wesentlicher Bestandteil zeitgemäßen Unterrichts ist: „Wenn digitale Medien zunehmend unser Denken und Handeln prägen, so wird es auch wichtiger, dass Kinder und Jugendliche Medien nicht nur effizient, sondern auch kritisch und mündig nutzen“ (Döbeli Honegger, 2016, S. 80). Coding als Teilbereich der Nutzung des Digitalen und der Auseinandersetzung mit dem Digitalen in der Schule bezieht sich insbesondere auf das Arbeitsweltargument als auch auf das Lebensweltargument.

Zum Arbeitsweltargument: die Informationstechnologie hat für die Wirtschaft enorme Bedeutung und bringt der Jugend gleichzeitig hervorragende Berufsaussichten. Der deutsche Bundesverband Informationswirtschaft, Telekommunikation und Neue Medien e.V. hat für 2010 für den Informations- und Telekommunikationsbereich ein Volumen von 150 Milliarden Euro veranschlagt. Das stellt auch die Schulbildung vor neue Herausforderungen, Faktenwissen ist heute sehr vergänglich, andere Kompetenzen werden von den Schulabgängern und Schulabgängerinnen verlangt. Bernhard Löwenstein und Monika Di Angelo sehen hier Handlungsbedarf an den Schulen: „Leider wird dieser Tatsache aber nicht immer Tribut gezollt und an vielen Schulen nach wie vor nur kurzfristiges Anwenderwissen anstatt langfristiges Konzeptwissen vermittelt“ (Löwenstein & Di Angelo, 2012, S. 293). Das Konzeptwissen der praktischen Informatik umfasst aber das algorithmische Denken, Datenstrukturen, Programmieren und Softwaretechnik (Humbert, 2006, S. 10). Bedenken sollte man dennoch, dass Informatikunterricht nicht ausschließlich ein Programmierkurs sein sollte, Themen wie Datenbanken, Security, Kryptologie, Modellbildung sind beispielsweise ebenso Bestandteil informatischer Bildung.

Berücksichtigen sollte man zudem, dass die bei der Auseinandersetzung mit Informatik erworbenen Kompetenzen nicht ausschließlich informatische Kompetenzen sind: „Die Informatik bietet zahlreiche Werkzeuge, um Dinge im virtuellen Raum mit Computern zu simulieren oder im realen Raum zum Beispiel mit Robotern zu konstruieren. Dadurch kann viel über Mathematik, Geometrie, Physik oder Volkwirtschaft gelernt werden“ (Döbeli Honegger, 2016, S. 92).

Zum Lebensweltargument: wenn man sich für die Implementierung des Themas Coding in den Unterricht einsetzt, ist man dem Vorwurf ausgesetzt, dass Inhalte für eine kleine Minderheit in der künftigen Arbeitswelt propagiert werden. Mittermeir et al. stellen polemisch fest: „Wer von unseren Schülern und Schülerinnen wird schon den Beruf des Programmierers / der Programmiererin ergreifen? Außerdem: Programmieren ist schwierig“ (Mittermeir, 2010, S. 57). Allerdings ist unsere Lebenswelt und nicht nur unsere Arbeitswelt durch digitale Medien geprägt (ausführlich: International Telecommunication Union, 2016). Durch die Leitmedientransformation (Brandhofer, 2015b, S. 716; Erdmann, 2011) werden wir vor neue Herausforderungen gestellt. Die Zeichen deuten darauf hin, dass sich das Erscheinungsbild der digitalen Medien zwar laufend wandeln wird, eine Rückkehr des Leitmediums Buch aber nicht stattfinden wird, wir erfahren gerade eine

Leitmedientransformation, die unumkehrbar sein wird. Transformation bedeutet in diesem Zusammenhang die Änderung von allem und von Grund auf, ein neues Emergenzniveau bildet sich heraus, diese Veränderung wirkt sich auch auf gekoppelte Systeme aus (Erdmann, 2011). In dieser digitalisierten Welt ist die Fähigkeit zu algorithmischen Denken und die Übung darin durch Coding und Robotik von ausschlaggebendem Vorteil für das Individuum, Modellbildung und Simulation sind Grundlage für die Entscheidungsfindung in vielen Bereichen der Industrie und damit Komponenten der Allgemeinbildung (Herper & Stahl, 2013, S. 139)

