SCHULE AKTIV!
Sonderheft des BMB

CODING

EIN BAUSTEIN DER INFORMATISCHEN BILDUNG

Foto: Samsung

Mit Beiträgen von: Anton Reiter • Gerald Futschek • Karl Josef Fuchs • Helmut Caba • Peter Micheuz • Miles Berry • Martin Bauer • Stephan Waba • Alois Bachinger • Anton Knierzinger • Elisabeth Weißenböck • Bettina Gruber • Bernhard Löwenstein • Johann Stockinger • Claus Zöchling • Horst Jens • Stefan Janisch • Martin Ebner • Wolfgang Slany

Vorwort des Herausgebers

Seit im Jahre 2013 die damalige EU-Kommissarin für Digitale Agenda, Neelie Kroes, eine europaweite Codeweek-Initiative (http://codeweek.eu) ausgerufen hat, macht Programmieren Schule und gewinnt zusehends an Bedeutung. Über eine halbe Million Menschen, vor allem Kinder und Jugendliche, in 46 Ländern Europas und darüber hinaus beteiligten sich 2015 an einer derartigen Aktionswoche bei ca. 7.600 registrierten Events in Form von Seminaren, Workshops, Wettbewerben, Barcamps, Livecoding über Streams udgl. Das Potenzial der Programmierung wird auf internationaler Ebene nicht nur von IT-Firmen, sondern auch von Bildungspolitikern als Qualifikationskriterium für zukünftige berufliche Anforderung in der digitalen Welt von heute gesehen. Coding wurde inzwischen in einigen EU-Mitgliedsstaaten in den schulischen Lehrplänen verankert.

Auch in Österreich werden im Rahmen der diesjährigen EU-Codeweek im Zeitraum vom 15. bis 23.Oktober (www.codeweek.at) wieder zahlreiche Schulen, Firmen und Entwickler daran teilnehmen und so ein Umfeld schaffen, das aus dem Tun heraus (beim sogenannten „maker") erste Programmiererfahrungen ermöglicht. Bei den zahlreichen Summer Coding-Veranstaltungen in fast allen Bundesländern wurde dafür schon vorgearbeitet.

Im Vordergrund steht bei den Coding-Initiativen aber zunächst der Spaß am Programmieren. Lernpsychologen und Fachdidaktiker haben festgestellt, dass bei aktiver Anwendung der vom Massachusetts Institute of Technology (MIT) in Cambridge/USA entwickelten visuellen Programmiersprache Scratch (https://scratch.mit.edu/) bei Kindern im Volksschulalter das logische Denken und die Kreativität gefördert werden.

Allerdings lassen sich die elementaren Grundlagen der Programmierung auch ohne Geräte vermitteln. Die Initiative Computer Science Unplugged (http://csunplugged.org/) setzt sich zum Ziel, mit Lernaktivitäten unter Zuhilfenahme von klassischen (analogen) Anschauungsmaterialien Kindern und Jugendlichen zu zeigen, wie Computer (programmgesteuert) funktionieren und was die geistige (denklogische) Leistung dahinter ist.

In der Zwischenzeit ist das Angebot an Coding-Initiativen (im Web) beinahe unüberschaubar geworden. Neben der Österreichischen Computer Gesellschaft (www.ocg.at) bietet z.B. die Codeacademy (https://www.codecademy.com/) ein breites Gratisangebot in Englisch an, um bspw. die Programmiersprachen Javascript oder Python in Online-Kursen zu lernen. Das Motto der auch auf Deutsch verfügbaren Plattform code.org (https://code.org/) lautet: „Jeder Schüler in jeder Schule sollte die Möglichkeit haben, Informatik zu lernen." Über das Web wird eine 20-stündige Informatikeinführung angeboten.

Die österreichische Niederlassung der Firma Samsung ist bis Oktober auf Roadshow in allen neun Bundesländern unter dem Titel „Coding for Kids" (http://www.samsung.com/at/microsite/digitale-bildung/coding-for-kids.html) unterwegs. Mit dem Institut für Softwaretechnologie der TU Graz wurde das mobile Klassenzimmer konzipiert. Den Kindern aus der Primarstufe und Jugendlichen aus der Sekundarstufe I soll im Rahmen von Workshops das Programmieren von Handy-Apps und Robotern vermittelt und sie so in Computational Thinking eingeführt werden.

Wer programmieren kann, ist in der Lage, Software, Webseiten und (kleine) Applikationen zu erstellen und damit Computern eine Anweisung zu geben. Für so manchen Experten ist Coding eine weitere neue Kulturtechnik, die den gleichen Stellenwert wie Lesen, Schreiben und Rechnen hat.

In Österreich ist Programmieren seit 1985 fixer Bestandteil des Informatiklehrplans an der allgemeinbildenden höheren Schule und auch im berufsbildenden höheren Schulwesen in den verschiedenen Formen des Informatikcurriculums integriert. Die europäischen Initiativen gehen allerdings in Richtung eines eigenen Unterrichtsfaches Coding mit dem Ziel, Programmieren wie eine zweite oder dritte Fremdsprache zu erlernen.

Im vorliegenden Sonderheft von „Schule aktiv" wird die Thematik aus der Sicht der (universitären) Lehre, Schulpraxis, Pädagogik und auch Bildungsplanung behandelt. Neben einigen theoretischen Beiträgen, die sich u.a. mit der gegenwärtig vielfach geforderten digitalen Bildung mit Programmieranteilen auseinandersetzen, werden auch praktische Anleitungen von „Makern für Maker" für erfolgreiches Codieren geboten.

Gerald Futschek plädiert dafür, dass „das Computational Thinking in allen Altersstufen vom Kindergarten bis zur Reifeprüfung und natürlich auch darüber hinaus gelernt und auch gelehrt werden kann." Informatisches Denken dient der Problemlösung, wer programmiert, bestimmt die Computertechnologie.

Karl Josef Fuchs und Helmut Caba unterstreichen in ihrem Gemeinschaftsbeitrag anhand der Berechnung des Binomialkoeffizienten, dass algorithmisches/lösungsorientiertes Denken in seiner klassischen Form über Teilschritte auch weiterhin eine Kernstrategie der Praktischen Informatik in der Schule darstellt.

Für Peter Micheuz ist die Informatik nach wie vor (trotz neuer Begrifflichkeiten und verwandter Disziplinen) das Fundament digitaler Bildung. Unter Bezugnahme auf die Forderungen deutscher Informatiker im Rahmen der Dagstuhl-Erklärung, den Modullehrplan 21 in der Schweiz und die Curriculum-Reform in England, nach der Computing in allen Schulstufen verankert werden soll, empfiehlt er, dass Österreich seinen eigenen Weg gehen möge.

Miles Berry schildert detailliert die Entwicklung und den Verlauf bei der Umsetzung des neuen Unterrichtsfaches Computing in England, das an die Stelle des 2014 ersetzten IT-Curriculums getreten ist. Die Initiative "Computing At School (CAS)", die Raspberry Pi Foundation und der BBC micro:bit tragen diese großangelegte Coding-Reform mit.

Martin Bauer und Stephan Waba erläutern in ihrem Beitrag „eEducation - Digitale Bildung für alle" die gegenwärtige Initiative eEducation Austria des Bundesministeriums für Bildung mit dem Ziel, sukzessive digitale und informatische Kompetenzen in allen Schularten und Schulstufen zu verankern. Dabei soll auf dem DigiComp-Modell aufgebaut werden, das von der Volksschule (digi.komp 4), über die Sekundarstufe 1 (digi.komp 8) bis hinauf in die Sekundarstufe 2 (AHS/BMHS; digi.komp 12/13) reicht.

Den praktischen Teil im Sonderheft eröffnen Alois Bachinger und Anton Knierzinger mit einem Gemeinschaftsartikel. Während Letzterer zur Verdeutlichung nochmals die Bedeutung des Coding für die didaktischen und pädagogischen Möglichkeiten insgesamt hervorhebt, stellt Ersterer das Projekt „Coding in der Grundschule" vor, in dessen Mittelpunkt „Denken lernen – Problem lösen" mit dem Bee-Bot, einem einfachen programmierbaren Spiel-Bodenroboter, steht.

Elisabeth Weißenböck, Bettina Gruber und Bernhard Löwenstein bieten „9 spannende Einstiege ins Programmieren". Vorgestellt wird eine Übung aus Computer Science Unplugged, das Brettspiel Code Master, die Organisation Code.org, die grafische Programmiersprache Scratch, das Browserspiel CodeCombat, der Bodenroboter BeeBot, der Lego Mindstorms Baukasten EV3, die Physical Computing Plattform Arduino mit vielen Aktoren und Sensoren sowie der humanoide Roboter NAO.

Johann Stockinger beschäftigt sich mit dem 2012 auf den Markt gekommenen, kostengünstigen Minicomputer Raspberry Pi, der seiner Einschätzung nach sogar dem bisher weltweit meistverkauften Homecomputer Commodore 64 den Rang ablaufen könnte. Auf der neuesten Version sind zahlreiche Anwendungen und auch Programmiersprachen vorinstalliert.

Claus Zöchling zeigt, wie mit Hilfe von Raspbotics Kinder und Jugendliche unter Zuhilfenahme von Controllern, Sensoren und sonstiger elektronischer Peripherie in die Welt der Technik eintauchen können, ganz gleich, ob es um die Entwicklung von Spielen, Prozessautomatisierung oder das Ansteuern von Robotern geht. Die Raspbotics-Boards können mit allerlei technischem Equipment kombiniert werden, auch mit dem Raspberry Pi.

Horst Jens ist der Auffassung, dass „für einfache Computerspiele keine großartigen Programmierkenntnisse" nötig sind. Bei seinen Programmierkursen vor allem für Kinder und Jugendliche setzt er die objektorientierte, für Entwickler offene Programmiersprache Python ein. Dem Beitrag ist ein längeres Programmlistung angeschlossen, das die Syntax resp. einen Befehlsvorrat von Python zwecks Veranschaulichung eindrucksvoll darstellt.

Stefan Janisch, Martin Ebner und Wolfgang Slany laden mit ihrem Gemeinschaftsartikel die Leserinnen und Leser ein, die an der TU Graz auf non-Profitbasis von Catrobat entwickelte App Pocket Code zu nutzen. Dank einer intuitiv verständlichen visuellen Programmiersprache können Kinder und Jugendliche ihre eigenen Spiele, Animationen und Geschichten direkt am Handy erstellen und werden so zu Repräsentanten des „Einfach-Machens". Ab Herbst 2016 wird auf der MOOC-Plattform iMooX ein kostenloser Online-Kurs zum Thema „Learning to code-Programmieren mit Pocket Code" angeboten.

Wolfgang Slany erläutert im Anschluss das didaktische Konzept des digitalen Klassenzimmers. Lehrerinnen und Lehrer sind aufgerufen, sich mit ihrer Klasse am GalaxyGameJam zu beteiligen und Pocket Code Spiele einzureichen.

Das Special wird – wie geplant – der regulären Ausgabe von „Schule aktiv" Ende September oder Anfang Oktober beigelegt werden. Damit ist sichergestellt, dass alle österreichischen Schulen im Verteiler die auf das Thema Coding spezialisierte Sonderausgabe erhalten. Bei der Interpädagogica in Wien im November wird ein weiteres kleines Kontingent aufgelegt, zudem ist vorgesehen, eine pdf-Version zum Download bereitzustellen.

Als redaktioneller Betreuer und Herausgeber danke ich allen Autorinnen und Autoren für Ihre wertvollen Beiträge sowie dem CDA-Verlag für die gute Zusammenarbeit seit bald drei Jahrzehnten.

Viel Spaß beim Lesen wünscht

Autor

MinR Dr. Anton Reiter
Bundesministerium für Bildung,
Abt. II/8
E-Mail: anton.reiter@bmb.gv.at

Computational Thinking im Unterricht

Der Begriff Computational Thinking, im Deutschen Informatisches Denken, wurde ursprünglich von Seymour Papert in seinem für den Konstruktionismus bahnbrechenden Buch „Mindstorms: Children, computers and powerful ideas“ [1] verwendet und von Jeannette Wing mit ihrem viel beachteten Artikel in der wissenschaftlichen Zeitschrift „Communications of the acm“ [2] so eindrucksvoll propagiert, dass er heute in aller Munde ist. Beide Autoren verwenden den Begriff in Zusammenhang mit jugendlichen Lernenden, obwohl Computational Thinking die Denkweisen der professionellen Computerwissenschaftler zur Lösung ihrer Problemstellungen bezeichnet. Es erhebt sich die Frage, was das Computational Thinking ausmacht, dass es für Schüler und Schülerinnen so bedeutsam ist, was davon im Unterricht verwendet werden soll und vor allem wie es unterrichtet werden kann.

Die Bedeutung von Computational Thinking

Computer und Informatik sind die aktuellen treibenden Kräfte von Wissenschaft, Industrie und Wirtschaft. Neue Erkenntnisse in den Wissenschaften, sei es Physik, Chemie, Biologie, Astronomie, Mathematik und oft auch in Geistes- und Humanwissenschaften sind ohne intelligenten Einsatz der Informationstechnologie mit ausgetüftelten Algorithmen nicht mehr denkbar. Das zeigt die Bedeutung des Computational Thinkings außerhalb der Informatik.

Für die Schule und das Erlernen durch Jugendliche müssen die Konzepte der Computerwissenschaft mit Hilfe geeigneter Beispiele und Aufgaben altersgerecht aufbereitet werden. Dies ist für alle Altersstufen möglich, altersgerechte Systeme, Roboter und Software, können das unterstützen.

Computational Thinking ist dem Mathematischen Denken in manchen Aspekten sehr ähnlich. Während das Mathematische Denken mehr auf das Beweisen von Zusammenhängen abzielt, geht es beim Computational Thinking mehr um das effiziente Erzielen von Ergebnissen. Um zu zeigen, dass die Modelle und Lösungsvorschläge des Computational Thinking richtig sind, können sie letztendlich auch am Computer ausgeführt und damit überprüft werden. Die Ergebnisse des Informatischen Denkens werden so zum Leben erweckt.

Welche Aspekte sollen unterrichtet werden?

Bei der Auswahl der Aspekte, die unterrichtet werden, kommt es darauf an, welchen Wert der Aspekt für die Allgemeinbildung und auch für das Berufsleben hat. Die amerikanische Informatiklehrer Organisation CSTA hat für die operationale Umsetzung im Unterricht die folgenden sechs Aspekte des Computational Thinkings herausgearbeitet [3]:

- Formulieren von Problemstellungen in einer Weise, dass sie mit Hilfe von Computern gelöst werden können
- Organisieren und Analysieren von Daten
- Repräsentieren von Daten durch Abstraktionen wie Modelle und Simulationen
- Automatisieren von Lösungen durch algorithmisches Denken
- Finden, Analysieren und Implementieren von möglichen Lösungen mit dem Ziel eines sparsamen Einsatzes von Ressourcen
- Verallgemeinern und Anwenden dieser Problemlösungsprozesse auf verschiedene andere Problemstellungen

Man kann deutlich erkennen, dass bei dieser Charakterisierung des Computational Thinkings der Problemlösungsprozess im Vordergrund steht. Allein die genannten Verben formulieren, organisieren, analysieren, repräsentieren, automatisieren, finden, implementieren, verallgemeinern und anwenden charakterisieren die Tätigkeiten, die relevant für das Computational Thinking sind. Es beginnt mit dem Formulieren der Problemstellung, das die Bedeutung des sprachlichen Aspekts des Computational Thinkings hervorhebt und schließt mit dem Verallgemeinern, das zeigt, dass versucht werden soll nicht nur sehr spezielle Problemstellungen zu lösen, sondern dass das Lösungsprinzip auch auf andere ähnliche Probleme angewandt werden soll.

Alle diese Aspekte des Computational Thinkings können in unserer von Informationsverarbeitung geprägten Gesellschaft von allgemeinbildendem Wert sein. Es kommt insbesondere auf die Art der Aufgabenstellungen und Art der Vermittlung im Unterricht an, damit die Schülerinnen und Schüler die Denkweisen verinnerlichen und davon profitieren können.

Natürlich ist Informatisches Denken auch für den berufsbildenden Bereich äußerst relevant. Dort hat man den Vorteil, dass man praxisrelevante Problemstellungen bei der Hand hat.

Wie soll unterrichtet werden?

Das Denken kann nur geschult werden, wenn immer wieder neue Problemstellungen gelöst werden. Es kommt also beim Computational Thinking insbesondere darauf an, dass in ausreichendem Maße altersgerechte, dem

Lernfortschritt entsprechende Aufgabenstellungen vorliegen. Die vorrangigste Aufgabe der Lehrkräfte ist es also geeignete Aufgabenstellungen anzubieten. Am besten wird das Denken durch selbstständiges Lösen dieser Aufgabenstellungen geschult, erst in zweiter Linie kann durch Nachvollziehen einer vorgezeigten Lösung, gelernt werden.

Projektorientierter Unterricht und Teamarbeit sind adäquate Methoden, um umfangreichere Problemstellungen des Computational Thinkings zu bearbeiten.

Die CSTA betont auch, dass Computational Thinking folgende Kompetenzen fördert:

- Vertrautheit im Umgang mit Komplexität
- Beständigkeit im Bearbeiten von schwierigen Problemen
- Toleranz für Mehrdeutigkeit
- Fähigkeit, offene Fragestellungen zu behandeln
- Fähigkeit zum Kommunizieren und im gemeinschaftlichen Problemlösen

Das sind Bildungswerte, die heute immer mehr an Bedeutung gewinnen. Die Problemstellungen können gemäß dieser Kompetenzen durchaus auch offen, mehrdeutig, komplex und so schwierig sein, dass sie nur im Team und durch gute Kommunikation (sprachliche Dimension!) gelöst werden können.

Es muss auch unbedingt betont werden, dass das Computational Thinking in allen Altersstufen vom Kindergarten bis zur Reifeprüfung und natürlich auch darüber hinaus gelernt und gelehrt werden kann.

Beispiele für mustergültige Aufgabenstellungen findet man zum Beispiel bei der „Computer Science Unplugged Initiative“ [4], bei der in vielen Aktivitäten das Informatische Denken spielerisch und ohne Verwendung von Computern erlernt wird. Bemerkenswert ist dabei, dass mit diesem Ansatz anspruchsvolle Themen der Informatik selbst Volksschulkindern vermittelt werden können.

Die „Biber der Informatik Challenge“ [5] ist eine weitere internationale Initiative, die für alle Altersstufen eine sehr große Zahl an kurzen Aufgabenstellungen zum Computational Thinking erstellt hat. Alljährlich nehmen an dieser Challenge etwa eine Million Schülerinnen und Schüler weltweit teil. Die Aufgaben sind so gestaltet, dass sie sowohl Schülerinnen als auch Schüler ansprechen und für das Informatische Denken begeistern können, auch oder gerade weil einige dieser Aufgaben schwieriger zu lösen sind.

Die Computerprogrammierung nimmt einen wesentlichen Teil des Informatischen Denkens ein. Es ist in unserer von Informationstechnologie und Computer geprägten Zeit notwendig, dass unsere Jugend ein klares Bild von Funktionsweise, Möglichkeiten und Auswirkungen der Informationstechnologie erhält. Nur ein Verständnis der Grundlagen der Programmierung ermöglicht hier ein klareres Bild. Darüber hinaus ist die Fähigkeit des Programmierens eine echte Ermächtigung, die es erlaubt, Technologie zu bestimmen, statt von Technologie bestimmt zu werden. Computerprogrammierung kann heutzutage mit entsprechenden Systemen altersgerecht in allen Schulstufen vom Kindergarten an unterrichtet werden.

Für den Unterricht an Schulen bedeutet es, dass alle Schülerinnen und Schüler vom Kindergarten an Computational Thinking üben sollen. Sowohl ein eigenes Fach als auch die Integration in alle Fächer ist notwendig. Alle Lehrerinnen und Lehrer müssen in Computational Thinking ausgebildet bzw. fortgebildet werden.

Zusammenfassend kann man sagen, dass das Informatische Denken für alle Lebensbereiche unserer Schüler und Schülerinnen von immer größer werdender Bedeutung ist. Die Inhalte, Bildungswerte und Denkweisen sind von langfristigem Nutzen und können in verschiedensten Situationen auch unabhängig vom Computer angewandt werden.

Autor

Gerald Futschek

Ao.Univ.Prof. Dipl.Ing. Dr. Gerald Futschek ist am Institut für Softwaretechnik und Interaktive Systeme an der Technischen Universität Wien tätig.
E-Mail: futschek@ifs.tuwien.ac.at

Referenzen:

[1] **Seymour Papert: Mindstorms:** Children, computers, and powerful ideas. Basic Books, Inc., 1980.
[2] **Jeannette Wing:** Computational thinking. Communications of the ACM, 49(3), 33-35, 2006.
[3] **CSTA Computational Thinking Task Force:** csta.acm.org/Curriculum/sub/CompThinking.html
[4] **Computer Science Unplugged:** csunplugged.org.
[5] **Biber der Informatik:** www.ocg.at/biber, international: www.bebras.org.

Algorithmisches/Lösungsorientiertes Denken – Eine Kernstrategie in der Praktischen Informatik in der Schule

Prolog

Wir sind umgeben von Informatik. Einzelne Themen der Informatik tragen wesentlich zum Wandel unserer Gesellschaft bei. Dem Prozess der Modellbildung, also dem ‚Nachbau' der Realität mit Mitteln der Informatik, kommt dabei zentrale Bedeutung zu. Dieser ‚Nachbau', der Prozess der Modellbildung, ist wiederum eng mit der Idee des Algorithmus verbunden. Als Thema vor allem in der Praktischen Informatik leistet der Algorithmus-Begriff einen bedeutenden Beitrag für eine Informatik als Wissenschaft. Die aus dem Begriff abgeleitete Fähigkeit zu Algorithmischem/Lösungsorientiertem Denken hat sich mittlerweile zu einer zentralen Strategie bzw. Technik (Schweiger 2010, Schwill 1993) als Outcome eines zeitgemäßen und sinnstiftenden Informatikunterrichts entwickelt.

Algorithmisches/Lösungsorientiertes Denken in der Praktischen Informatik

Algorithmisches/Lösungsorientiertes Denken spielt sicherlich auch in anderen Teilbereichen der Informatik, also der Theoretischen und Angewandten Informatik, auf ihre jeweils eigentümliche Art, eine bedeutende Rolle. Die Behandlung der Strategie im Kontext jedes dieser Themenbereiche würde jedoch den Rahmen unseres Diskussionsbeitrags sprengen. Aus diesem Grund haben wir uns für eine exemplarische Betrachtung im Rahmen der Praktischen Informatik entschieden. Als weiteres Argument für die Wahl der Praktischen Informatik möchten wir anführen, dass prototypisches Problemlösen in der Praktischen Informatik wohl am stärksten verankert ist (s. auch Rechenberg & Pomberger 2006, S. 473ff).

Die Praktische Informatik lenkt damit unseren Blick auf den Prozess des Informatischen Modellierens (Caba & Fuchs 1988; Fuchs 1994; Fuchs & Landerer 2007; Hubwieser 2007).

Bekannte – seit jeher zum Kerncurriculum der Mathematik in den Sekundarstufen, Sek. I & Sek. II, gehörige – Algorithmen sind der Euklidische Algorithmus zur Bestimmung des größten gemeinsamen Teilers zweier natürlicher Zahlen oder das Gaußche Eliminationsverfahren zum Lösen von linearen Gleichungssystemen. Die Implementierung derartiger ‚Lösungsverfahren' in einem Informatiksystem stand Mitte der 80er Jahre zu Beginn eines selbstständigen Informatikunterrichts stark im Fokus des Unterrichtsgeschehens.

Lösungsorientiertes Denken spielt in der Informatik eine zentrale Rolle.

Starten wir unsere Diskussion mit einer Behelfsdefinition für Algorithmisches/Lösungsorientiertes Denken: Algorithmisches/Lösungsorientiertes Denken beschreibt die Fähigkeit …

- … Aufgaben zu analysieren.
- … eindeutige Anweisungen zur Lösung der Aufgabe zu formulieren, um diese Anweisungen schließlich
- ... in eine Programmiersprache zu übersetzen, damit die Implementierung am Computer vorgenommen werden kann.

Der erste Blick jeder Problembetrachtung gilt der Problemanalyse. Je nach Komplexität der Aufgabenstellung wird es sich als sinnvoll erweisen, die Aufgabe in Teilaufgaben zu zerlegen, eine Strategie, die von den beiden Fachdidaktiker(inne)n Schubert und Schwill (2011) als Modularisierung bezeichnet wird. In diesem Zusammenhang plädieren wir jedoch gerade für den Einführungsunterricht in Informatik für eine Behandlung ‚kleiner, atomarer' Aufgaben wie die Übersetzung einfacher mathematischer Formeln/Verfahren. Wir sind nämlich überzeugt, dass bereits durch diese ‚überschaubaren' Aufgaben Grundfragen und Basiskonzepte der Informatik (befreit von jeglichem Ballast) beantwortet und klar herausgestellt werden können.

Der Problemanalyse folgt schließlich die Übersetzung des eigentlichen Algorithmus, die Formulierung der Anweisungen. Die Anweisungen setzen sich im Wesentlichen aus den Kontrollstrukturen Sequenz, Verzweigung und Wiederholung zusammen.

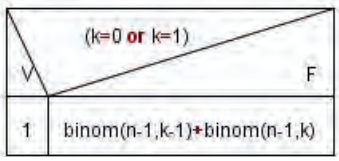
Zudem empfiehlt es sich, den Text für die Anweisungen nicht in der Umgangssprache, sondern als Pseudocode zu formulieren und bereits zu strukturieren. Dies erreicht man etwa durch die beiden folgenden Maßnahmen:

- Die Verwendung einheitlicher Schlüsselwörter zur Kennzeichnung von Zuweisungen und Kontrollstrukturen (etwa: ‚wird zu' und ‚solange'; ‚bis schließlich').
- Die Verwendung von Parametern und Symbolen (etwa: zahl wird zu zahl+1; solange wert <5; bis schließlich betrag<=7).

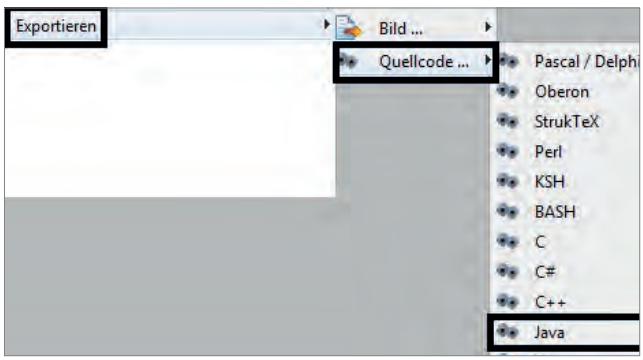
Eine zusätzliche Übersetzung der einzelnen Strukturelemente (Quelle: http://structorizer.fisch.lu/index.php?include=downloads) in grafische Repräsentationen (Nassi & Shneiderman 1973) ist schon aufgrund des Wechsels in der Repräsentationsform von verbal auf ikonisch (Fuchs 2008, S. 60ff) sehr zu empfehlen und zu befürworten. Die angesprochenen Übersetzungen werden in einem zeitgemäßen Modellierungsprozess durch Software wie den Structurizer unterstützt. Nachfolgend beschreiben wir in einzelnen Schritten den prototypischen Weg von der Konzeption eines Struktogramms bis hin zur Code-Generierung in einer bekannten Programmiersprache.

Schritt 1. Entwurf eines Struktogramms für die Berechnung eines Binomialkoeffizienten.

$$binom(n,k) := \begin{cases} 1 & für\ k = 0\ oder\ k = 1 \\ binom(n-1,k-1) + binom(n-1,k) & sonst \end{cases}$$



Schritt 2. (Code_Engineering-Step - Export): Auswahl der Programmiersprache, in die übersetzt werden soll.



Schritt 3. (Eigentlicher Code_Engineering-Step): Unser Struktogramm wird in einen Quellcode in Java übersetzt. Mit der Übersetzung sind wir schließlich beim Schritt der Implementierung angelangt. Die erste Antwort auf die Frage ‚Wie sage ich es meiner Maschine?' haben wir bereits erhalten (Abb. 03). Mit der Wahl der Programmiersprache hat unsere Software, der Structurizer, die Kodierung vorgenommen.

```
// Generated by Structorizer 3.24-11
import java.util.Scanner;

/**
 */
public class Funct_binom
{

    // TODO Declare and initialise class variables here

    /**
     * @param args
     */
    public static void main(String[] args)
    {

        // TODO: Declare and initialise local variables here:

        if (k==0 || k==1)
        {
            1;
        }
        else
        {
            binom(n-1,k-1)+binom(n-1,k);
        }
    }
}
```

Abb: 03

Zumeist wird es sich bei einer Problemlösung nicht um ein einfaches ‚Wasserfallmodell' mit einem Durchlauf wie hier bei der Implementierung eines klar definierten Verfahrens (s. rekursive Definition des Binomialkoeffizienten) handeln.

Für die Praxis ist jedoch ein sich wiederholender Prozess charakteristisch. Der Abschnitt der Implementierung Algorithmischen/Lösungsorientierten Denkens in der Praktischen Informatik enthält deshalb ganz zentral die Elemente Testen und Debuggen.

Dabei stehen beim Testen und Debuggen komplexerer Aufgaben vor allem Laufzeitüberlegungen im Vordergrund. Diese können Modifikationen in der Problemanalyse, in den Übersetzungs- und damit in den Implementierungsschritten, nach sich ziehen. Ein weiterer Durchlauf des gesamten Modellierungsprozesses ist unumgänglich.

Resümee

Kompetenzorientierung ist mittlerweile zu einem Paradigma des Informatikunterrichts geworden. In der Inhaltskompetenz der Praktischen Informatik für die Sekundarstufen I und II spielt der Begriff des Algorithmus eine zentrale Rolle. Mit diesem Beitrag wollen wir die Verknüpfung der Inhaltskompetenz mit der Handlungskompetenz des Algorithmischen/Lösungsorientierten Denkens bei Schülerinnen und Schülern in einem modernen, sinnstiftenden Informatikunterricht vorschlagen.

Diplomarbeiten, Dissertationen und Habilitationen zu diesem Themenbereich und auch zu verschiedensten Themen der Fachdidaktik Informatik finden sich auf der Forschungsseite der Paris Lodron Universität Salzburg http://bit.ly/salzburg_uni unter Betreuung Dipl/Diss/Habil.

Autoren

Ao. Univ.-Prof. Mag. Dr. Karl Josef Fuchs

School of Education – AG Didaktik der Mathematik und Informatik & Fachbereich Mathematik
E-Mail: Karljosef.fuchs@sbg.ac.at

OStR. Prof. Mag. Helmut Caba

School of Education – AG Didaktik der Mathematik und Informatik Universität Salzburg und Pädagogische Hochschule Salzburg
E-Mail: helmut.caba@salzburg.at

Literatur:

Caba, H. & Fuchs, K. (1988). Informatik Heute 5. Salzburg: Salzburger Jugend Verlag
Fuchs, K. J. (2008). Teacher Studies in Austria – Bridging the Gap between Mathematics and Informatics Education. In: Informatics Education Europe III – Proceedings (Cortesi, A. & Luccio, F. Hrsg.). Venedig: Universita Ca'Foscari, S. 52-66

Fuchs, K. J. (1994). Didaktik der Informatik: Die Logik fundamentaler Ideen. In: Medien + Schulpraxis, 4/5, S. 42-45

Fuchs, K. J. & Landerer, C. (2007). Problembasiertes Lernen im Informatikunterricht. In: Problembasiertes Lernen (Zumbach, J.; Weber, A.; Olsowski, G. Hrsg). Bern: h.e.p. verlag ag, S. 159-175
Hubwieser, P. (2007). Didaktik der Informatik: Grundlagen, Konzepte, Beispiele. Berlin, Heidelberg:Springer Verlag

Nassi, I. & Shneiderman, B. (1973). Flowchart techniques for Structured Programming. In: SIGPLAN Notices 8,8, S. 12-26

Rechenberg, P. & Pomberger, G. (2006). Informatik-Handbuch. München, Wien: Hanser Verlag
Schubert, S. & Schwill, A. (2011). Didaktik der Informatik. Heidelberg: Spektrum Akademischer Verlag

Schweiger, F. (2010). Fundamentale Ideen. In K. J. Fuchs (Hrsg.), Schriften zur Didaktik der Mathematik und Informatik an der Universität Salzburg. Aachen: Shaker Verlag

Schwill, A. (1993). Fundamentale Ideen der Informatik. In: Zentralblatt für Didaktik der Mathematik, 25, 1, S. 20-31

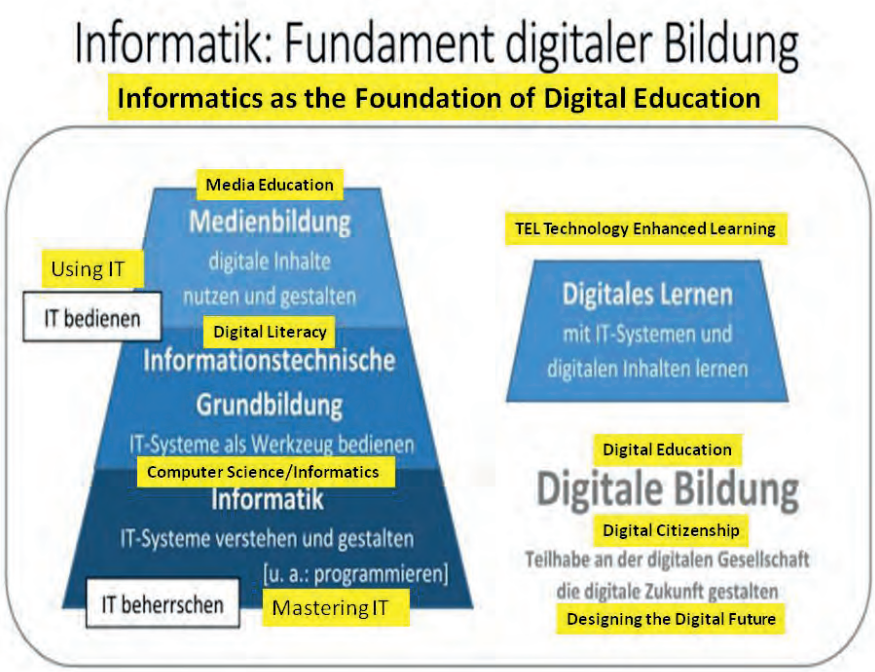
Anmerkungen zur Digitalen Bildung

Der Begriff Informatik und davon abgeleitet das Schulfach mit der gleichen Bezeichnung, das in vielen österreichischen Schulen seit den 1980-er Jahren angeboten, gewählt und unterrichtet wird, haben zu unterschiedlichen Interpretationen und (Fehl)Vorstellungen geführt. Für den (inter)nationalen Beobachter eines ebenso (inhärent?) inhomogenen wie in Zeiten zunehmender Digitalisierung bedeutender werdenden Bildungsbereiches ist das Bild der „Schulinformatik" in den letzten Jahren bunter, vielfältiger und gleichzeitig klarer geworden. Ihre Einordnung in ein unscharfes und weites Begriffsfeld, das derzeit von „eEducation" („elektrisierender und/oder exzellenter Erziehung") und „Digitaler Bildung" angeführt wird, kann gelingen und damit für eine interessierte Öffentlichkeit und für österreichische Bildungsverantwortliche verständlich gemacht werden.

Dazu möchte ich mit ein paar Anmerkungen aus einem Aufsatz beginnen, den ich für eine internationale Konferenz anlässlich 20 Jahre Schulinformatik (ISSEP: Informatics in Schools: Situation, Evolution and Perspectives) an der Universität Klagenfurt im Jahr 2005 veröffentlicht habe.

- The digital gap between pupils at the end of lower secondary schools is unacceptably wide.
- The situation/role/importance of ICT/informatics differs extremely from school to school, due to autonomy and inhomogenious informatics competences of teachers as well.
- There is a need for a reasonable framework which ensures also a certain level of digital literacy.
- Students leaving lower secondary level should prove a reasonable standard of informatics competence.
- Concretion of the curriculum in grade 9 is of high concern.
- Standardizing measures especially up to and for grade 8 (one year before end of compulsory education) should be taken.
- Simplification and clarification of terminology in the context of ICT and informatics is urgent. (Mathematics in schools covers the range from primitive calculating to abstract proofs).
- Can the subject „Informatics" still hold for elementary ICT competences as well as for pure and core Informatics?

Im Gegensatz zum nach wie vor „schlampigen" Zustand Digitaler Bildung im österreichischen Schulwesen vor allem im schulpflichtigen Alter, um es einmal salopp und österreichisch auszudrücken, haben sich die in diesen Feststellungen spürbaren terminologischen Unsicherheiten zum Besseren geändert. Dazu soll folgende veranschaulichende Grafik (© Fachdidaktische Gespräche Königstein, BRD) beitragen.



Wir haben es nicht nur in der Theorie, sondern auch praktisch-pragmatisch mit drei unterschiedlichen Bereichen zu tun, die leider immer noch von vielen und oft in einen Topf geworfen werden. Eine zentrale Rolle spielt der erst in letzter Zeit ins Spiel gekommene und möglicherweise unglücklich gewählte Oberbegriff „Digitale Bildung" (weil Bildung per se nicht digital sein kann). Dabei weiß ich nicht, von wem und ab wann dieser geprägt wurde und in der Bildungsdiskussion zunehmend als wichtiges Thema wahrgenommen wird. Früher sprach man einfach vom „Computer in der Bildung".