Einige Beispiele zu Coding und Robotik im Unterricht

Wenn man die Forderung zur Berücksichtigung von Programmieren und Robotik im Unterricht vorbringt, so trifft man nicht selten auf Gesprächspartner/innen die Logo und Basic kennen, mit denen sie eventuell während ihrer Schulzeit Kontakt hatten. Mittlerweile gibt es aber eine Vielzahl an erziehungsorientierten Programmiersprachen und Robotern, die algorithmische Fähigkeiten fördern können, kindgerecht sind, mit denen man Elemente des Game Based Learning aufgreifen kann und die den Kindern Spaß machen. In der Folge sollen einige dieser Lernumgebungen kurz vorgestellt werden und ein Stufenplan zum Einsatz in der Schule entwickelt werden.

Um den Überblick bei Lernumgebungen für den Einstieg ins Programmeiren zu behalten, ist eine Kategorisierung zweifelsohne hilfreich. Eine ganz hervorragende Übersicht von derartigen Lernumgebungen hat Michael Hielscher mit Hilfe einer Kategorientafel unternommen (Hielscher, 2016). Jede Lernumgebung wird anhand von sieben Kriterien kategorisiert. Das Repräsentationskriterium beschreibt, ob die Umgebung eine enaktive und/oder eine simulierte ist. Die Interaktivität des Programms (keine, interaktiv, simuliert) wird im Interaktivitätskriterium bestimmt. Das Koordinationskriterium beschreibt, ob ein einzelnes Objekt programmiert wird, oder mehrere Objekte Teil des Programms sein können. Ob die Programmierung eher imperativ oder ereignisbasiert erfolgt, beschreibt das Ausführungskriterium, das Notationskriterium die Darstellung des Programms (keine, Symbole, Textblöcke, geschriebener Text). Viele der Lernumgebungen sind universelle Werkzeuge, Lehrende können eigene Aufgabenstellungen vorgeben, Schüler/innen eigene Ideen verwirklichen. Bei manchen Lernumgebungen ist das nicht möglich, es gibt ausschließlich vorgegebene Aufgaben, diese beiden Varianten sind im Didaktisierungskriterium festgehalten. Schließlich hat Hielscher auch das Mächtigkeitskriterium der jeweiligen Lernumgebung angeführt, es wird bei jedem Programm angegeben, welches der sieben vorab definierten Konzepte des Programmierens (Sequenz, bedingte Anweisung, Wiederholung, Prozeduren, Rekursion, Variablen, Datentypen, Objektorientierung) durch die Lernumgebung abgebildet werden können (Hielscher, 2016). Dieses Mächtigkeitskriterium soll bei der folgenden Darstellung eine Matrize für den Stufenplan sein.

BeeBots

Die BeeBots sind kleine Bodenroboter. Sie werden mit wenigen Funktionstasten direkt am Gerät gesteuert. Die Programmierung erfolgt mit den Tasten vorwärts, rückwärts, links drehen, rechts drehen, Pause und löschen. Erst mit Drücken der Taste Go wird die Programmsequenz gestartet. Das bedeutet, dass man vorab die Schritte zum Ziel planen muss. Mit den **BeeBots** kann man sich spielerisch dem Programmieren annähern - und das schon in Kindergarten und Volksschule. BeeBots sind simpel in der Nutzung und machen Freude.

Ziel wäre, die nötigen Sequenzen hin zu einem definierten Ziel mit Hilfe von Buchstaben aufzuschreiben:

x v v l v r v go

Dadurch, dass die Schritte mit jeweils 15 cm festgelegt sind, ist es möglich, neben vorgefertigten Matten auch selbst erstellte Unterlagen zu verwenden. Das bedeutet, dass Kinder hier auch aktiv in die Materialiengestaltung eingebunden werden können. Bei der ersten Verwendung der BeeBots überrascht die Variabilität dieser vermeintlich simplen Roboter. So sind zahlreiche Variationen möglich: es dürfen bestimmte Tasten nicht verwendet werden (z.B.: keine Vorwärts-Taste), Parallelslalom mit zwei BeeBots, Verfolgungsrennen, zwei BeeBots müssen zu je einem Ziel gelangen, ohne dass sich die Wege kreuzen, die Verwendung großer Matten für mehrere BeeBots, physische Hindernisse auf bestimmten Feldern, Arbeiten mit Mehrfachlösungen (Pause-Taste bei der ersten Lösung, dann weiter zur nächsten Lösung) oder: die BeeBots werden verkleidet.