Digitale Bildung ist von digitalem Lehren und Lernen zu unterscheiden. Es herrscht Klarheit darüber, dass „von und über digitale Medien/Technologien zu lernen, sich digital bilden" etwas völlig anderes bedeutet als „mit digitalen Medien technologiegestützt lernen". Letzteres steht in diesem Artikel nicht unmittelbar zur Diskussion. Oder sollte es doch? Zweifellos ist es eher Regel als Ausnahme, dass digitale Medien auch beim Erwerb Digitaler Bildung eine nicht unwesentliche Rolle spielen. Diese gegenseitige Abhängigkeit und Rückbezüglichkeit schwingen natürlich immer mit. Aber nicht so stark, dass auf TEL (technologieerweitertes Lernen) in diesem Aufsatz

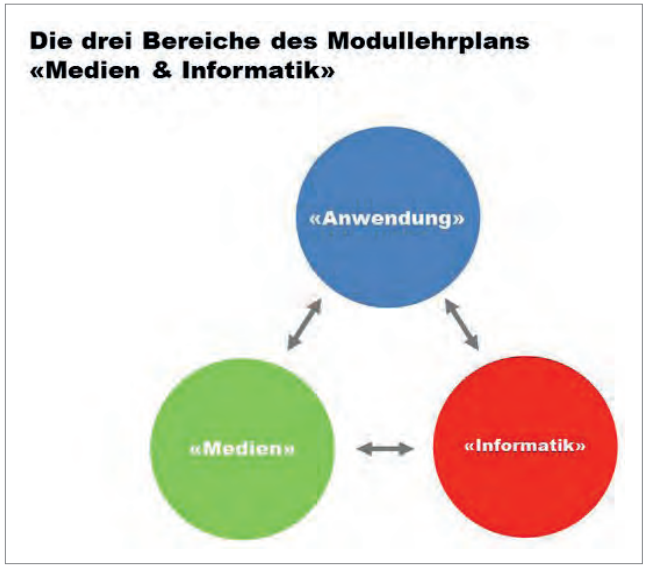
besonders eingegangen werden muss. eLearnerinnen und eLearner (das e steht hier auch für „eingefleischte") mögen mir das verzeihen.

Wenn ein vorausschauender und zurückblickender, in die Jahre gekommener „digital immigrant" hierzulande Forderungen an die Bildungspolitik stellt, mag das etwas anmaßend sein. Wenn Anliegen und Wünsche zur besseren Verankerung Digitaler Bildung im Schulwesen von einer hochkarätigen deutschsprachigen Expertengruppe kommen, wie im Februar 2016 im Rahmen der Dagstuhler Erklärung zur Digitalen Bildung, hat das ein ganz anderes Gewicht. Dort heißt es unter anderem:

In gemeinsamer Verantwortung von Medienpädagogik, Informatik und Wirtschaft fordern wir:

- Bildung in der digitalen vernetzten Welt (kurz: Digitale Bildung) muss aus technologischer, gesellschaftlich-kultureller und anwendungsbezogener Perspektive in den Blick genommen werden.
- Es muss ein eigenständiger Lernbereich eingerichtet werden, in dem die Aneignung der grundlegenden Konzepte und Kompetenzen für die Orientierung in der digitalen vernetzten Welt ermöglicht wird.
- Daneben ist es Aufgabe aller Fächer, fachliche Bezüge zur Digitalen Bildung zu integrieren.
- Digitale Bildung im eigenständigen Lernbereich sowie innerhalb der anderen Fächer muss kontinuierlich über alle Schulstufen für alle Schülerinnen und Schüler im Sinne eines Spiralcurriculums erfolgen.

Deutschland ist noch auf dem Weg zur flächendeckenden Verankerung Digitaler Bildung in Schulen. Konzeptionell und präskriptiv kommt dies durch die kürzlich verabschiedete Dagstuhl-Erklärung zum Ausdruck. Im Zentrum dieses Manifests spielt das von den hegemonial-bipolaren Kampfbegriffen Medienpädagogik und Informatik befreite Dagstuhl-Dreieck eine zentrale Rolle. In der Schweiz ist man in der schulischen Umsetzung Digitaler Bildung bereits einen Schritt weiter.



Dazu empfehle ich den pointierten Beitrag von Beat Döbeli auf http://blog.doebe.li/Blog/DagstuhlDreieck. Er versteht wie kein anderer das „Jonglieren mit Digitaler Bildung". Nach langjährigen Diskussionen im Zuge des Lehrplanes 21 hat das Teilfach „Medien und Informatik" mit einem „Modullehrplan" einen zumindest theoretisch abgesicherten Platz im Kanon der neu konzipierten Fächer gefunden, nicht zuletzt wegen einer sauberen begrifflichen Klarheit. Derzeit ist von den Mühen der Umsetzungsebenen in den einzelnen Kantonen die Rede. Aber das ist allen Reformen innewohnend. Den künftigen Entwicklungen bei den Eidgenossen darf mit Interesse entgegengesehen werden.

Mindestens als ebenso spannend ist die - nicht nur aus österreichischer Perspektive – revolutionäre Entwicklung der Digitalen (Schul)Bildung in England zu bezeichnen. Oberflächlich betrachtet, geht es in englischen Schulen derzeit um den Paradigmenwechsel „ICT raus, Computer Science rein", und zwar von der Primarstufe (Keystage 1, 5-7 Jahre) bis zur mittleren Reife (Keystage 4, 14-16 Jahre).

Begonnen hat diese imposante (und/aber auch in Österreich mögliche/wünschenswerte?) Entwicklung vor einigen Jahren konzept- und evidenzbasiert. Das „Curriculum Framework for Computer Science and Information Technology" schaffte zusammen mit vielen anderen Publikationen einigermaßen begriffliche Klarheit und gleichzeitig die Basis für ein neues Fach „Computing". Begrifflich etwas irreführend und wohl beeinflusst vom angelsächsischen Pendant zur Informatik, nämlich von „Computer Science" und dem für mich unverständlichen großen Hype „Computational Thinking", deckt dieses Fach im Wesentlichen die gleichen Bereiche ab wie die deutschsprachige Interpretation von Digitaler Bildung (Informatik, Medien und Anwendung). Zum Thema „Computational Thinking" darf ergänzt werden, dass bereits im Jahr 1985 und somit vorausschauend im Lehrplan Informatik AHS

5. Klasse „die Schüler den gegenwärtigen Stand der Informatik, insbesondere ihre Denk- und Arbeitsweisen […] kennenlernen sollten“. Die österreichische Variante von Computational Thinking: Informatische Denkweisen, sic! Aber leider nur „kennenlernen“ …

Ein Auszug aus dem Report der Royal Society Anfang 2012 kann als einer der Auslöser der für viele überraschenden substanziellen Lehrplanreform in England gesehen werden. „The current delivery of Computing education in many UK schools is highly unsatisfactory. Although existing curricula for Information and Communication Technology (ICT) are broad and allow scope for teachers to inspire pupils and help them develop interests in Computing, many pupils are not inspired by what they are taught and gain nothing beyond basic digital literacy skills such as how to use a word-processor or a database. This is mainly because the current national curriculum in ICT can be very broadly interpreted and may be reduced to the lowest level where non specialist teachers have to deliver it.“

Die Reaktion darauf war ein nationales Computing Curriculum und ein bildungspolitisches Experiment, das seinesgleichen sucht und von einer imposanten institutionalisierten und gleichzeitigen bottom-up Bewegung „CAS – Computing at School“ mit über 20.000 Mitgliedern begleitet wird. Dass solche politische Weichenstellungen in einem demokratischen Staat nicht von einem Tag auf den anderen erfolgen, ist verständlich und im Web gut dokumentiert (www.computingatschool.org.uk). Eine kompakte Zusammenfassung findet sich auf www.ahs-informatik.com/internationales/computing-in-uk.

Wir in Österreich sollten die internationalen Entwicklungen im Bereich der Digitalen Bildung nicht nur beobachten, sondern uns auch vermehrt in die internationale Diskussion aktiv einbringen. Auf Basis dieses Erkenntnisgewinns und Blickes über den österreichischen Gartenzaun hinaus gilt es, unter den hiesigen soziokulturellen, wirtschaftlichen und politischen Rahmenbedingungen den digitalen Königsweg zu finden.

Mit der Bündelung der Kräfte im eEducation-Projekt des Bildungsministeriums könnte es mit passenden Strategien besser als bisher gelingen, Digitale Bildung in all ihren Facetten in unseren Schulen breit, ausbalanciert und nachhaltig zu verankern.

Autor

Peter Micheuz:

Peter Micheuz ist seit 1981 EDV-, ab 1985 Informatiklehrer und IT-Kustos (ab 2016 IT-Manager) am Alpen-Adria-Gymnasium Völkermarkt und seit 2000 Lehrbeauftragter für Informatikdidaktik an der Alpen-Adria-Universität Klagenfurt. Er ist mehrfacher Teilnehmer und Organisator von einschlägigen regionalen, nationalen und internationalen Konferenzen sowie Lehrbuchautor, Herausgeber von Sammelbänden und sechs CDA-Sonderausgaben, Mitarbeiter in diversen Arbeitsgruppen des Bildungsministeriums, und Publizist von einschlägigen Fachartikeln. Er war eLSA-Bundeslandkoordinator, ist derzeit langjähriger ARGE-Leiter für Informatik an AHS in Kärnten, Vorstandsmitglied im Verein ECDL an Schulen, und seit 2015 Vice-Chair der IFIP Working Group 3.1 (International Federation for Information Processing, Arbeitsgruppe „Informatics and Digital Technologies in School Education“) und damit auch Kenner der internationalen Szene.

E-Mail: peter.micheuz@aon.at

Er ist Betreiber der Portale:
www.digitale-bildung.at

www.ahs-informatik.at

Computing in English Schools

In September 2014 England replaced its old ICT curriculum, which had a focus on building skills using a range of software tools, with a new subject, 'computing', that included significant elements of computer science for pupils from age five to 16, including an expectation that even our youngest pupils would learn to write their own simple programs. Whilst it's still early days, the indications are that the change has been largely successful, with pupils enjoying the challenge of the new subject, most teachers feeling confident teaching computing and a significant rise in the numbers of students taking qualifications in computing at 16+.

Rationales

Why should we teach computer science to children? I think it's possible to give a number of rationales for this.

I'm sure that for ministers, the main argument is an economic one: to sustain or develop vibrant technology sectors requires a stream of computer science graduates, and a firm foundation in computing at school level helps ensure this. This makes sense at individual level too: software engineering is a meritocratic field, and learning to program can do much to promote social mobility.

However, school is perhaps too early for vocational training for the software industries. We don't teach poetry or music in schools because we need more professional poets or musicians, but because these fields provide unique insights into human experience and great scope for pupils' creative expressions: the same is true of programming. Given the role of digital technology in our lives and society, some understanding of the principles on which it is built should be an entitlement for all. More than that, whilst still a relatively new field, computer science like the natural sciences as providing insights into the nature of reality: for example, some problems are easy, some are impossible, some are hard and some are hard unless we think about them the right way.

There's also an argument that learning to program helps develop a particular way of thinking, of looking at problems or systems in such a way that a computer can help us solve or understand them. As Seymour Papert wrote back in 1980,

I began to see how children who had learned to program computers could use very concrete computer models to think about thinking and to learn about learning and in doing so, enhance their powers as psychologists and as epistemologists.

From ICT to Computing

England's journey from ICT to Computing was a rapid one. As recently as 2009, proposed changes to the ICT curriculum emphasised the need for pupils to use and apply ICT skills in their learning and everyday contexts, and to become independent and discerning users of technology: no one would argue that these are bad things, but the ambition here seems, in retrospect, somewhat limited.

In 2010, the Royal Society began a review the state of computing education, taking a view that curricula should ensure students could engineer technology rather than merely consume it. Ian Livingstone and Alex Hope, key figures in Britain's games and visual effects industries, recommended that computer science be brought into our national curriculum as an essential discipline to ensure the long term success of their industries. Google's Eric Schmidt used his speech at the 2011 Edinburgh Television Festival to express surprise that we weren't teaching CS in schools and thus risked throwing away our 'great computing heritage'. England's education inspectorate and the commission reviewing our national curriculum also were critical of the then state of ICT education.

In light of these views, the then education minister Michael Gove announced in January 2012 that the old ICT curriculum would be 'disapplied', allowing schools the freedom to introduce more computer science themselves. Subsequently he announced that an expert panel would be convened under British Computer Society / Royal Academy of Engineering chairmanship to develop a new curriculum, drawing its membership from industry, academia and schools.

Curriculum content

The computing curriculum developed by the panel, and subsequently revised through public consultation, recognised that computing as a school subject had three elements: computer science, information technology and digital literacy. It's helpful to think of these as the foundations, applications and implications of computing, respectively. Important as computer science is, as the core of computing, it's still necessary that children leave school being able to find information, develop content for the web, edit video and so on. It's also vital that they can think through the ethical implications of technologies and consider their own moral responsibility when working with digital technology.

The new programmes of study for computing set out an ambitious vision:

A high-quality computing education equips pupils to use computational thinking and creativity to understand and change the world.

Creativity characterised much of pupils' work under the old ICT curriculum, and remains central to computing. Computational thinking is a golden thread running throughout the new curriculum. It includes concepts such as logical reasoning, algorithms, abstraction, decomposition and generalisation which are central to programming, but also to the much more general use of computers to solve problems and get useful things done. Computational thinking also encompasses a range of approaches, the ways of working common to software engineering, but certainly not limited to this domain. These include tinkering, making, debugging, persevering and collaborating: good computing education needs to develop pupils' capabilities here as well as their grasp of the underpinning concepts. (Abb. 01)

implemented on digital devices (which are as likely to be floor robots or iPads as laptops or desktops at this age). Pupils learn to create and debug programs, as well as reasoning about what a program will do - this latter is crucial. They also use technology to create and manipulate digital content, as well as learning the basics of online safety, including privacy.

Seven to 11 year olds build on this, including sequence, selection, repetition and variables in their programs, and using logical reasoning to debug their algorithms or programs. In most schools, the Scratch toolkit from MIT proves well suited to these requirements. Pupils learn about how the internet, the web and search engines work. They're expected to select software to accomplish goals, to design systems and content, and to analyse and evaluate both data and information. The digital literacy expectations include pupils taking responsibility for their actions. (Abb. 02)

Abb. 01

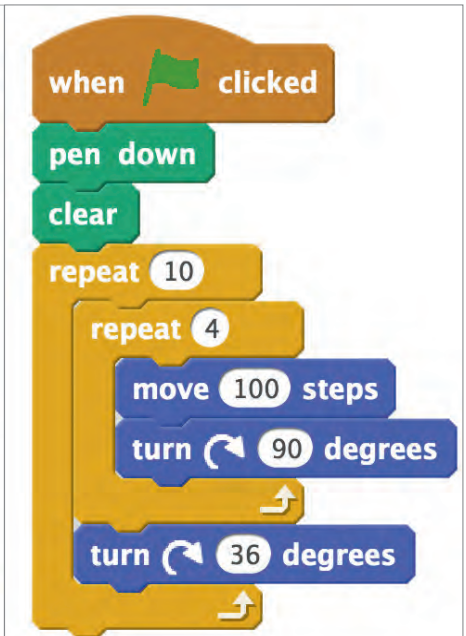


Abb. 02

The foundations of computational thinking are laid even before pupils start formal education. Our guidance for pre-school education includes developing 'characteristics of effective learning', which include 'finding ways to solve problems', 'testing ideas', and 'planning, making decisions about how to approach a task, solve a problem and reach a goal'. These are all part of pupils' everyday experience in pre-school, but are also characteristics of software engineers at Google, Microsoft and Facebook.

For five to seven year olds, the new curriculum includes an expectation that pupils should understand what an algorithm is, as well as how algorithms are

Between 11 and 14, the expectations are greater still, including the design of computational abstractions, the use of more complex data structures such as lists or arrays, and functions or procedures in their code. There's an expectation that pupils will learn at least one text-based programming language at this stage: Python seems to be the popular choice. Some of the underpinning theory of CS is covered here too, including classic algorithms for search and sort, the fetch-decode-execute cycle, Boolean logic and binary arithmetic. There's an expectation that pupils engage in creative projects, including remixing content created and licensed by others. Digital literacy recognises the importance of security, identity and privacy.

```
import random
for i in range(5):
        a = random.randint(1,12)
        b = random.randint(1,12)
        question="What is "+str(a)+" x "+str(b)+"? "
        answer = int(input(question))
        if answer == a*b:
            print("Well done!")
        else:
            print("No.")
```

From 14 to 16, there's a minimum statutory entitlement for computing, phrased in such a way that schools could embed this in cross-curricular provision, but there's also an expectation that students should be able to opt-in to a more demanding specialist curriculum leading to a rigorous, academic qualification in computer science. The requirements for these qualifications include material on how computers and programs work, questions on computational thinking and a practical programming project conducted in response to a detailed, unseen brief under closely supervised conditions.

Between 16 and 18, computer science is one of a portfolio of courses which students can elect to study. New specifications are in place emphasising the academic rigour of these qualifications, including topics ranging from working with big data to functional programming. The project work here is much more open ended, with students choosing for themselves a software product to develop or an area to investigate independently. These elective courses are seen as good preparation, not just for computer science degrees but also for degree courses in social and natural sciences, medicine, engineering and mathematics.

Implementation

To change the curriculum in this way needed changes in statutory regulation, but beyond that government has largely stepped back from implementing the change, taking the view that 'government should only do what only government can do.' Thus the specifics of implementing the new curriculum have largely been left to teachers, local authorities, publishers and not-for-profit organisations. Fortunately, England has evolved a vibrant, multi-layered ecosystem supporting pupils and teachers in this domain. A few initiatives deserve specific mention:

Computing At School (CAS)

CAS is a membership association of teachers and others keen to promote and support computer science teaching for all. It has over 23,000 members, and has led many of the initiatives to see computing established in schools. CAS has built up a network of teaching excellence in computer science education, including some 400 or so 'master teachers', lead schools and regional university partners. CAS has developed professional development programmes for primary and secondary teachers, including Barefoot Computing and Quickstart Computing. There's also a vibrant online community with a culture of sharing resources. (Abb. 03)

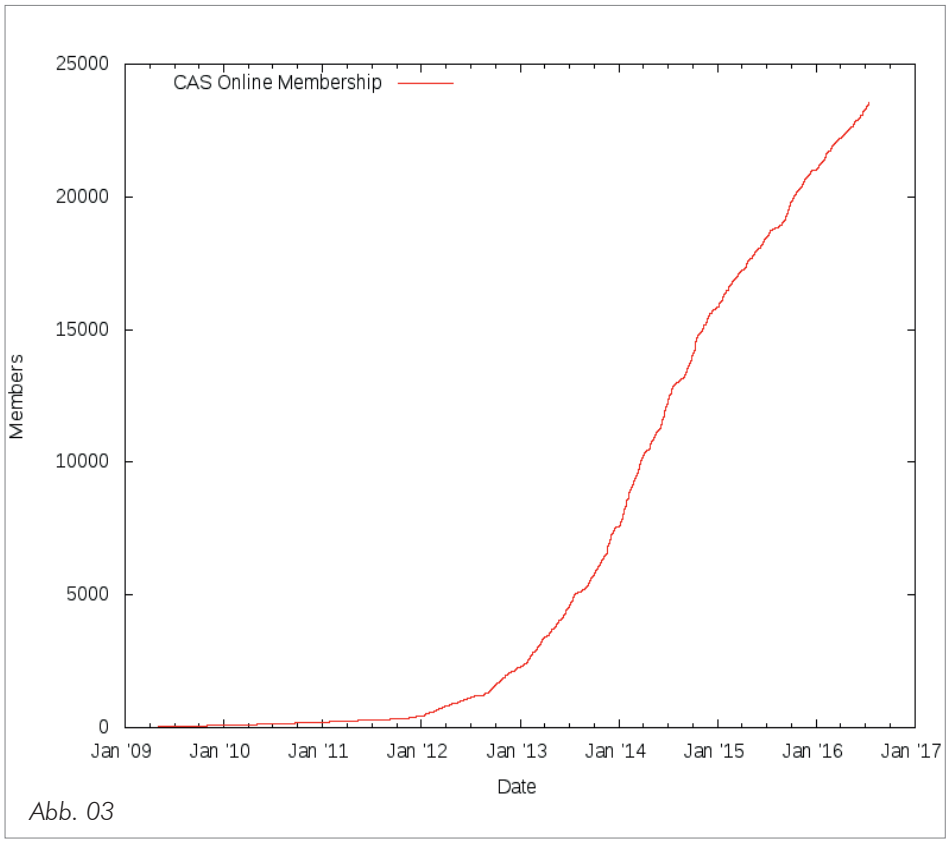
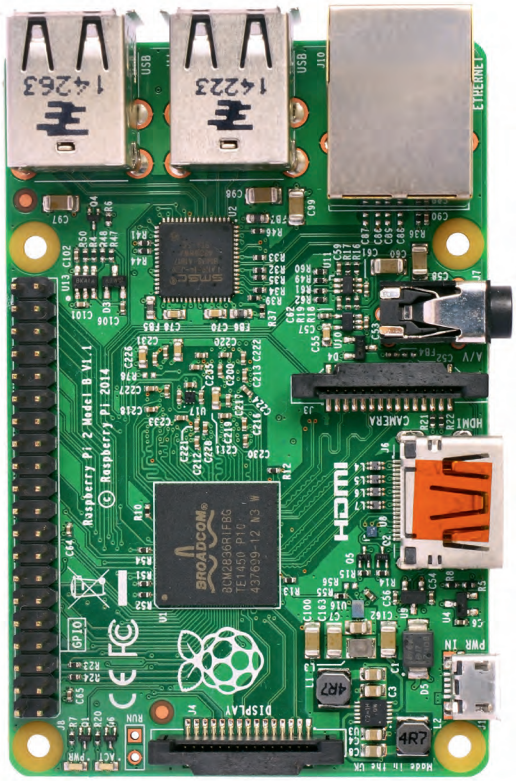


Abb. 03

Raspberry Pi

The UK's most successful computer, the $35, Linux powered Raspberry Pi, was developed with the purpose of providing a platform on which young people can experiment with programming, networking and control. All profits from sales of the Pi go to support the Raspberry

Pi Foundation, whose mission is to 'put the power of digital making into the hands of people all over the world.' The foundation produces some very high quality teaching materials and resources, and runs the innovative Picademy professional development programme. The foundation merged last year with Code Club, who run after-school coding clubs for pupils aged 9-11 as well as a professional development programme for primary teachers. Their teaching resources are also freely available online.



BBC Make It Digital

The UK national broadcaster, the BBC, ran a year long initiative including broadcast and online content to support coding and other aspects of digital making. As part of this, the BBC with 29 partner organisations put a micro-controller based device, the BBC micro:bit into the hands of every eleven year old in the country, with the aim that this would inspire a new generation of digital makers. The micro:bit has a 25 pixel (!) LED display, a couple of input buttons, input/output connectors, accelerometer and Bluetooth connectivity. It can be programmed online in Javascript, Blockly, TouchDevelop and Python.

There are many other organisations that have played a key role in helping to establish computing as a curriculum subject. There's been generous support from Microsoft, Google and ARM, some great work from universities, such as Michael Kölling's Greenfoot Java environment developed at the University of Kent and Paul Curzon's CS4FN project at Queen Mary University of London, and mainstream education publishers including Rising Stars and Hodder Education.

It would be wrong to leave the impression that England has all this sorted already. There's plenty more on the agenda, including delivering a genuinely inclusive entitlement to computing for all; ensuring that computing is taught, and taught well, in all schools; a need to research what makes for effective pedagogy in computing at school; attention to the transition from primary to secondary; and developments around how computing can best be assessed, both formatively and summatively. The journey from ICT to computing has been an exciting one, but the road ahead looks even more interesting.

Autor

Miles Berry

Miles is principal lecturer in Computing Education at the University of Roehampton. Prior to joining Roehampton, he spent 18 years in four schools, much of the time as an ICT coordinator and most recently as a head teacher.

He is a member the management boards of Computing At School and the CSTA, the UK Forum for Computing Education and the Raspberry Pi Foundation. He is a fellow of the BCS, RSA and HEA.

Over the years he has contributed to a number of computing related projects including the national curriculum computing programmes of study, Switched on Computing, Barefoot Computing, QuickStart Computing, CAS TV and Project Quantum.

He gives regular keynotes and CPD workshops on computing and education technology in the UK and abroad and has worked on a number of international consultancy projects involving curriculum development and CPD.

E-Mail: mgberry@gmail.com

eEducation - Digitale Bildung für alle

„Ohne Bildung 4.0 gibt es keine Industrie 4.0. Eine vom digital vernetzten Leben geprägte Gesellschaft braucht das neue Lernen und Lehren. Wir können es uns nicht weiter leisten, Kinder mit Methoden der Vergangenheit für die Zukunft fit zu machen."(Özcan Mutlu, http://bit.ly/gruene_bundestag 10.06.2016)

Digitale Medien haben unsere Welt und unser Leben in einem Ausmaß verändert, wie dies zuletzt wohl bei der Einführung des Buchdrucks der Fall war. Für die kommenden Jahre sagen Experten und Expertinnen eine zunehmende Dynamik des digitalen Wandels voraus. Stichworte dafür sind beispielsweise Industrie 4.0, Internet of Things und Bildung 4.0. Daher sind zeitgemäße Bildungs- und Arbeitsprozesse ohne die Nutzung digitaler Technologien kaum denkbar – digitale und informatische Kompetenzen sind für die Teilhabe an unserer Gesellschaft unerlässlich. Für zahlreiche Berufe, auch viele, die künftig erst entstehen, werden Userkenntnisse nicht mehr ausreichen – eine neue Generation an Arbeitskräften wird verstärkt nachgefragt sein, die Maker-Generation, also jene, die auch über informatische Kompetenzen (Computational Thinking) verfügen und Programmiersprachen beherrschen.

Die Initiative eEducation Austria

Unsere Schülerinnen und Schüler wachsen mit digitalen Medien auf und nutzen diese meist unbefangen und vielseitig. Die notwendigen Kompetenzen zu erwerben, um Technologien bewusst und produktiv für die eigene Weiterentwicklung einzusetzen oder in entsprechenden zukunftsträchtigen Berufsfeldern Fuß zu fassen, fördert die Initiative eEducation Austria des Bundesministeriums für Bildung, mit dem Ziel, digitale und informatische Kompetenzen in alle Klassenzimmer Österreichs zu tragen – von der Volksschule bis zur Reife- und Diplomprüfung. Digitale Bildung für alle!

eEducation Austria fördert die Verbreitung, Intensivierung und Qualitätssicherung von digital-inkludierter Fachdidaktik sowie informatischer Bildung und den verlässlichen Erwerb digitaler Kompetenzen einschließlich ergänzender eLearning-Aktivitäten teilnehmender Schulstandorte im gesamten Bundesgebiet mit nachhaltiger Wirkung.

eEducation Austria bietet eine gemeinsame organisatorische Plattform für schulische eLearning-Netzwerke in Österreich und verfolgt eine einheitliche Strategie zur Verbreitung digitaler und informatischer Kompetenzen. Lehrerinnen und Lehrer von eEducation.Expert.Schulen erhalten im Netzwerk der eEducation-Expertinnen und Experten die Möglichkeit, gemeinsam an geförderten nationalen und internationalen Projekten zu arbeiten, an einschlägigen Fachtagungen teilzunehmen und diese mitzugestalten, sowie voneinander und miteinander zu lernen.

Schulen, die sich der Wichtigkeit des digitalen Themas für den Unterricht und den Schulstandort aktiv annehmen wollen, sind eingeladen, als Mitglied von eEducation Austria von anderen zu lernen, sich zu vernetzen und zu entwickeln. Lehrerinnen und Lehrer benachbarter eEducation.Expert.Schulen und Mitarbeiter/innen des Bundes- und Koordinationszentrums eEducation Austria begleiten mit Fortbildungsmaßnahmen, individueller Entwicklungsberatung und passenden Materialien den Schulentwicklungsprozess.

Im Mittelpunkt aller Aktivitäten von eEducation Austria steht der didaktisch sinnvolle Einsatz digitaler Medien in allen Gegenständen sowie die Steigerung der digitalen und informatischen Kompetenzen von Schülerinnen und Schülern. Es geht um Einsatzszenarien, die einen Mehrwert für das Lernen und Lehren generieren bzw. die Schülerinnen und Schüler darauf vorbereiten, digitale Technologien am Arbeitsplatz kompetent zu benutzen.

Der Weg zur eEducation.Expert.Schule

Schulen, die in einem Innovationsnetzwerk, wie z.B. eLSA, eLC, ENIS, IT@VS, eCool, tätig oder zertifiziert sind, werden aktiv als eEducation.Expert.Schulen eingeladen. Sie sollen als Experts ihr Know-how an andere Schulen weitergeben, die sich am Beginn der digitalen Unterrichts- und Schulentwicklung befinden. Dabei werden sie seitens eEducation Austria unterstützt und erhalten die Möglichkeit, finanzielle Unterstützung zur Finanzierung von Aktivitäten bzw. Projekten anzufordern oder an Tagungen und exklusiven Fortbildungsprogrammen teilzunehmen. Darüber hinaus werden alle Schulen die Möglichkeit erhalten, sich durch Aktivitäten und Leistungen (Badges) als eEducation.Experts zu qualifizieren. Zur Beibehaltung ihres Status müssen alle eEducation.Expert.Schulen jährlich die laufende Dokumentation von Aktivitäten und Leistungen in Form von „Badges" (Sammelpass) über die eEducation-Website nachweisen.

Von einer eEducation.Expert.Schule sind folgende Qualifikationen zu erbringen:

- eEducation-Schulkonzept für den eigenen Schulstandort und Partnerkonzept für Mitgliedsschulen.
- Nachweis der geforderten Punkte pro Schuljahr in Form von Badges, wofür die Punktwerte der einzelnen Badges addiert werden. Ein Badge ist ein

Leistungsnachweis/eine Aktivität im Bereich eEducation, z.B. die Erstellung eines digitalen Contents, die Organisation und Durchführung einer Veranstaltung oder Fortbildung, die Durchführung digitaler Kompetenzchecks durch Schüler/innen, die Anwerbung eines neuen eEducation-Mitglieds, die Beteiligung an einem Projektschwerpunkt (z.B. Game based Learning) etc.

- Jährlicher Aktivitätenbericht an das Bundes- und Koordinationszentrum eEducation Austria, der über die Website www.eEducation.at automatisch generiert wird, indem die Aktivitäten in Form der Badges durch die Schule dort dokumentiert werden.

eEducation-Mitglied werden

Das Einstiegsniveau in das Netzwerk ist der Member-Level. Hierfür ist ein Commitment der Schulleitung notwendig, dass die Schule Interesse an der Vermittlung digitaler und informatischer Kompetenzen habe und bereit sei, einen Schulentwicklungsprozess zu beginnen und an einem Schulkonzept zu arbeiten. Unterstützung erhalten sie hierbei durch eine oder mehrere Expert.Schule/n.

Von einer eEducation.Member.Schule sind folgende Qualifikationen zu erbringen:

- Antrag auf Mitgliedschaft über www.eEducation.at und Bekanntgabe der Ansprechpartner
- Entwicklung eines eEducation-Schulkonzeptes für den Schulstandort mit Unterstützung durch mindestens eine Expert.Schule
- Aktivitäten zur Umsetzung der eEducation-Strategie am eigenen Schulstandort und Dokumentation dieser Aktivitäten über www.eEducation.at zur Darstellung der Entwicklung

Als Nachweis für die Weiterentwicklung und Professionalisierung einer Schule kann diese über die eEducation-Website Aktivitäten dokumentierten, für die der Schulstandort Badges mit entsprechenden Punktwerten erhält. Wird ein von der Schulgröße abhängiger Leistungsnachweis in Form der erforderlichen Punktesumme erbracht, qualifiziert sich die Schule als eEducation.Expert und kann somit selbst zur weiteren Verbreitung von eEducation-Fördermaßnahmen aktiv beitragen, z.B. indem sie selbst Projekte beim Bundes- und Koordinationszentrum zur Finanzierung einreicht und durchführt.

Folgende Kategorien von Badges stehen für eEducation-Aktivitäten zur Verfügung:

- Einsatz digitaler Medien im Unterricht
- Entwickeln und Erproben von eLearning-Szenarien
- Schulübergreifende Kooperation
- Schulentwicklung
- Erwerb digitaler Kompetenzen
- Aktive Verbreitung von eLearning in der Bildungslandschaft
- Einsatz innovativer Lerntechnologien oder Förderung informatischer Bildung
- Einsatz innovativer Lehr-/Lernmethoden
- Sonderbadges

Abbildung: Der Weg zur eEducation.Expert.Schule

Digitale Kompetenzen

Alle Aktivitäten rund um eEducation Austria werden anhand von drei Kompetenzstufen organisiert und strukturiert, die inhaltlich auf dem europäischen DigComp-2.0-Modell digitaler Kompetenzen basieren:

- digi.komp 4 (Grundstufe, Volksschule)
- digi.komp 8 (Sekundarstufe 1)
- digi.komp 12/13 (Sekundarstufe 2, AHS/BMHS)

Für jede der drei Kompetenzstufen/Schulstufen existieren entsprechende Kompetenzraster und Begleitmaterialien/Unterrichtsmaterialien für Lehrer/innen, die erweitert und ausgebaut werden (www.digikomp.at). Daneben ermöglichen Instrumente zur Kompetenzmessung (http://digikomp8.digicheck.at) Schüler/innen und Lehrer/innen, die eigenen digitalen und informatischen Kompetenzen zunächst zu reflektieren und durch eine anschließende Wissensüberprüfung realistisch einzuschätzen und in der Folge möglichst auszubauen. Weitere Tools zur Kompetenzmessung befinden sich in Entwicklung und werden das Angebot künftig erweitern.

Organisation und Vernetzung

Die Idee von eEducation Austria basiert auf Vernetzung und Austausch in so viele Richtungen und zwischen so vielen Ebenen wie möglich:

- Vertikale Vernetzung von der Bundessteuerung (Bildungsministerium, Bundes- und Koordinationszentrum eEducation Austria) zu den eEducation.Expert.Schulen und zurück.
- Horizontale Vernetzung innerhalb der Schulform österreichweit und innerhalb des Bundeslandes mit den Koordinatorinnen und Koordinatoren der anderen Schulformen, der Schulaufsicht, der Fachinspektoren und der Pädagogischen Hochschulen.

Insbesondere die regionale Vernetzung mit Schulaufsicht und Pädagogischer Hochschule ist zentral, um für jedes Bundesland ideale Vorgangsweisen und Lösungswege zu entwickeln, die es Schulen ermöglichen, sich bestmöglich zu entwickeln und von eEducation Austria zu profitieren.

Tagungen und internationale Projekte

Das Bildungsministerium und das Bundes- und Koordinationszentrum eEducation Austria veranstalten pro Jahr die eEducation-Praxistage und eine eEducation-Fachtagung, an denen alle eEducation.Member.Schulen und eEducation.Expert.Schulen teilnehmen können. Für aktive Beiträge, wie z.B. die Abhaltung eines Workshops oder eines Vortrags können Badges verdient werden. Auch an internationalen Projekten, die das Bildungsministerium als Mitglied des European Schoolnet (www.eun.org) vergibt, können eEducation-Schulen teilnehmen und so zur internationalen Vernetzung einen aktiven Beitrag leisten.

Wie fit ist Ihre Schule für die digitale Zukunft?

Nutzen Sie die Möglichkeiten und die Unterstützung, die eEducation Austria und das eEducation-Netzwerk Ihrer Schule bieten! Beginnen Sie die digitale Schulentwicklung und lassen Sie sich von Skeptikern und Zweiflern nicht entmutigen, denn „wir können es uns nicht weiter leisten, Kinder mit Methoden der Vergangenheit für die Zukunft fit zu machen!"

Die digitale Bildungsrevolution hat längst begonnen, sie findet statt und schreitet voran. Ob mit Ihnen oder ohne Sie, das haben Sie selbst in der Hand.

Autoren

Mag. Martin Bauer, MSc

Martin Bauer leitet die Abteilung II/8 IT-Didaktik und digitale Medien im Bundesministerium für Bildung. Er ist Lektor im Wirtschaftspädagogik-Masterprogramm der WU Wien, langjähriger Autor bei Manz Verlag Schulbuch im Bereich Informatik sowie geschäftsführender Gesellschafter der Bauer Software & Co KG.
E-Mail: Martin.bauer@bmb.gv.at

Mag. Waba Stephan, M.A.

Studium Lehramt Deutsch und Englisch an Universität Wien, Ausbildung zum Mediator, zahlreiche Lehrgänge und Ausbildungen zu Mediendidaktik. Erfahrung als Erwachsenenbildner, AHS-Lehrer, Lehrer/innenaus- und fortbildner an PH Wien bzw. KPH Wien/Strebersdorf (Fachdidaktik, Mediendidaktik und Soziales Lernen) und Trainer und Coach für Konzeption, Entwicklung und Betreuung von Blended Learning Szenarien. Schulbuchautor für Verlage Langenscheidt („Your Turn") und ÖBV („Prime Time" und „Killinger Literaturkunde"). Derzeit Mitarbeiter der Abteilung II/8 im Bundesministerium für Bildung (IT-Didaktik und digitale Medien) und Lehrer/innenaus- und fortbildner an Pädagogischen Hochschulen und Universität Wien.
E-Mail: stephan.waba@bmb.gv.at

Coding, ein Bildungsprinzip?