Die BeeBots eignen sich entsprechend Hielschers Kategorisierung für Kindergarten und Primarstufe, aus Sicht der Informatik kann das Notieren von Sequenzen mit ihnen erlernt werden (Hielscher, 2016). Eine Abwandlung der BeeBots sind die BlueBots, diese können sich via Bluetooth mit einem Setzkasten verbinden, in diesem werden die einzelnen Befehle sequentiell abgelegt.

Ozobots

Von den Ozobots sind zurzeit drei Varianten verfügbar: *basic, bit und evo.* In der Folge möchte ich über die Möglichkeiten und Grenzen der Version *bit* berichten. Der Ozobot bit ist ein kleiner Roboter mit fünf Farbsensoren an der Unterseite und Akku, der ab der Primarstufe im Unterricht eingesetzt werden kann. Er kann auf Tablets oder einem Spielbrett Linien folgen und Farben erkennen, man kann ihn aber auch Programmieren. Der Ozobot bit vereint somit analoges und digitales Spielen und Lernen. Die Macher/innen des Ozobots versuchen sich an das *low floor - wide walls - high ceiling* Prinzip zu halten, welches auch ursprünglich als Maxime bei der Entwicklung von Scratch ausgegeben wurde (Resnick et al., 2009, S. 63). Der Einstieg ist einfach, es gibt verschiedene Zugangsweisen, es können aber auch komplexe Programme erstellt werden.

Bei der Verwendung des Ozobot bit können vier Zugänge unterschieden werden. Die erste Option in der Verwendung ist jene des Linienfolgeroboters. Der Ozobot bit kann einem selbst gezeichneten Weg folgen, er übernimmt die Farbe des Pfades für die Aktivierung seiner LEDs und kann so auf einer selbstgeplanten Rennbahn seine Runden ziehen.

In einer weiteren, etwas anspruchsvolleren Variante, kann der Ozobot mit Farbcodes gesteuert werden. Für das Erstellen der Codes gibt es Befehlsvorlagen. Mit vier Farben können alle verfügbaren Codes aufgezeichnet werden. Diese Funktion kann auch am Tablet mit einer eigenen App gesteuert werden. Wer keine eigenen Ideen umsetzen möchte, kann auf Vorlagen in der App zugreifen. Der Ozobot bit kann an die 1000 digitale Codes und Befehle verarbeiten, Geschwindigkeit, die Farbe der LEDs und vieles mehr ist kontrollierbar.

Die Förderung algorithmischen Denkens wird aber insbesondere durch die Nutzung der zugehörigen Programmierumgebung – Ozoblockly – möglich. Diese überrascht durch ihren modularen Aufbau und der unkonventionellen Methode bei der Übertragung des Programms auf den Ozobot bit. Die Programmieroberfläche erinnert in ihrer Struktur an Scratch (siehe später), sie kann in fünf Komplexitätsebenen benutzt werden. Dadurch ist es möglich, sowohl für Kindergartenkinder aber auch für Oberstufenschüler/innen eine passende Programmieroberfläche bereit zu stellen. Für Letztgenannte ist auch eine JavaScript-Vorschau implementiert. Die Übertragung des fertiggestellten Programms auf den Ozobot bit erfolgt

über eine definierte Fläche am Bildschirm. Der Ozobot bit wird an diese Stelle gehalten und empfängt über Farbsignale den Code.

Mit dem Ozobot bit können 6 der 8 aufgelisteten informatischen Konzepte abgebildet werden (Hielscher, 2016). Zusätzlich haben sie durch die digital-analoge Dualität im Bedienkonzept einen sehr reizvollen Zugang und eignen sich hervorragend für den Einsatz im Unterricht im Anschluss an die BeeBots und vor (oder auch nach) einer Einführung in Scratch in der Primarstufe und der Sekundarstufe I.

Auch bei den Ozobots verblüfft die Variabilität, einige Ideen zur analogen Verwendung seien hier angeführt: Schüler/innen sollen in vorgegebene Lücken Codes einfügen, die den Ozobot an einer bestimmten Stelle zum Stillstand bringen. Code auslesen – Schüler/innen beschreiben wie der Bot sich verhalten wird. Schüler/innen lösen mit Hilfe des Ozobot bit ein Labyrinth. Eine Rennbahn wird so gestaltet, dass sie innerhalb einer bestimmten Zeit durchfahren wird.

Neben den genannten Vorteilen zeigen sich bei der Arbeit mit dem Ozobot bit aber auch einige Herausforderungen. Beim Zeichnen der Linien ist eine gewisse Genauigkeit erforderlich, diese dürfen weder zu dünn noch zu dick sein. Damit die Farbcodes erkannt werden, müssen diese auch exakt gezeichnet werden, Kurven sollten nicht zu eng entworfen werden.

Scratch

Scratch ist eine visuelle Programmiersprache und wird oft als erziehungsorientierte Programmiersprache kategorisiert. Die Programmierumgebung wurde von der Lifelong Kindergarten Group am MIT entwickelt. In Österreich ist Scratch mittlerweile gut verbreitet und zumindest an zahlreichen Schwerpunktschulen in Österreich in Verwendung. Die Möglichkeiten von Scratch wurden von mir im Jahr 2008 im Rahmen der eLearning Didaktik Fachtagung in Wien vorgestellt, gemeinsam mit der OCG wurden Initiativen gesetzt und Fortbildungen angeboten, mit der Intention, die Programmierumgebung bei Lehrenden bekannter zu machen (Bundesministerium für Unterricht, Kunst und Kultur, 2008). Viele neuere erziehungsorientierte Programmiersprachen nehmen sich das Design von Scratch als Vorbild (Snap!, Ozoblockly, Touchdevelop).

Das bereits zuvor erwähnte *low floor - wide walls - high ceiling* Prinzip war Ausgangspunkt bei den Überlegungen zum Programmdesign. Resnick et al. erkannten bald, dass sich im Laufe der Zeit neue Herausforderungen für kindgerechte Programmierumgebungen ergeben haben: „To achieve these goals, we established three core design principles for Scratch: Make it more tinkerable, more meaningful, and more social than other programming environments“ (Resnick et al., 2009, S. 63).

Scratch ist so einfach in der Bedienung, dass bereits Volksschulkinder damit erste Erfahrungen im Programmieren machen können: „Scripts are created by snapping blocks together, much in the same way that Lego blocks are snapped together to create all sorts of unique creations“ (Ford, 2009, S. 4). Scratch läuft ab Version 2 im Browser, eine Installation ist nicht nötig. Die einzelnen Befehle werden per Drag & Drop zu Sequenzen verbunden, eine kindgerechte Bedienung steht im Vordergrund. Die Projekte werden online gespeichert und können veröffentlicht werden, so sind erste Ergebnisse schnell online verfügbar, aber auch ein lokales Speichern ist möglich. Mit einer großen Vielfalt an Zugangsmöglichkeiten und der Multimedialität lädt Scratch zum spielerischen Forschen ein und schlägt damit auch die Brücke zum Game Based Learning (Gabriel, 2016, S. 27).