Wenn man vom Bildungsprinzip liest, ist das Unterrichtsprinzip nicht mehr weit. Jeder Lehrer in Österreich aber weiß, dass oft in diesen Prinzipien alles untergebracht wird, was von diversen Gruppen in unserer Gesellschaft an die Bildungspolitik herangetragen wird, wofür man aber nicht bereit ist, Ressourcen in größerem Maßstab zur Verfügung zu stellen. Schaut man sich die Liste der Unterrichtsprinzipien von Politischer Bildung über Sexualerziehung bis Verbraucherbildung aber doch einmal genauer an, dann sieht man, dass ihnen allen ein gewisser transversal thematischer Charakter innewohnt. Das sehen wir auch bei Coding so. Soll also algorithmisches Denken ein Unterrichtsprinzip werden?

Gibt man jedoch Bildungsprinzip in Wikipedia ein, so wird man sogleich auf das „Humboldtsche Bildungsideal" verwiesen. Dieses dreht sich um die beiden Begriffe: „Autonomes Individuum" und „Weltbürgertum". Das gefällt dann schon viel besser. Coding, wie wir es sehen, ist ja nicht das Erlernen einer Programmiersprache, sondern hat mit der Begründung jedes Unterrichtens – „Die Menschen stärken, die Dinge klären" (Hartmut von Hentig) – zu tun, ist dadurch eben auch die Vorbereitung auf das Leben in einer globalen und digitalen Welt.

Will man die Einführung von Coding, algorithmisches Denken, Computational Thinking, Computer Science oder wie immer man unser Vorhaben bezeichnen will, in der Schule verankern, gibt es dafür im Wesentlichen vier Aspekte:

1. Software ist die zukünftige Herausforderung in der IT.

Die heute am stärksten beachteten Entwicklungen der Informationstechnik sind Robotik und Automatisierung, Visualisierung, Datenanalyse (Big Data) sowie Anwendungen der KI-Forschung. In allen diesen Fällen zeigt sich, dass die derzeitigen Herausforderungen in der IT viel weniger die Hard-, sondern viel mehr die Softwareentwicklung betreffen.

2. Algorithmen beginnen immer mehr, oft unsichtbar, eine beherrschende Stellung in unserer Gesellschaft einzunehmen.

Algorithmen sind allgegenwärtig in Wissenschaft, Technik, Wirtschaft und Gesellschaft. Selbstfahrende Autos, automatisierter Aktienhandel und Cyberwar sind dafür Beispiele. Der Anstieg der Geschwindigkeit dieser Anwendungen ist erstaunlicherweise mehr der Verbesserung der zugrunde liegenden Algorithmen als der sie ausführenden Hardware zu verdanken. Daher macht es Sinn, sich im Unterricht mit ihnen zu beschäftigen, ihre Wirkungsweise verstehen zu lernen, ihre Auswirkungen, d.h. die Chancen, die wir durch sie gewinnen, und die Gefahren, die sie bringen, kennenzulernen.

Wissenschaft und Technik

Algorithmen haben, weit über Informatik hinaus, einen großen Einfluss auf wissenschaftlichen und technischen Fortschritt. Fast alle Forschungsgebiete bauen auf Daten und deren Analyse auf. So geht die Wettervorhersage immer mehr in die Zukunft, weil die Algorithmen zur Analyse der Wetterdaten verbessert werden konnten. Der Erfolg selbstfahrender Autos, und damit vielleicht unser Leben, hängt von den Algorithmen zu ihrer Steuerung ab. Roboter sind in vielen Bereichen, über die industrielle Fertigung hinaus, mit Einfluss auf unsere Berufe im Kommen.

Gesellschaft

Algorithmen und ihr Besitz verleihen in unserer Gesellschaft Macht und Einfluss. Big Data Analyse von Wählerdaten ermöglichen in Kombination mit sozialen Netzwerken bessere Information, im schlechtesten Fall Manipulation der Wähler. Algorithmen bestimmen über individuell angepasste Preise für Produkte, unsere Bonität, ob wir mit Flugzeugen reisen dürfen, ob wir einen Job bekommen und welcher Partner zu uns passt.

Wirtschaft

Google ist nicht deshalb so erfolgreich, weil die Suchmaschine so viele Daten hat, sondern weil sie über den besten Algorithmus zu deren Auswertung verfügt. So kann Werbung individuell angepasst und ein zukünftiger Trend erkannt werden. Algorithmen betreiben selbständig Handel an Börsen und sind erfolgreich dabei. Industrie 4.0, Landwirtschaft 4.0, Smart Home usw. sind derzeit gängige Themen bei Veranstaltungen in der Wirtschaft.

3. Computational Thinking fördert die persönlichen und methodischen Kompetenzen unserer Schüler.

Computational Thinking, it represents a universally applicable attitude and skill set everyone, not just computer scientists, would be eager to learn and use.
Jeannette M. Wing

Wir wissen, dass Computational Thinking sowohl die methodischen wie auch die persönlichen Kompetenzen der Schüler erweitert.

Persönlichen Kompetenzen

Arbeiten in Teams entwickelt und fördert die Kommunikationsfähigkeit. Lernen an motivierenden Projekten stärkt Problemlösungsverhalten und -kompetenz. Aber auch Selbstbewusstsein und Selbsteinschätzung werden entwickelt.

Wie sich die persönlichen Kompetenzen verbessern, kann sehr schön an den Ergebnissen des Projekts „Power Girls" gezeigt werden. Dieses Education Group Projekt zeigt, welche positiven Effekte die spielerische Heranführung von Mädchen an die Technik auf deren persönlichen Kompetenzen und ihre Einstellung zur Technik hat. Mädchen, die am Projekt „Power Girls" (siehe www.edugroup.at/praxis/portale/powergirls) teilnehmen, zeigen

- eine emanzipiertere Lebenseinstellung und eine stärkere Orientierung an nicht-traditionellen Vorbildern,
- schätzen ihre eigenen technischen Kompetenzen höher ein,
- sind mehr an technischen Berufen interessiert und
- entscheiden sich signifikant öfter für technische Ausbildungen.

Dieselben Effekte zeigen sich durch Beschäftigung mit Computational Thinking in der Schule aber auch in Freizeitaktivitäten, wie sie z. B. in Form von CoderDojo Vienna von der Österreichische Computer Gesellschaft (OCG) angeboten werden. Die Seite www.coding4you.at gibt für Lehrer einen guten Überblick, wohin Schüler, die besonderes Interesse zeigen, weiterverwiesen werden können.

Methodischen Kompetenzen

Der Umgang mit IT und digitalen Medien stellt eine wesentliche methodische Kompetenz im Leben der Schüler dar. Computational Thinking beinhaltet nicht nur das Erlernen des Umgangs mit Computern. Vielmehr geht es ebenso um kritische Reflexion.

Es hat sich gezeigt, dass Computational Thinking auch die Sprachkompetenz fördert. Codieren von Lösungen verlangt eine Genauigkeit in deren Formulierung, und das Erlernen einer Programmiersprache scheint allgemein die Entwicklung von Strukturen für eine Grammatik zu fördern.

Berufliche Chancen

Sollte Computational Thinking in der Schule eine stärkere Rolle spielen, werden auch die Chancen vieler Schüler im zukünftigen Berufsleben verbessert. Dort entstehen neue Jobs, dort werden viele Schüler in Zukunft ihre Chance finden.

Die Frage, warum wir in Deutschland und Österreich zu wenig Nachwuchs in technischen Berufen auf allen Ebenen haben, hängt aber sehr stark mit der Einstellung der Schüler zu Technik und ihrer Begeisterung dafür zusammen. Die Entwicklung von „Computational Thinking" ist gerade auch ein Weg, Schüler für technische Berufe zu begeistern.

4. Coding bereichert unsere pädagogischen und didaktischen Möglichkeiten

Coding als Unterrichtsprinzip bietet eine Chance zur Weiterentwicklung von Unterricht, weil es Kreativität und Verbindung zum außerschulischen Leben sowie Spaß am Lernen erlaubt. Sehr schön ist dies im 4P-Modell für kreatives Lernen von Mitchel Resnick dargestellt.

Resnick gibt folgende Faktoren für kreatives Lernen an:

- **projects**: Man lernt am besten durch aktives Arbeiten an bedeutsamen angesehenen Themen, Inhalten und Projekten und mit passenden Werkzeugen. Daher soll Unterricht fächerübergreifend, projektorientiert, zielorientiert, kreativ und praxisorientiert sein.

- **peers**: Lernen floriert als soziale Aktivität, wenn Schüler Ideen austauschen, an gemeinsamen Zielen arbeiten und dann ihre Ergebnisse zusammenlegen. Unterricht ist spannend, wenn er teamorientiert und altersadäquat ist.

- **passion**: Wer an Projekten arbeitet, die ihm sinnvoll erscheinen, die Spaß machen und herausfordernd sind, der arbeitet länger, härter und effektiver. Dies führt beim Lernerfolg zu Nachhaltigkeit.

- **play**: Lernen soll Spaß machen, Experimentieren ermöglichen, zu Grenzen führen und immer wieder auch neue Wege erlauben. In den Aufgabenstellungen sollen sich Situationen mit spielerischem Charakter, mit Bezügen zu Alltag, Technik und Phantasiewelten mischen und ergänzen.

5. Die Praxis:

Ein Projekt für Coding in der Grundschule oder „Denken lernen – Problem lösen".

An der Pädagogischen Hochschule der Diözese Linz wird derzeit an der Konzeption eines Projekts „Denken lernen - Probleme lösen" (Leitung Prof. Alois Bachinger) gearbeitet. Das Projekt soll eine didaktisch begründete Einführung in die Nutzung von digitalen Medien in der Grundschule mit besonderer Berücksichtigung der As-

pekte des Problemlösens und des Umgangs mit neuen, kreativen Aufgabenstellungen sein.

Reell - virtuell ©Bachinger

Ziele

Die digitalen Medien sollen in ihrer gesellschaftlichen Bedeutung beschrieben, in Alltagssituationen des Unterrichts erprobt und zur Generierung von Mehrwert im Unterricht eingesetzt werden. Das Projekt beinhaltet die technische Ausstattung von Testschulen, die Erstellung von didaktischen und organisatorischen Konzepten, eine wissenschaftliche Untersuchung des Themas und die Begleitung des Einsatzes an den ausgewählten Schulen sowie Vorschläge für einen späteren Roll-out an den Volksschulen in Österreich.

Didaktik

Als Schlüsselfaktoren für einen erfolgreichen Unterricht in Coding sehen wir, unabhängig von der Altersstufe:

- eine steigende und stets angepasste Abstraktion,
- für die jeweilige Unterrichtssituation richtig ausgewählte Werkzeuge und
- eine Lernatmosphäre, die Kreativität und Begeisterung erlaubt.

Werkzeuge

Bauklötze

„Wenn man etwas lernen will, muss man es in der physischen Welt konstruieren" das ist das Grundprinzip des Konstruktionismus (Seymour Papert, Schüler von Piaget und Erfinder der Programmiersprache „Logo"). Im Projekt wird daher der haptische Aspekt sehr betont – Befehle und Zustände werden zuerst mit Holzmodellen erarbeitet und festgehalten.

Bee-Bot und Bauklötze ©Bachinger

Bee-Bot

Ein Bee-Bot ist ein kleiner, sehr einfacher Spiel-Bodenroboter, der wie eine Biene aussieht. Mit insgesamt sieben Tasten, die direkt am Bee-Bot angebracht sind, kann dieser programmiert werden, einfache Bewegungsabläufe auszuführen. Die Biene kann sich dabei vorwärts und rückwärts bewegen sowie eine 90 Grad Drehung nach rechts oder links durchführen. Bis zu 40 aufeinanderfolgende Befehle können auf den Tasten eingegeben, „programmiert" werden, mit einem „Go-Button" in der Mitte wird die programmierte Sequenz gestartet und abgearbeitet.

Bee-Bot programmieren ©Bachinger

Das didaktische Prinzip für den Einsatz dieses Mini-Roboters realisiert gut die wichtigen Forderungen nach einem vernünftigen, altersgerechten Umgang mit digitalen Medien. Der spielerische Ansatz macht neugierig, ist faszinierend und bringt Freude. Das ist eine, eurobiologisch gesehen effiziente, Grundlegung für erfolgreiches Lernen.

Ganz wesentlich ist, dass durch diese didaktische Vorgangsweise die Fähigkeit des Problemlösens gefördert wird. Die Kinder erwerben und üben die grundlegende Strategie der Hypothesenprüfung, und dies durch Kooperation in der Gruppe. Dies deckt auch den dritten Bereich des Problemlösens ab, die Reflexion = Bewertung der Lösungsschritte in der Hypothesenprüfung.

Computer

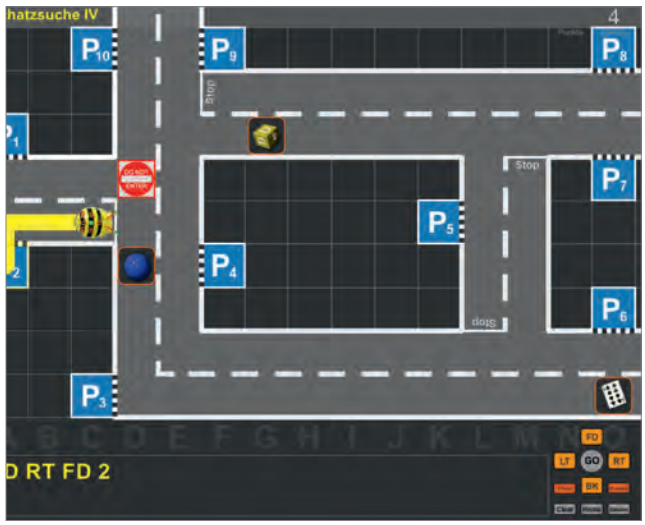
Ganz wesentlich in diesem Projekt ist der Weg vom „Haptischen zum Abstrakten". Die Applikation „Bee-Bot digital" verzichtet nach den obigen Vorübungen bereits auf das haptische Medium „Bee-Bot" und forciert kreatives Spielen mit einer virtuellen Biene am Bildschirm, um Coding weiter auszubauen. Dort ist es für den Spieler dann möglich „die eigenen virtuellen BeeBots" mit dem Smartphone zu steuern.

Bee-Bot App ©Bachinger

men können wir im Alltag entdecken, in Sprache und Schrift beschreiben lernen, ihre einzelnen Schritte definieren.

Stufe 2: Algorithmen als Anweisungen

Lernende geben sich gegenseitig einfache Befehle zur Bewegungssteuerung. Bewegungen werden als Befehls-



Bee-Bot im Verkehr ©Bachinger

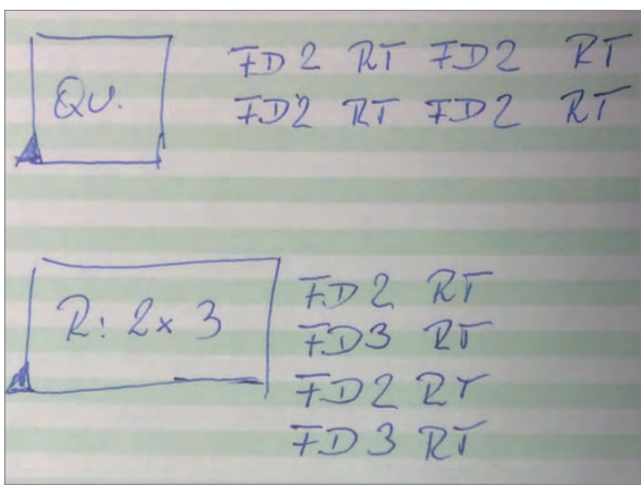
Konzept

Zur didaktischen Umsetzung werden sieben Stufen vorgeschlagen. Das Erlernen des „Programmierens" erfolgt schrittweise – vom „Angreifen" der Bauklötze bis hin zur Steuerung mit dem Tablet – , also vom Dreidimensionalen zum Zweidimensionalen, vom Konkreten zum Abstrakten. Schritt für Schritt werden in sieben Stufen Schwierigkeitsgrad und Herausforderungen gesteigert. Durch permanente Rückmeldungen werden auch kleine Erfolge „sichtbar" und fördern so über das Belohnungssystem des Gehirns effizientes Lernen.

7-Stufen Modell:

Stufe 1: Algorithmen im Alltag entdecken

Algorithmen sind Ketten von Anweisungen. Algorithketten dokumentiert. Die Darstellung von Algorithmen erfolgt mit Befehlskarten und Bauklötzen, eine einfache Schwierigkeitssteigerung durch Variation des Befehlssatzes (verbotene Befehle).



Anweisungen ©Bachinger

Stufe 3: Steuerung eines tastenprogrammierbaren Roboters

Als Roboter wird ein Bee-Bot verwendet. Die Eingabe des Algorithmus zu seiner Bewegungssteuerung erfolgt noch direkt am Gerät, zuerst step-by-step, dann als Befehlsfolge.

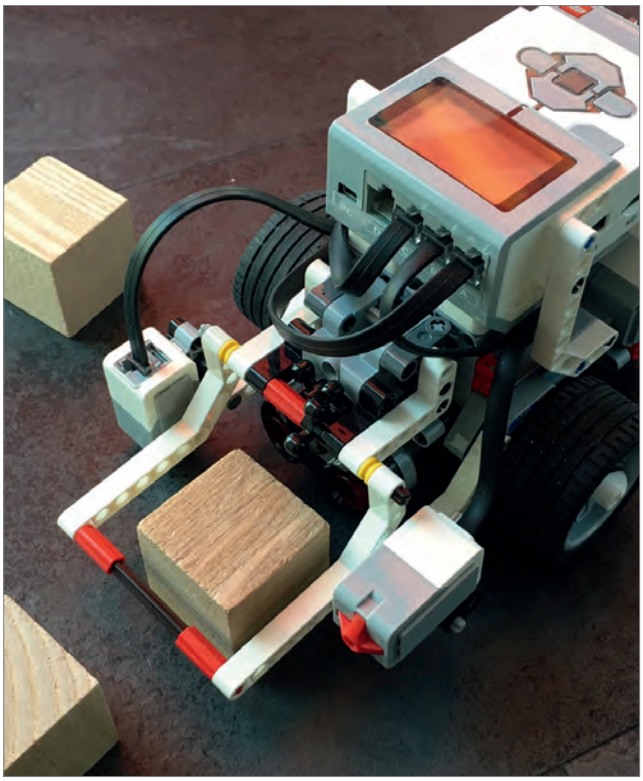
- Schwierigkeitsstufe 1:
  Einführung eines Rasters (Koordinatensystems), ev. mit Landkarte, Brettspielen, …
- Schwierigkeitsstufe 2:
  Übergang vom relativen zum absoluten Bezugssystem
- Schwierigkeitsstufe 3:
  Einführung von Hindernissen, Umwegen und Stopps
- Schwierigkeitsstufe 4:
  Mehrere Roboter, gesteuert von einzelnen Teams, Kommunikation und Kollaboration untereinander, festlegen von Hierarchien

Stufe 4: Algorithmen in maschineller Darstellung

Eine erste Umsetzung in maschineller Codierung erfolgt mit Darstellung der Bewegung des Roboters (Biene) am Bildschirm mit Hilfe der Bee-Bot 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 dann zu lösen.

Stufe 6: Bauen und Programmierung von Robotern
Ausgehend von einem, über Computersteuerung programmierbaren, fertigen Roboter (z.B. Auto) wird zu Roboterbaukästen (zB. Cubelets, WeDo, LegoMindstorm) übergegangen. Es erfolgt die Einführung von Sensoren, Aktoren und Controls. Die Programmierung erfolgt über zur Verfügung gestellte Tools. Einfache Programmstrukturen (Alternative, Iteration, Module, Objekte) werden intuitiv erfasst.



LEGO Mindstorms Robote ©Bachinger

Stufe 7: Programmieren in Scratch
Scratch und verwandte Sprachen, wie BYOB und Blockly, sind vollwertige, block-basierte Programmiersprachen, die eine Einführung in das Programmieren in allen Altersstufen, bis hin auf Universitätsniveau, und dabei auf allen Abstraktionsstufen ermöglicht. Mit Scratch können auch die selbst gebauten Roboter gesteuert werden.

Grundlagen und Theorie

In Fortbildungsseminaren für Lehrer werden die lerntheoretischen Grundlagen (gamebased Learning, emotionales Lernen) erarbeitet. Genauso wird die Verbreitung und Auswirkung von Algorithmen, Coding und Computational Thinking in Technik, Gesellschaft und Bildung beleuchtet.

Durchführung an den Schulen

Die Konzeption des Projektes geht nach obigem Schrittmodell vor und wird in einem betreuten Setting durchgeführt, d. h. IT-erfahrene Schulen (Expertschulen) implementieren vorerst das Konzept in der eigenen Schule und begleiten in den darauffolgenden Semestern Partnerschulen (Memberschulen) in der Durchführung (Schilf, Schülf, Projektkooperationen, …). Das Konzept ist über drei Jahre geplant.

Schlussbemerkung

Algorithmen sind eines der wichtigsten Themen der Zukunft. Coding - Algorithmic Thinking wird immer mehr zur Schlüsselkompetenz der „Citizens of the Digital Age". Wir können zeigen, dass Coding eine informative, kreative und begeisternde Art sowohl des Lernens wie auch des Lehrens bewirken kann. Coding ist weniger „neuer Lernstoff" als eine Weiterentwicklung von Didaktik und trägt somit, wie Informatik erfahrungsgemäß nicht selten, zur Weiterentwicklung von Schule bei.

Autoren

Mag. Anton Knierzinger

Professor em. an der Pädagogischen Hochschule Linz, Gründer und em Geschäftsführer von Education Highway, Mitglied des TC3 - Education - der IFIP
E-Mail: anton.knierzinger@ph-linz.at

Prof. Alois Bachinger

beschäftigt sich seit 28 Jahren mit den Möglichkeiten des Computereinsatzes in der Schule. Viele didaktische Softwareprodukte wurden an der Pädagogischen Hochschule der Diözese Linz in seinen Projektteams gemeinsam mit Kolleginnen und Kollegen erstellt – zB. Schulcad, Interaktiv-Serie (Österreich, Oberösterreich, Wien, Musik, …), Mathematik-Assistent, MultiMedia-Tastaturschreiben, EDU-Puzzle, Denken-Lernen und viele andere. In seiner langjährigen pädagogischen Tätigkeit versucht er den Mehrwert des Computereinsatzes herauszustellen.
E-Mail: baa@ph-linz.at

„Hello World“ - neun spannende Einstiege ins Programmieren

Ziel der Schule ist es, Kinder und Jugendliche auf das Leben von morgen vorzubereiten. Dazu gehört neben den Naturwissenschaften, Sprachen und Kunst auch die wichtigste Kernkompetenz der Zukunft: das komplexe Problemlösen. Ohne diese Fähigkeit werden zukünftige Generationen in der wachsenden technologischen Welt nur sehr schwer Fuß fassen können, da viele einfache Jobs wegfallen werden. Ein geeigneter Weg, um sich diese Qualifikation anzueignen, ist das Erlernen des Programmierens, da jenes mitunter strukturiertes, logisches und strategisches Denken, Analysefähigkeiten und das Entwickeln von Abfolgen fördert.

Welche Möglichkeiten haben Sie als Lehrende/r, um diese Kompetenz bei Ihren SchülerInnen zu fördern? Dieser Artikel stellt Ihnen verschiedene Ansätze zum Erlernen der Programmierung vor - beginnend bei analogen Methoden über kostenlose, spielerische digitale Ansätze bis hin zur Programmierung von Robotern.

Computer Science Unplugged

Computer Science Unplugged bietet eine Fülle an Aktivitäten und Spiele ohne Computer, die an die Denkweise der Informatik heranführen sollen. Die Aktivitäten richten sich an Kinder und Jugendliche und behandeln viele verschiedene Themen aus der Informatik. So gibt es etwa Übungen, die erklären, was binäre Zahlen sind, wie die Fehlersuche funktioniert, was Such- und Sortieralgorithmen sind und wie diese angewendet werden.

Die Übungen von Computer Science Unplugged sind eine gute Einführung in die Informatik, da die Kinder und Jugendlichen nicht stillsitzen und zuhören müssen, sondern durch die Aktivitäten zum Ausprobieren und Mitmachen angeregt werden. Ein Beispiel: Die Fehlersuche. Fehler können immer passieren, sobald Informationen gespeichert oder übertragen werden.

Diese Aktivität zeigt auf einfache Weise, wie diese Fehler gefunden werden und zwar mithilfe eines „magischen Tricks“. Benötigt werden hierzu 36 Post-its oder Kärtchen in zwei verschiedenen Farben. Die Kinder dürfen nun 5x5 Kärtchen beliebig mit den zwei verschiedenen Farben auflegen. Der/Die Lehrende legt danach mit dem Argument, die Aufgabe dadurch noch zusätzlich zu erschweren, eine sechste Reihe und Spalte hinzu. Das Hinzulegen passiert aber nicht willkürlich, sondern dient dazu, dass in jeder Reihe und in jeder Spalte eine gerade Anzahl an Farben liegt. Liegen z.B. in der ersten Reihe drei gelbe und zwei rosa Kärtchen, legt der/die Lehrende ein gelbes Kärtchen hinzu. Somit liegen in dieser Reihe

vier gelbe und zwei rosa Kärtchen. Das muss so schnell passieren, dass den SchülerInnen gar nicht auffällt, dass ein System dahintersteckt. Die restlichen Reihen und Spalten werden ebenfalls um ein Kärtchen jener Farbe ergänzt, die davor in einer ungeraden Anzahl vorhanden war. Gesamt liegen danach 36 Kärtchen auf. Nun gilt es jemand Freiwilligen auszuwählen, der eine Farbe eines einzelnen Kärtchens ändert. Der/Die Lehrende verlässt den Raum oder dreht sich kurz um. Durch dieses Umändern einer Farbe wird in einer Spalte und in einer Reihe die Anzahl der Farben ungerade, wodurch der Fehler vom Lehrenden gefunden werden kann.

In der Praxis werden ähnliche Verfahren zur Fehlersuche z.B. bei der ISBN-13, also der internationalen Standardbuchnummer, verwendet. Dort ist die letzte Ziffer eine Prüfziffer, die sich aus den vorangegangen zwölf Zahlen ergibt. Hat sich ein Fehler eingeschlichen, stimmt diese Prüfziffer nicht mehr.

Auf csunplugged.org können die kostenlosen Aktivitäten inklusive genauer Anleitung für die LehrerInnen heruntergeladen werden.

Wilfried Baumann (OCG): *„Computer Science Unplugged ist eine wertvolle Ergänzung zum Unterricht am Computer. Anhand von spielerischen Aktivitäten werden abstrakte Inhalte für die Lernenden unmittelbar erfahrbar. In den Computer Science Unplugged-Einheiten bleiben die Notebooks zugeklappt, denn die Jugendlichen übernehmen die Rolle des Computers und spielen teils komplexe Abläufe selbst durch. Für viele Lerninhalte ist diese Vorgangsweise unverzichtbar."*

Link: csunplugged.org. Siehe dazu auch die Wiener Zauberschule der OCG (http://www.ocg.at/wizik)

Code Master

Code Master ist ein Brettspiel und erinnert durch seine Aufmachung stark an Minecraft. Dadurch wird das Interesse der Kinder sehr schnell geweckt. Ziel des Spiels ist es, mit einer Spielfigur auf einer Karte vom Ausgangspunkt zum Portal zu gelangen. Je nach Schwierigkeitsgrad gilt es z.B. Kristalle einzusammeln, Gegnern auszuweichen oder Wege so lange zu wiederholen, bis eine Bedingung eintrifft. Wege sind durch färbige Linien vorgegeben, in der sogenannten „Guide Scroll" und am Spielplan ist festgehalten, wie viele Aktionen in welcher Reihenfolge und zu welchen Bedingungen benutzt werden dürfen.

So entsteht ein erster Zugang zur Programmierung. Insgesamt beinhaltet das Spiel zehn verschiedene Karten mit 60 Levels, unterteilt in drei Schwierigkeitsgrade.

Das Spiel richtet sich an Kinder ab 8 Jahren, kann aber durchaus auch für wesentlich ältere Jugendliche eingesetzt werden, da die schwierigeren Levels auch so manchen Erwachsenen ins Schwitzen bringen können. Code Master ist ein Spiel für eine einzelne Person. Daher eignet es sich gut für Kinder und Jugendliche, die im Regelunterricht schnell mit Aufgaben fertig sind und eine zusätzliche Herausforderung gerne annehmen. Die Kosten belaufen sich auf rund 20 Euro pro Spiel.

Link: www.thinkfun.com/products/code-master/

Code.org

Code.org ist eine im Jahr 2013 in den USA gegründete Organisation, die sich zum Ziel gesetzt hat Kinder, Jugendliche und Erwachsene für die Informatik und die Programmierung zu begeistern. Es gibt für jede Altersgruppe eigene Kurse und Curricula, insgesamt richtet sich code.org an Kinder und Jugendliche im Alter von 4-18.

Für Vor- und Grundschüler gibt es wie für Mittel- und Oberstufe einen 20-stündigen Kurs, der die Kinder und Jugendliche altersgerecht in die Programmierung heranführt. Verwendet wird eine grafische Programmiersprache, damit auch jüngeren Kindern der Einstieg in die Programmierung erleichtert wird und Tippen obsolet wird. Auch Aktivitäten ohne Computer sind in den Curricula fest verankert – sei es als Einstieg, zum Festigen von gelernten Konstrukten oder als Auflockerung. Die Lehrinhalte sind für alle frei zugänglich und können jederzeit im Unterricht verwendet werden. Sie sind gut strukturiert und LehrerInnen können sich im Schülerkonto jederzeit einen Überblick über den aktuellen Stand ihrer SchülerInnen machen. Eigens für Lehrende erstellte Arbeitspläne, Unterrichtspläne und Materialien unterstützen dabei einzelne Lektionen umsetzen zu können.

Besonders gut für den Einstieg im Volksschulunterricht eignen sich die sogenannten „Hour of Code". Diese Einheiten sind auf ca. eine Stunde konzipiert und können ohne große Vorbereitung in jedem Computerraum oder auf jedem Tablet durchgeführt werden.

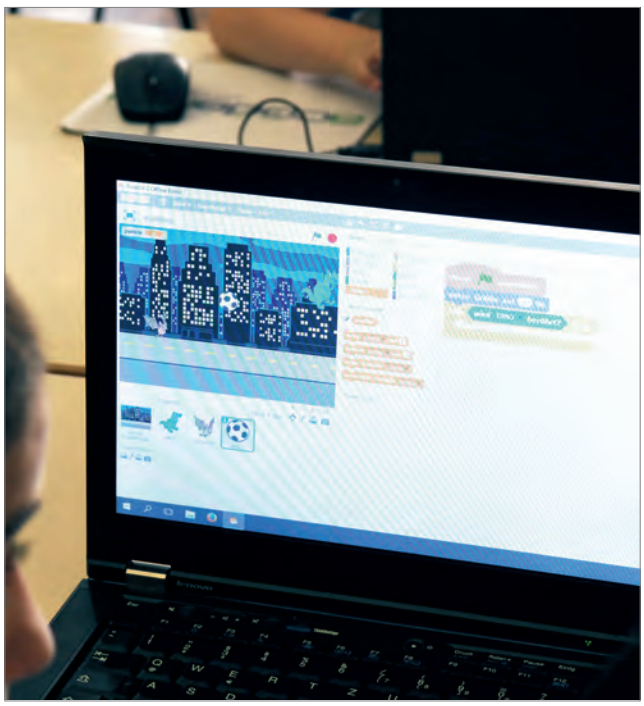
Auf code.org sind alle Kurse und Aktivitäten kostenlos. Lediglich eine Anmeldung ist für Lehrende notwendig, damit sie Zugriff auf alle Unterrichtsmaterialien erhalten. Die Kinder und Jugendlichen benötigen ein Tablet oder einen Laptop, um die Übungen zu absolvieren.

Bettina Gruber (HTL Spengergasse): *„Das zweite Jahr in Folge habe ich mit code.org die SchülerInnen der ersten Klasse HTL an das Programmieren herangeführt. Selbstständig und im eigenen Tempo konnten die SchülerInnen die Aufgaben durcharbeiten. Im Laufe des Jahres, als wir dann zur textuellen Programmierung übergegangen sind, konnten wir Bezug auf code.org nehmen. Die Bausteine daraus waren noch allen ein Begriff. Ich finde den Einstieg mit code.org auf jeden Fall empfehlenswert, da innerhalb von wenigen Stunden und ohne großen Materialaufwand vermittelt wird, worum es beim Programmieren geht. Ich werde es immer wieder verwenden."*

Links:
Kursübersicht: studio.code.org
Hour of Code: code.org/learn

Scratch

Scratch ist eine grafische Programmiersprache und wurde vom MIT-Media Lab entwickelt. Befehlsblöcke werden ganz einfach wie Puzzleteile per Drag & Drop aneinandergereiht.



Durch diese Blöcke können ganz leicht Bewegungen, Klänge, aber auch Schleifen und Verzweigungen verwendet werden. Kinder und Jugendliche können mithilfe dieser Blöcke verschiedene Spiele und Geschichten erstellen. Aufgrund der vom MIT zur Verfügung gestellten Figuren- und Hintergrundbibliothek kann der Fantasie freien Lauf gelassen werden. So kann eine Unterwasserwelt mit Fischen und einem gefährlichen Hai genauso erstellt werden wie ein Schloss, in dem ein Geist spukt, oder eine Ballerina, die auf einer Bühne tanzt. Durch diese Freiheiten in der Figur- und Hintergrundauswahl kann an die persönlichen Interessen der Kinder und Jugendliche angeknüpft werden und der positive Einstieg in die Programmierung verstärkt werden. Der Einsatz von Scratch eignet sich ab der 2. Schulstufe, grundlegende Rechen- und Lesekenntnisse sollten vorhanden sein. Für interessierte jüngere Kinder gibt es ScratchJr, das speziell für den Einsatz mit Tablets entwickelt wurde.

Der große Vorteil von Scratch ist, dass die Software kostenlos entweder zur Verwendung im Browser oder zur Installation auf verschiedenen Betriebssystem zur Verfügung steht. So können die Kinder und Jugendlichen nicht nur in der Schule eigene Spiele und Geschichten erstellen, sondern auch zuhause weiter daran arbeiten. Anna Relle Stieger (acodemy): „Durch das Programmieren mit Scratch lernen Jugendliche und Kinder am Computer selbst Inhalte zu produzieren statt nur zu konsumieren. Sie haben dabei sehr schnell erste Erfolgserlebnisse und verwenden ganz selbstverständlich das, was sie in der Schule lernen."

Links:
Scratch: scratch.mit.edu/projects/editor/
Materialien Scratch: lehrerweb.wien/medienbildung/coding-macht-spass/
ScratchJr: scratchjr.org

CodeCombat

CodeCombat ist ein Browserspiel, in dem Kinder und Jugendliche programmieren lernen können. Mithilfe von

Python- bzw. JavaScript-Code wird eine Spielfigur durch die Abenteuerwelt gesteuert. Nur so lassen sich Gegner attackieren, Hindernisse überwinden und Kristalle einsammeln. Das Absolvieren unterschiedlicher Level und Welten ermöglicht den Zugang zu neuen Gegenständen, die wiederum neue Methoden freischalten. So ermöglicht z.B. ein Schwert mithilfe der Methode attack() Gegner anzugreifen, eine Brille mit findNearestEnemy() den nächstgelegenen Gegner zu finden und ein magisches Buch das Verwenden von Schleifen und Verzweigungen. Je weiter man im Spiel voranschreitet, desto komplexer werden die Aufgaben und desto wichtiger werden auch eigene Lösungsansätze.

Da CodeCombat eine textuelle Programmiersprache verwendet, eignet sich der Einsatz in der Schule vor allem ab der Sekundarstufe I. Der spielerische Ansatz hilft es SchülerInnen zu motivieren und durch den langsam ansteigenden Schwierigkeitsgrad werden auch komplexe Konstrukte schnell verstanden. Durch das Anlegen eines Accounts können Fortschritte jederzeit gespeichert und wieder aufgerufen werden. Die Unterteilung der Welten in verschiedene Level ermöglicht rasch Erfolge, da die Aufgaben kurz gehalten sind. Insgesamt gibt es aktuell fünf Welten mit verschiedenen Schwerpunkten wie etwa Basic Syntax, if-Verzweigung, boolesche Logik, Funktionen und Arrays. Aufgrund der großen und gut zusammenarbeitenden Community sind sehr viele der eigentlich englischen Aufgaben ins Deutsche übersetzt.

Die Grundversion von CodeCombat ist kostenlos, möchte man als LehrerIn jedoch ganze Klassen verwalten, fallen Gebühren an.

Elisabeth Weißenböck: *„Der Einsatz im Rahmen der Begabtenakademie NÖ hat mir gezeigt, dass Jugendliche mithilfe von CodeCombat sehr schnell erste Programmierkonzepte erlernen. Sie haben Spaß indem sie etwas, das sie sowieso tagtäglich tun – nämlich Spiele spielen – mit Programmierung und Problemlösen verbinden."*

Links:
codecombat.com
Guide für LehrerInnen: codecombat.com/teachers/courses (in Englisch, Registrierung erforderlich)

Bee-Bot

Beim Bee-Bot handelt es sich um einen einfach, robust und kindgerecht gestalteten Bodenroboter. Programmiert wird er über Tasten auf seinem Rücken. Möglich sind Bewegungen um 15 cm nach vorne oder zurück sowie Drehungen um jeweils 90 Grad nach links oder rechts.

Die Bienenroboter eignen sich ausgezeichnet für einen spielerischen Einstieg in die Welt der Informationstechnologie und Steuerung. Sie schulen neben der räumlichen Wahrnehmung vor allem auch die Sequenzierungsfähigkeit. Darüber hinaus bietet die Arbeit mit den verschiedenen Spielfeldern ein breites Feld, um dies mit der Förderung der Lese- und Rechenfähigkeit zu verknüpfen.