Den Scratcheditor kann man ohne Anmeldung ausprobieren. Zusätzlich ist es möglich, einen Blick in den Code veröffentlichter Projekte zu werfen. Auch auf diese Weise kann man sich Ideen und Anregungen holen. Für große Projekte ist diese Plattform weniger geeignet, der primäre Zweck ist, den Einstieg ins Programmieren so einfach und kindgerecht wie möglich zu machen. Ein rechtzeitiger Umstieg auf andere – mächtigere – Programmierumgebungen sollte daher eingeplant werden, im Anschluss bietet sich etwa Python an. Der Scratcheditor ist für die Arbeit am Desktop optimiert, für Tablets eignet sich die weniger mächtige App ScratchJr. Für Scratch sind mittlerweile zahlreiche Anleitungen und Hilfestellungen verfügbar (Brandhofer, 2009, 2016, 2017). Aus dem informatischen Blickwinkel betrachtet hat Scratch einen besonders hohen Reiz für den Einführungsunterricht ins Programmieren, weil alle acht von Hielscher ausgewählten Konzepte mit Scratch bedient werden können: Sequenz, bedingte Anweisung, Wiederholung, Prozeduren, Rekursion, Variablen, Datentypen und Objektorientierung (Hielscher, 2016).

Lehramtsstudierende der Primarstufe zeigen sich besonders von der Multimedialität von Scratch angetan, so kann ein Einstieg ins Programmieren beispielsweise über Bildnerische Erziehung (Schmetterling mit mehreren Kostümen zieht seine Kreise) oder auch Musik (Tastatur als Schlagzeug) erfolgen. In Kombination mit Picoboard oder MakeyMakey können viele weitere Ideen umgesetzt werden. Studierenden der Informatik gefällt wiederum die Professionalität in der Umsetzung und die Unterstützung vieler informatischer Kategorien und die gute Skalierbarkeit. Tatsächlich wird Scratch sowohl in der Volksschule, wie in mancher Neuen Mittelschule oder HTL aber auch an Universitäten in Österreich und weltweit für eine Einführung ins Programmieren eingesetzt (Antonitsch & Hanisch, 2014; Badura, 2012; Brandhofer, 2008; Modrow, 2011, S. 6; Universität Wien, 2016).

Weitere Programmierumgebungen, Making

Im Anschluss an die Arbeit mit Scratch bieten sich einige Optionen zur Vertiefung der Kenntnisse an. Zum einen ist ein Wechsel zu Snap! möglich. Diese Programmierumgebung basiert auf Scratch, wurde ursprünglich entwickelt, um das Erstellen eigener Blöcke zu ermöglichen, beinhaltet First-Class Functions, eine prozedurale, objektorientierte, funktionale Programmierung ist durchführbar. Snap! verknüpft die Usability von Scratch mit der Mächtigkeit von Universalprogrammiersprachen wie Scheme (eine der wesentlichen Lisp Dialekte, Steele, 1990). Eine weitere Option ist der mutigere und größere Umstieg zu Python. Hiermit eröffnen sich für die Lernenden neue Welten, sie bekommen nicht mehr vom Gleichen vorgesetzt. Möchte man in die App-Programmierung einsteigen, bietet sich der MIT App Inventor an.

Weiters ist aber auch die Kombination von Coding mit Making Elementen sinnvoll. Hier bieten sich in erster Linie die MakeyMakeys gemeinsam mit Scratch an. Wie der Name schon sagt, steht beim MakeyMakey das Selbermachen im Vordergrund. Der MakeyMakey ist aus technischer Sicht ein Tastaturersatz, an Stelle der Standardtastatur können die einzelnen Tasten dann Bananen, Äpfel, Plastilin, Alufolie oder Personen sein. Die Platine ist ein vereinfachter Arduino, Treiber sind nicht nötig. Auf der Vorderseite finden sich die sechs typischen Spielcontrollertasten, auf der Rückseite zusätzlich Anschlüsse mit der Belegung W, A, S, D, F, den Maustasten und Bewegungstasten. Der Einstieg erfolgt meist mit vorgegebenen Beispielen die anschließend abgewandelt werden und schließlich Ideengeber für eigene Kreationen sein können.

Einen anderen Zugang zum Thema Coding & Making bieten der BBC Micro Bit und der neu entwickelte Calliope mini (Make, 2016; Micro:bit Educational Foundation, 2016). Beides sind Mikrocontroller, der BBC Micro Bit wird in Großbritannien in der Grundschule eingesetzt, der Calliope mini soll ab 2017 im Saarland ebenfalls in der Grundschule zum Einsatz kommen. Zur Nutzung der Controller benötigt man lediglich eine Spannungsquelle und ein Gerät auf dem programmiert wird, das kann ein Smartphone, Tablet oder Desktop Computer sein. Beide Controller haben eine LED Matrix als Display, zwei Eingabetasten, und einige Pins für optionale Erweiterungen, Programmieren kann man die Geräte via Javascript, Microsoft Block Editor, Microsoft Touch Develop oder Python. Durch diese Modularität sind die Mikrocontroller sowohl für Einsteiger/innen als auch für Fortgeschrittene ein sinnvolles Werkzeug.