Altersmäßig zielt der Bee-Bot auf Kinder zwischen 4 und 8 Jahren ab. Jedes Kind sollte unbedingt mit einer eigenen „Biene" arbeiten können. Der Einarbeitungsaufwand zur Beherrschung des Bienenroboters ist mit rund 30 Minuten selbst für Personen ohne jeglichen technischen Background sehr gering.

Die Kids lernen anfangs durch Rollenspiele die Grundlagen kennen. Danach geht es mit der Roboterprogrammierung, dem einfachen Vorwärts und Rückwärts los. Dazu sitzen alle hinter einer Linie und erhalten vom Lehrenden die auszuführenden Programmierbefehle bevor sie nach dem Startkommando alle gleichzeitig das Programm ablaufen lassen. Weiter geht es mit dem isolierten Erlernen der Drehbewegungen. Sind die Grundlagen halbwegs gefestigt, folgen kompliziertere Bewegungsaufgaben. Danach bietet sich ein Stationsbetrieb an, bei dem die Kinder gruppenweise verschiedene Spiele spielen. Auch Aufgabenstellungen, bei denen die Kinder zur Lösung kollaborieren müssen,

sind denkbar. Mit dem größten Sortiment an Bee-Bots wartet der Hersteller TTS Group auf, wobei ein Bot rund 80 Euro kostet. Zusätzlich zu den Robotern sollte man für den raschen Einstieg auch einige Spielraster anschaffen. Schon recht bald wird man sich aber eigene Spielfelder basteln.

Pia Brüner (IFIT): *„Ich finde es immer wieder faszinierend, wie sich durch das Bee-Bot Konzept bei den Kindern innerhalb weniger Stunden eine völlig neue räumliche Vorstellungskraft entwickelt. Noch faszinierender ist, wieviel Freude die Kinder am Umgang mit den Bienenrobotern haben. Am Ende der intensiven und lehrreichen Workshops höre ich immer wieder von den kleinen Teilnehmern: „Es war so schön, dass wir heute den ganzen Tag ‚nur' gespielt haben. Können die Bienen morgen wiederkommen?"*

Link: beebot.ibach.at

Lego Mindstorms EV3

Bei Lego Mindstorms EV3 handelt es sich um die letzte Generation der Mindstorms-Baukästen. Sie beinhalten neben unzähligen Lego Technic-Bauteilen einen Mikrocontroller, an den sich verschiedene Aktoren und Sensoren anschließen lassen. Die Kinder und Jugendlichen können damit unterschiedlichste Robotermodelle konstruieren und anschließend das Verhalten der Roboter über spezielle Computerprogramme programmieren.

Für die verschiedenen Altersgruppen stehen unterschiedliche Umgebungen und Sprachen zur Verfügung. So kann in der Primarstufe und Sekundarstufe I eine grafische Programmierumgebung eingesetzt werden, während in der Sekundarstufe II auf eine objektorientierte Programmiersprache wie Java zurückgegriffen wird.

Erste Kursaktivitäten sind ab der 3. Schulstufe möglich. Nach oben hin gibt es keine Grenze, selbst Studierende lassen sich damit noch begeistern. Idealerweise dauert eine Einheit mindestens 4 Schulstunden, weniger als 2 am Stück sollten es auf keinen Fall sein. Ein mehrtägiger Projektunterricht stellt die beste Variante dar. Von der Gesamtlänge sind Kurse von vier bis insgesamt 40 Schulstunden denkbar. Das Interessante an Mindstorms ist, dass es sich auch wunderbar für interdisziplinären Unterricht nutzen lässt. Vom Betreuungsverhältnis her sind maximal acht Zweierteams pro

TrainerIn ideal, mit einer Gruppengröße von bis zu 24 Kindern lässt sich im Worst-Case-Fall noch arbeiten. Jedes Zweierteam sollte einen eigenen Baukasten zur Verfügung haben. Für einen ersten Überblick reicht als Einarbeitungszeit ein Nachmittag. Um eigene Kurse sattelfest durchführen zu können, sollte man aber schon einige Projekte bau- und programmiermäßig umsetzen. Da gehen gleich einige Nachmittage drauf. Diverse Ausbildungsangebote der Pädagogischen Hochschulen vereinfachen den Einstieg.

Mit der ikonischen EV3 Software lassen sich so gut wie alle wesentlichen Programmierkonzepte einführen. Von den Nutzenden ist kein Programmcode zu schreiben, sondern das Programm wird mittels entsprechender Mausaktionen erstellt. Durch das Verschieben und Verbinden der einzelnen Funktionsblöcke entsteht nach und nach das gewünschte Programm. Sind Texteingaben vonnöten, erfolgen diese dialogbasiert. Neben klassischen Kontrollstrukturen wie Schleifen und Verzweigungen stehen für komplexere Aufgaben auch Variablen und Datenleitungen zur Verfügung. Zusätzlich lässt sich auch auf bestimmte Ereignisse warten und reagieren. Das Erarbeiten der jeweiligen Konstrukte sollte stets aufgabengesteuert erfolgen.

Die Kosten für einen Baukasten belaufen sich auf rund 350 Euro für die Home Edition und rund 430 Euro für die Education Edition. Für den Schulbetrieb muss nicht notwendigerweise die Education Edition angeschafft werden, aufgrund des inkludierten Akkus empfiehlt sich das aber. Die offizielle Programmierumgebung kann man sich kostenlos von der zugehörigen Webseite herunterladen, sämtliche für die Java-Programmierung erforderlichen Softwareprodukte sind ebenfalls frei verfügbar. Nicht zu unterschätzen ist in Verbindung mit Lego Mindstorms EV3 der Aufwand, um die jeweiligen Baukasteninhalte zusammenzuhalten. Nur allzu gerne tauschen die SchülerInnen nämlich Teile während der Kurse untereinander aus.

Klaus Marterbauer (IFIT): *„Meiner Erfahrung nach sind so gut wie alle SchülerInnen von Lego Mindstorms EV3 begeistert. Sei es die Kreativität beim Bauen und Programmieren, das Erreichen von (realistischen) Zielen oder schlicht das Ausprobieren von etwas Neuem. Gerade beim Benützen der Software ist es interessant zu beobachten, wie die Lernenden immer mutiger werden und die Angst vor Fehlern ablegen.“*

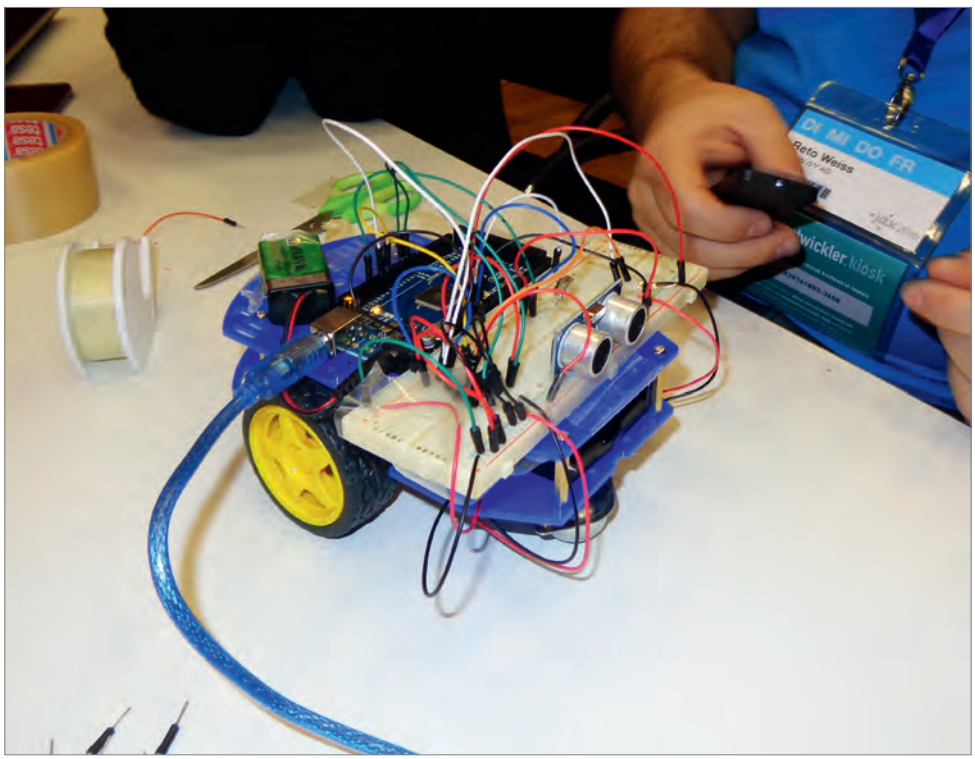
Link: mindstorms.lego.com

Arduino

Bei Arduino handelt es sich um eine sehr populäre Physical Computing-Plattform, für die es unzählige Aktoren und Sensoren gibt. Sowohl die Hardware als auch die Software sind im Sinne von Open Source quelloffen. Die Programmierung erfolgt auf Basis von C/C++.

Dank Arduino wurde die Welt der Mikrocontroller für jedermann zugänglich und einfach nutzbar gemacht. Interessierte können damit unterschiedlichste Elektronikprojekte umsetzen. Zum Bau eigener Roboter wird die Plattform ebenfalls gerne verwendet. Das Besondere an Arduino ist, dass sich zusätzlich zur Programmierung auch die elektrotechnischen und elektronischen Grundlagen anschaulich vermitteln lassen.

Altersgruppenmäßig ist ein Einstieg ab der 6. Schulstufe sinnvoll. Niveaumäßig lässt sich die Komplexität beliebig nach oben skalieren – wer will, kann auch direkt den Mikrocontroller per Registerzuweisungen programmieren und sich so eine eigene API schreiben. Der Einarbeitungsaufwand ist im Vergleich zur Lego-Technologie schon deutlich größer. Das hat damit zu tun, dass Arduino-Programme auf textueller Basis erstellt werden. Es gibt im Web aber jede Menge Tutorials, die ein schnelles Loslegen ermöglichen.



Das klassische Einstiegsbeispiel ist die am Board integrierte LED zum Blinken zu bringen. Danach folgen oftmals Ampelsteuerungen, wobei hier dann bereits verschiedenfarbige LEDs mit passenden Vorwiderständen auf dem Steckbrett aufgebaut werden müssen. Anschließend kann man sich an das Erforschen der beigepackten digitalen oder analogen Sensoren machen. Dank der bereitgestellten Bibliotheken ist auch das Ansteuern von Servos einfach möglich. Mithilfe von Spezialhardware, sogenannten Shields, lassen sich auch klassische Motoren mit überschaubarem Aufwand ansteuern. Es bieten sich kleinere Projekte, wie bspw. die Realisierung einer Alarmanlage, nach den Basisaufgaben an.

Arduino-Lernsets für die Schule gibt es um rund 60 Euro. Für ungefähr 15 Euro bekommt man auch noch eine mobile Roboterplattform dazu. Die zur Programmierung erforderliche Arduino IDE steht auf der offiziellen Arduino-Website kostenlos zum Download bereit.

Dietmar Bodner (VMS Nenzing): *„Wir wagten heuer erstmals den Versuch mit Arduino in die Textprogrammierung einzusteigen und ich muss zugeben, dass ich selbst sehr überrascht war, wie begeistert die SchülerInnen von dieser Art des Programmierens sind. Gleichzeitig ist es notwendig sich mit der Funktionsweise und dem Einsatz verschiedener elektronischer Bauteile zu beschäftigen, was einen gänzlich neuen und stark motivierenden (da authentischen) Zugang zum Thema Elektronik ermöglicht."*

Link: arduino.cc

NAO

Beim NAO handelt es sich um den fortschrittlichsten humanoiden Roboter für den Ausbildungsbereich. Er hat 25 Motoren, zwei hochauflösende Videokameras sowie WLAN an Bord und erfasst seine Umgebung mittels verschiedenster Sensoren. Neben Sprach-, Gesichts- und Objekterkennung kann der NAO-Roboter auch mehrsprachig kommunizieren. Die Programmierung ist in unterschiedlichen Umgebungen und Sprachen möglich.

Nach nur wenigen Unterrichtseinheiten können auch Kinder und Jugendliche einen großen Teil des Roboterpotenzials nutzen. Bereits für Kinder der 4. Schulstufe lassen sich NAO-Workshops gestalten. Auch hier kann der Lehrende die Komplexität beliebig nach oben skalieren. Für Einstiegskurse sollten zwei Stunden als Untergrenze vorgesehen werden, nach oben hin gibt es hier keine Grenze. Von der Gruppengröße her sollte man beachten, dass je drei Zweierteams mindestens ein NAO-Roboter zur Verfügung steht, damit die Kinder und Jugendlichen ihre Programme auch ausreichend ausprobieren können. Das macht einen großen Teil der Faszination aus, selbst wenn sich viele Funktionalitäten auch mithilfe des in der Software integrierten virtuellen Roboters testen lassen. Um selbst Kurse leiten zu können, ist wohl mindestens ein eintägiges Training vonnöten. Zur Festigung des Erlernten sollte man noch einige Nachmittage zum Üben einplanen. Wer tiefer in die Materie eindringen möchte, muss aber deutlich mehr Zeit aufwenden.

Bildquelle: shutterstock [DeymosHR]

NAO hat 25 Motoren, zwei Videokameras und WLAN an Bord.

Am einfachsten ist die Programmierung mittels Choregraphe. In dieser grafischen Programmierumgebung erstellt man die Verhaltensmuster zur Robotersteuerung in klassischer Flussdiagramm-Manier.

Aktuell ist der NAO um rund 7.000 Euro bei offiziellen NAO-Distributoren erhältlich. Für Österreich ist dies das deutsche Unternehmen noDNA. Die Programmierumgebung Choregraphe kann man sich nach Registrierung bei SoftBank Robotics als 90-tägige, voll funktionsfähige Testversion herunterladen. Für Kursaktivitäten reicht diese üblicherweise aus.

Dietmar Bodner (VMS Nenzing): *„Wo auch immer wir mit unseren beiden NAOs auftauchen, sind die Besucher sofort in den Bann der sympathisch wirkenden Humanoiden gezogen. Selbstverständlich freuen sich unsere Mädchen und Buben darauf, unseren Robotern Lucky und Sani Leben einhauchen zu dürfen. Die graphische Programmieroberfläche ermöglicht es mit einfachen Mitteln sehr ansprechende Programme zu schreiben und wer tiefer in die Kunst des Programmierens einsteigen will, wird wohl nicht so schnell an die Grenzen dessen stoßen, was mit den niedlichen Robotern alles möglich ist.“*

Link: ald.softbankrobotics.com/en/cool-robots/nao

Wie so oft im Leben führen auch bei der Vermittlung der Programmierung sowie der informatischen Konzepte viele Wege ans Ziel. Gleichgültig, für welchen Ansatz Sie sich als Lehrende/r letztendlich entscheiden, wenn Ihre SchülerInnen Freude beim Erlernen neuer Inhalte im Unterricht empfinden, stehen die Chancen um ein Vielfaches besser, dass sich das Erlernte auch tatsächlich in den Schülerköpfen verankert.

Viel Erfolg und Happy Coding!

AutorInnen

Elisabeth Weißenböck

ist für die Österreichische Computer Gesellschaft (OCG), die HCI-Group der TU Wien und die Begabtenakademie NÖ tätig.
E-Mail: elisabeth.weissenboeck@ocg.at

Bettina Gruber

ist Programmierlehrerin an der HTL Spengergasse und hat unter anderem mehrere Workshops für Programmieranfänger unterschiedlicher Altersstufen abgehalten.
E-Mail: bettina.gruber@gmx.net

Bernhard Löwenstein

ist Gründer und Obmann des Instituts zur Förderung des IT-Nachwuchses (IFIT). Sein gemeinnütziger Verein hat sich in den letzten Jahren mit mehr als 200 Technologie-Workshops pro Jahr (von Schulkursen über Wochenendkursen bis hin zu Ferienprogrammen) zur größten aktiven MINT-Förderorganisation in Österreich entwickelt.
E-Mail: b.loewenstein@gmx.at

Das Tor zur „realen Welt“: Physical Computing mit dem Raspberry Pi

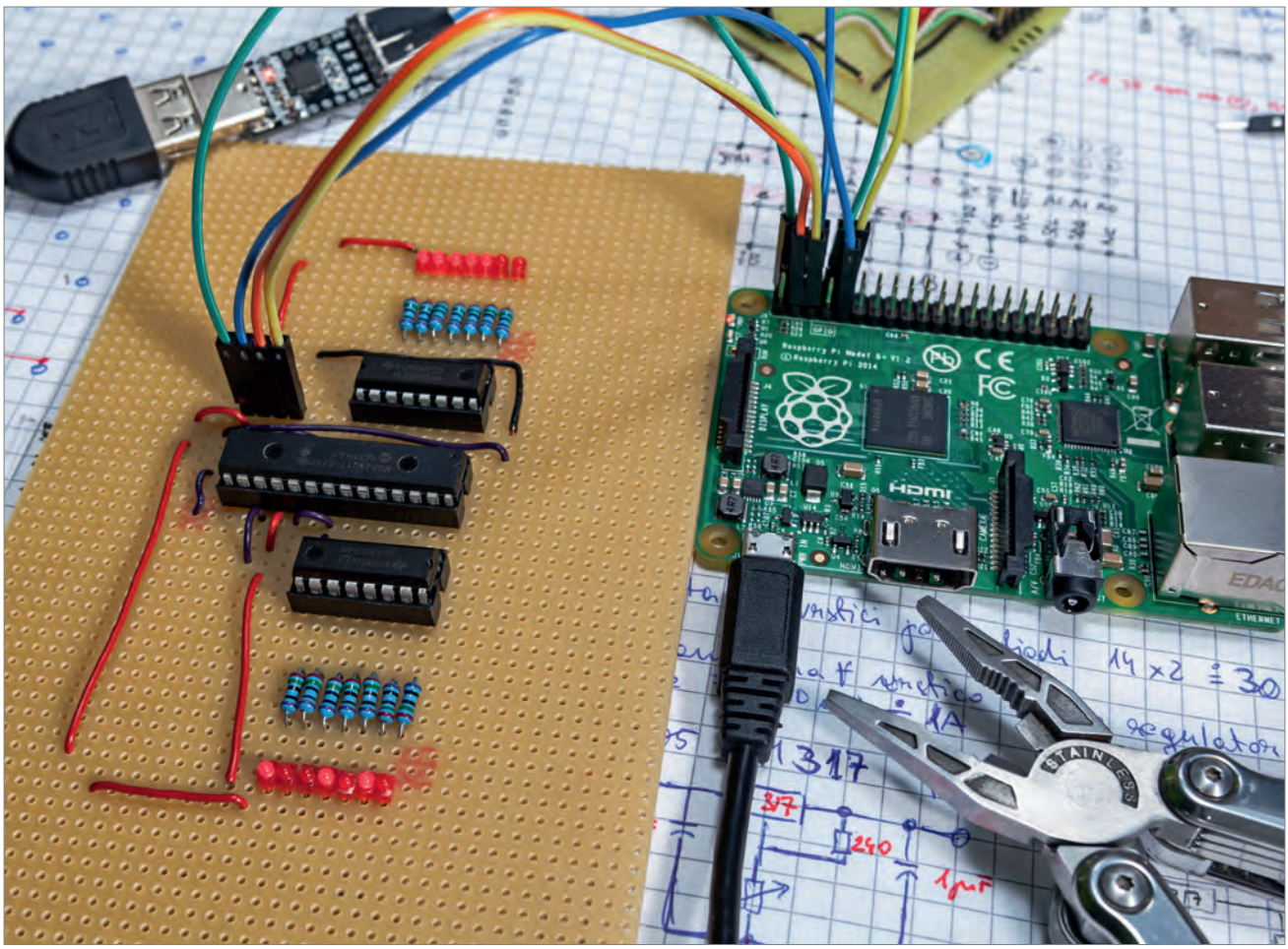
Nach der konstruktionistischen Lerntheorie von Seymour Papert lernen Kinder und Jugendliche besonders gut, wenn sie dabei selbst etwas schaffen oder zusammenbauen können.