Zusammenfassung

Meine Intention war es, darzustellen, dass es nicht am Mangel an kindgerechten Lernumgebungen zum Erlernen des Programmierens liegt, dass dieses Thema nach wie vor nur fragmentarisch im österreichischen Schulalltag behandelt wird. Mit den vorgestellten Plattformen sind hervorragende und für unterschiedliche Altersgruppen adäquate Einstiegsmöglichkeiten zum Erlernen des Programmierens vorhanden, ein Überstieg auf die nächste Ebene ist durch mittlerweile sehr ähnliche Oberflächen keine allzu große Hürde, hier hat Scratch neue Standards gesetzt und ist Vorbild für viele andere Lernumgebungen.

Zu berücksichtigen ist bei diesen Darstellungen, dass Coding nur ein Teilbereich und nicht gleichzusetzen mit informatischer Bildung an sich ist. Ebenso wichtig ist, dass Coding im Unterricht so umgesetzt werden sollte, dass es tatsächlich die Kreativität der Kinder fördert, algorithmisches Denken gefördert wird und man nicht bei Rekonstruktion und Dekonstruktion vorhandener Inhalte verharrt.

LITERATUR

Antonitsch, P. & Hanisch, L. (2014). Computational Thinking im Unterricht der Primarstufe. IMST.

Badura, L. (2012). Das Multimedia-Projekt: Besuch in der VS Neusserling. Zugriff am 7.11.2016. Verfügbar unter: http://www2.mediamanual.at/blog/?p=479

Brandhofer, G. (2008). Scratch – Programmieren mit Kindern. *e-LISA Academy Dossier*.

Brandhofer, G. (2009). Scratchcards. *mostblog*. Zugriff am 2.3.2017. Verfügbar unter: http://www.brandhofer.cc/scratchcard/

Brandhofer, G. (2014). Ein Gegenstand „Digitale Medienbildung und Informatik" – notwendige Bedingung für digitale Kompetenz? *R&E-Source*, *1*, 109–119.

Brandhofer, G. (2015a). *Die Kompetenzen der Lehrenden an Schulen im Umgang mit digitalen Medien und die Wechselwirkungen zwischen Lehrtheorien und mediendidaktischem Handeln* (Dissertation). Dresden: Technische Universität Dresden. Verfügbar unter: http://nbn-resolving.de/urn:nbn:de:bsz:14-qucosa-190208

Brandhofer, G. (2015b). Was ist digitale Bildung? *Erziehung und Unterricht. Österreichische Pädagogische Zeitschrift*, *7–8*, 709–720.

Brandhofer, G. (2016). Scratch-Einführung mit Hilfe eines Online-Tutoriums oder Karten (Handbuch Making-Aktivitäten). *Medienpädagogik Praxis-Blog*. Zugriff am 10.10.2016. Verfügbar unter: https://www.medienpaedagogik-praxis.de/2016/04/28/scratch-einfuehrung-mit-hilfe-eines-online-tutoriums-oder-karten-handbuch-making-aktivitaeten/

Brandhofer, G. (2017). Allgemeines zu Scratch – Education Innovation Studio. *Education Innovation Studio der PH Niederösterreich*. Zugriff am 28.2.2017. Verfügbar unter: http://eis.ph-noe.ac.at/allgemeines-zu-scratch/

Bundesministerium für Unterricht, Kunst und Kultur. (2008). eLearning - Didaktik Fachtagung 2008 - ConfTool Pro Printout. Zugriff am 2.3.2017. Verfügbar unter: http://edidaktik.at/fachtagung08/C2/index.html

Döbeli Honegger, B. (2016). *Mehr als 0 und 1* (1. Aufl.). hep verlag.

Erdmann, J.W. (2011). Didaktische Konzepte aus dem Hut zaubern? Habilitationsvortrag. Zugriff am 12.2.2014. Verfügbar unter: http://ebookbrowsee.net/jwe-habil-vortrag-text-pdf-d52726144

Ford, J.L.J. (2009). *Scratch 2.0 Programming for Teens*. Cengage Learning.

Gabriel, S. (2016). Why Digital Game Based Learning Should be Included in Teacher Education. (N. Pachler, Hrsg.) *Reflecting Education*, *10* (1), 26–38.

Gesellschaft für Informatik. (2016). Dagstuhl-Erklärung: Bildung in der digitalen vernetzten Welt. Zugriff am 29.10.2016. Verfügbar unter: https://www.gi.de/aktuelles/meldungen/detailansicht/article/dagstuhl-erklaerung-bildung-in-der-digitalen-vernetzten-welt.html

Herper, H. & Stahl, I. (2013). Diskrete Modellierung und Simulation - Methoden und Werkzeuge für den Informatikunterricht. In H.U. Hoppe & W. Luther (Hrsg.), *Informatik und Lernen in der Informationsgesellschaft: 7. GI-Fachtagung Informatik und Schule INFOS'97 Duisburg, 15.–18. September 1997* (S. 139–151). Berlin: Springer-Verlag.

Hielscher, M. (2016). Lernumgebungen – ProgrammingWiki. *Lernumgebungen*. Zugriff am 1.11.2016. Verfügbar unter: http://programmingwiki.de/Lernumgebungen

Humbert, L. (2006). *Didaktik der Informatik: Mit praxiserprobtem Unterrichtsmaterial*. Berlin: Springer DE.

International Telecommunication Union. (2016). *Measuring the Information Society Report 2016*. Genf.

Löwenstein, B. & Di Angelo, M. (2012). Lego Mindstorms-Roboter - Coole Klassenkameraden im Programmierunterricht. *Zukunft des Lernens* (S. 127–144). Glückstadt: vwh.

Make. (2016). Calliope mini: Mikrocontroller für die Grundschule. *Make*. Zugriff am 6.3.2017. Verfügbar unter: http://www.heise.de/make/meldung/Calliope-mini-Mikrocontroller-fuer-die-Grundschule-3361271.html

Micro:bit Educational Foundation. (2016). All the bits that make up your BBC micro:bit. Zugriff am 6.3.2017. Verfügbar unter: http://microbit.org/

Mittermeir, R. (2010). Informatikunterricht zur Vermittlung allgemeiner Bildungswerte. In G. Brandhofer, G. Futschek, P. Micheuz, A. Reiter & K. Schoder (Hrsg.), *25 Jahre Schulinformatik in Österreich. Zukunft mit Herkunft* (S. 54–73). Wien: Österreichische Computer Gesellschaft.

Resnick, M., Silverman, B., Kafai, Y., Maloney, J., Monroy-Hernández, A., Rusk, N. et al. (2009). Scratch: programming for all. *Communications of the ACM*, *52* (11), 60. doi:10.1145/1592761.1592779

Steele, G.L. (1990). *COMMON LISP: The Language*. Digital Press.

Universität Wien. (2016). Vorlesungsverzeichnis 2016. *u:find*. Zugriff am 10.11.2016. Verfügbar unter: https://ufind.univie.ac.at/de/vvz_sub.html?details=true&path=26432

ZUM AUTOR

Mag. Dr. Gerhard BRANDHOFER, Studium Lehramt Mathematik, Physik, Chemie, Informatik; Diplomstudium Philosophie, Psychologie, Soziologie; Doktoratsstudium Erziehungswissenschaften. Arbeitsschwerpunkte: Planung, Lehre und Forschung im Bereich des Einsatzes von digitalen Medien im Unterricht der Primar- und Sekundarstufe. Der Einsatz visueller Programmiersprachen im Unterricht, digitale Kompetenzmodelle für Schüler/innen und Lehrende, Entwicklung von Lehrgangscurricula zum

Einsatz digitaler Medien in der Schule, Lehrgangsleitung an der PH Niederösterreich, Lehrender in der Aus-, Fort- und Weiterbildung.