Für die informatische Bildung bot der Einsatz von Educational Robotic Bausätzen lange Zeit den Hauptweg dazu. Neue Wege eröffneten sich durch die Entwicklung der Arduino-Plattform, die aus einem quelloffenen Mikrocontroller-Board und Software besteht. Dadurch wurde Physical Computing auch im Schulbereich populärer. Allgemein versteht man unter „Physical Computing“ Systeme, die sich mit der Beziehung zwischen Menschen und der digitalen Umwelt befassen, wobei mittels Hard- und Software interaktive, physische Systeme geschaffen werden (Wikipedia).

Mit der Markteinführung des Raspberry Pi im Jahre 2012 ergaben sich weitere neue Möglichkeiten. Dieser extrem kostengünstige Minicomputer ist mit einer GPIO Schnittstelle (General Purpose Input/Output) ausgestattet. Dabei sind verschiedene Pins vom Programmierer selbst als Ein- oder Ausgabe definierbar. Damit steht die Verbindung zur Außenwelt offen. Im Unterschied zum Arduino handelt es sich beim Raspberry Pi aber um einen vollständigen Computer, Betriebssystem und Daten befinden sich auf einer SD-Karte, was die Wartung sehr erleichtert.

Obwohl in letzter Zeit auch einige Alternativen zum Raspberry Pi auf den Markt gekommen sind, besticht dieser besonders durch seine unerreichte Entwickler- und Anwender-Community. Schon jetzt wurden über acht Millionen Stück des Einplatinencomputers verkauft. Sogar Microsoft unterstützt den Raspberry Pi ab der Version 2 mit dem Betriebssystem Windows 10 IoT Core (developer.microsoft.com/en-us/windows/iot).

In der neuesten Version des Betriebssystems Raspian mit der Bezeichnung „Jessie“ sind viele interessante Programme bereits vorinstalliert, wie das gesamte Office-Paket Libre Office und die Bildverarbeitungssoftware



Bildquelle: shutterstock [goodcat]

Raspberry Pi Single-Board-Computer, der mit einer selbstgebauten Schaltung mit einigen LEDs verbunden ist.

GIMP. Mit dem Epiphany-Webbrowser und dem E-Mail Client Claws wären sogar alle Voraussetzungen gegeben, um mit dem Raspi den Europäischen Computer Führerschein (ECDL) zu machen.

Zusätzlich wird auch eine spezielle Version der Mathematiksoftware Wolfram Mathematica kostenlos zur Verfügung gestellt. Dazu kommt die Programmiersprache Wolfram Language, mit der auch Cloud-Operationen programmiert werden können (Siehe: www.wolfram.com/cloud/).

Neben der bereits erwähnten Wolfram Language sind folgende Programmiersprachen vorinstalliert: die kindergerechte visuelle Programmiersprache Scratch, Blue Java IDE, Greenfoot Java IDE zur Entwicklung zweidimensionaler graphischer Animationen, Python in der Version 2 und 3, sowie die Musikprogrammiersprache Sonic Pi. Natürlich könnte man auf dem Raspberry Pi auch in der Sprache C, C++ oder sogar Assembler programmieren.

Eine weitere Neuerung ist Node-RED, ein von IBM entwickeltes Visualisierungstool für das Internet der Dinge (nodered.org). Knoten („nodes“) werden als Icons dargestellt und können per Drag & Drop zu Anwendungen zusammengefügt werden. Steht ein Raspberry Pi zur Verfügung, können diese Knoten z.B. auch Sensoren sein, die über die GPIO-Schnittstelle eingebunden werden. Eine erste einfache Anwendung könnte dann die Temperatur eines Raumes twittern.

Interessant sind auch die vorinstallierten Spiele-Möglichkeiten. Mit Python-Games stehen Progammierbeispiele in Python zur Verfügung. Minecraft Pi, eine spezielle Version des beliebten Spiels Minecraft, wird kostenlos angeboten. Mit der Programmiersprache Python kann in Minecraft „hineinprogrammiert“ werden und so können interessante Mods (Modifikationen) selbst geschaffen werden.

Der Raspberry Pi bietet günstige Einstiegsmöglichkeiten in den Educational Robotics-Bereich. Laufend werden neue Modelle angeboten wie z.B. der ZeroBorg (www.piborg.org/zeroborg), der auch mit dem Rasperry Pi Zero (www.raspberrypi.org/products/pi-zero/), kompatibel ist.

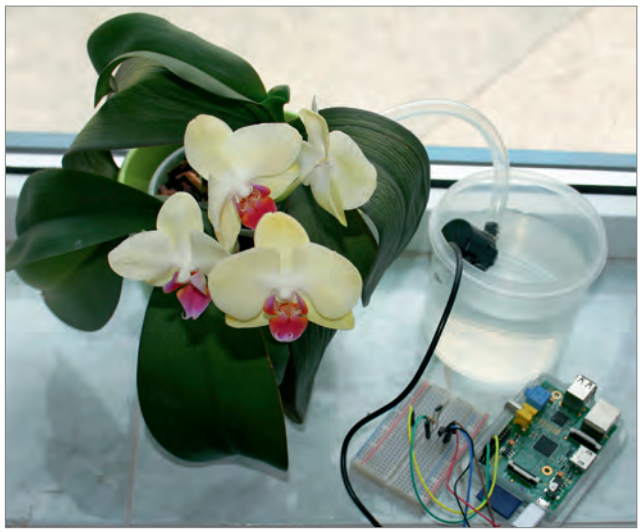
Es geht allerdings noch einfacher und günstiger. Mit einer H-Brücke für nur wenige Euros können ferngesteuerte Autos aus Pappkarton selbst gebastelt werden (siehe dazu die Anleitungen von Ingmar Stapel (custom-build-robots.com/).

Der rechtzeitige Umstieg von einer visuellen Programmiersprache wie Scratch zu einer textbasierten Programmiersprache wie Python ist für SchülerInnen sehr wichtig. Das wird im neuen Lehrplan Großbritanniens vorbildlich umgesetzt, um danach auch erweiterte informatische Konzepte vermitteln zu können. In der Schweiz wurde dazu speziell für den Unterricht mit TigerJython (www.tigerjython.ch/) eine Plattform entwickelt, die entsprechende Lehrmittel für einen sinnvollen Einstieg in die Python-Programmierung bereitstellt. TigerJython ist für den Einsatz ab der vierten oder fünften Schulstufe konzipiert.

In Großbritannien wurde mit dem Astro Pi-Projekt ein Schulprojekt in Kooperation mit der internationalen Raumstation ISS durchgeführt (Siehe www.ocg.at/de/ocg-journal-12016).

Dafür wurde mit dem Sense HAT ein eigenes Aufsteckboard für den Raspberry Pi entwickelt, das bereits mehrere Sensoren wie Temperatur-, Luftdruck- und Feuchtigkeitssensor eingebaut hat, aber auch Sensoren zur Raumorientierung wie Beschleunigungssensor, Gyroskop und Magnetometer, sowie ein Display aus 8x8 Leuchtdioden zur Informationsdarstellung.

Sense HAT ist auf dem Markt erhältlich und bietet einen kompakten und günstigen Einstieg für die Erfassung von Umweltdaten.



„Lazy Planting“: Bewässerungssteuerung mit dem Raspberry Pi Mit einer kleinen Schaltung, bestehend aus Transistor, Widerstand und Diode wird eine Tauchpumpe angesteuert. (Foto: Johann Stockinger)

Wie schon erwähnt, können über die GPIO Pins des Raspi Sensoren und Aktoren direkt angesteuert werden. Einfacherer und sicherer ist es jedoch, man verwendet eines der auf dem Markt erhältlichen Aufsteckboards. Mit GrovePi (www.dexterindustries.com/grovepi/) steht ein solches Board zur Verfügung, das den Anschluss über ein standardisiertes Steckersystem bereitstellt. Zahlreiche für den Arduino hergestellte Sensoren sind damit auch direkt an den Raspi anschließbar. Für jeden der Sensoren

werden Programmbibliotheken und Beispielcodes in mehreren Programmiersprachen bereitgestellt.

Andere neue interessante Zugänge zur Welt des Internets der Dinge sind z. B. das Wio-Link Kit (www.seeedstudio.com/wiki/Wio_Link) oder Googles neue App Science Journal (makingscience.withgoogle.com/science-journal).

Wio-Link kommt ohne Raspberry Pi aus, kostengünstige Boards werden direkt an das Smartphone oder Tablet angeschlossen. Sensoren und Aktoren können wieder über das Grove-Steckersystem angeschlossen werden.

In Googles Science Journal werden die in den Smartphones eingebauten Sensoren zur Messung benutzt und die Werte abgespeichert und grafisch dargestellt. Erste Drittanbieter stellen dafür bereits Hardware-Erweiterungskits bereit.

Eine weitere Initiative startete 2015 die britische Rundfunkanstalt BBC mit der Experimentierplatine Micro Bit, mit der Kindern und Jugendlichen Coding und die Konstruktion einfacher Schaltkreise näher gebracht werden sollen. Der BBC Micro Bit war ursprünglich nur zur Verteilung an britische SchülerInnen gedacht, ist aber seit Kurzem für ca. 15 Euro auch auf dem Markt erhältlich (www.microbit.co.uk/).

Für den Raspberry Pi steht inzwischen eine große Menge an Weiterbildungsmaterial zur Verfügung. Neben zahlreichen Büchern werden eigene Zeitschriften angeboten. The MagPi, das offizielle Pi-Magazin, kann in der digitalen Version kostenlos herunterladen werden (www.raspberrypi.org/magpi/) und enthält neben grundlegenden Informationen Berichte über teils skurrile Anwendungen wie z.B. eine automatische Katzenfütterungsanlage.

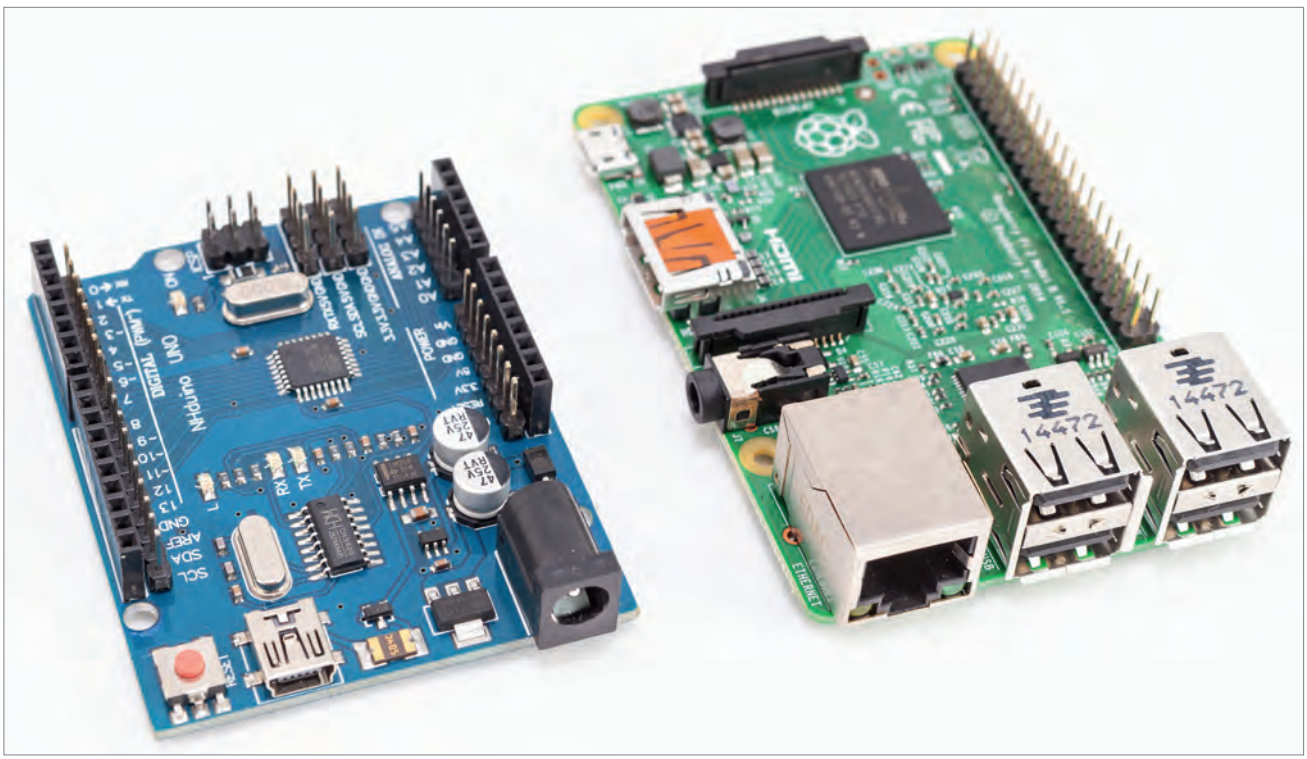
Das Hasso Plattner Institut in Deutschland hatte auch erstmals auf seiner MOOC-Plattform OpenHpi (www.raspberrypi.org/magpi/) einen Einführungskurs zum Thema „Embedded Smart Home“ angeboten, der auf dem Raspberry Pi aufbaut.

Mit der „Thinkering Academy“ hat die OCG eine neue Initiative gestartet, bei der gemeinsam mit Technischen WerklehrerInnen sowie anderen interessierten LehrerInnen Konzepte für einen fächer- und schulstufenübergreifenden Unterricht erarbeitet werden (siehe: www.coding4you.at).

Autor

Johann Stockinger

Dr. Johann Stockinger ist in der Österreichischen Computer Gesellschaft in den Bereichen Bildung, Innovation und Forschung tätig. Für die TU Wien arbeitet er am Erasmus+ Projekt „Head in the Clouds: Digital Learning to Overcome School Failure“ mit.
E-Mail: johann.stockinger@ocg.at



Ein Arduino-kompatibler Microcontroller und ein Raspberry Pi von der Größe einer Kreditkarte.

Easy Coding mit Raspbotics

„Für die Zukunft sind Computer mit weniger als 1,5 Tonnen Gewicht vorstellbar." Es waren Redakteure des US-Technik-Magazins „Popular Mechanics", die im Jahre 1949 diese wenig kühne Prognose wagten. Sie lagen mit ihrer Vorhersage ziemlich richtig, aber ob Sie so weit gegangen wären? Denn ein halbes Jahrhundert später sind Tablets wenige hundert Gramm schwer, Smartphones passen in die Hosentasche, und beide Systeme verfügen über Prozessorgeschwindigkeiten, Speicherkapazitäten und weitere Performances, die den 1949er Computer ziemlich alt aussehen lassen. Der war verglichen mit heutigem Gerät ein Monster, nahm im Gebäude komplette Räume in Beschlag, aber er war kein Geheimniskrämer. Denn egal ob Mac oder Monster, beide arbeiten mit den Zuständen „1" und „0". Und der 1949er Computerjahrgang bedurfte richtiger Programmierarbeit, wo wir heute einfach ein paar Fenster öffnen.

E-Mails senden, nach einer Antwort googlen, große Datenmengen in der Cloud verwalten und stets erreichbar sein: Jung und Alt nutzen die aktuellsten Apps und die neueste Software mit einer Selbstverständlichkeit und Sicherheit, als hätten sie in dreizehn Jahren Schule nichts anderes gelernt. Wollen Eltern mit den Kiddies mithalten, müssen sie ganz schön auf Zack sein. Bei aller Performance jedoch sind wir heute nur noch Anwender, aber keine Programmierer mehr.

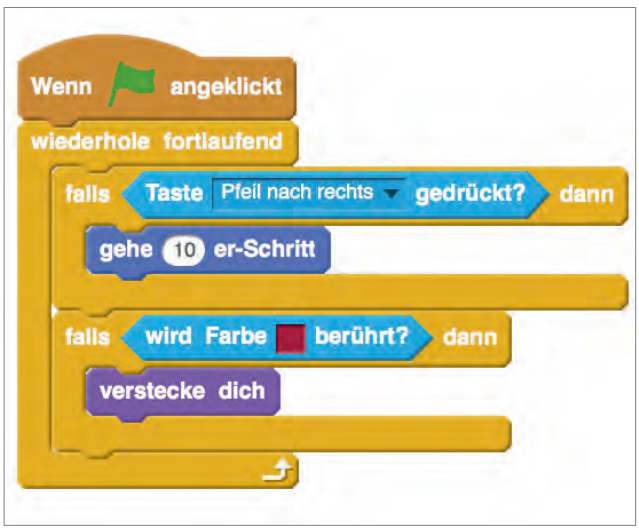
Dabei ist das Schreiben eines Programms eine faszinierende Entdeckung, die im Umgang mit dem PC ganz neue Perspektiven aufzeigt. Und Programmierenlernen mit Raspbotics macht darüber hinaus auch eine Menge Spaß. Durch intuitives Bedienen von grafischen Programmblöcken können selbst Programmieranfänger eine Verbindung zur physischen Außenwelt herstellen. Somit steht der Umsetzung kreativer Ideen nichts mehr im Wege.

„Everybody in this country should learn how to program a computer because it teaches you how to think." Steve Jobs, einer der Mitgründer von Apple, meinte, dass jeder lernen sollte, einen Computer zu Programmieren – denn das lehrt einem das Denken.

Genau diesen Ansatz verfolgt Raspbotics. Spielerisch und intuitiv können Kinder und Jugendliche mit Hilfe von Controllern, Sensoren und jeglicher elektronischer Peripherie in die Welt der Technik eintauchen. Ganz gleich ob es um die Entwicklung von eigenen Spielen geht, um Prozessautomatisierung oder das Ansteuern von Robotern – mit Raspbotics wird der Unterricht zur spannenden Lernstunde mit zahlreichen interessanten Ansätzen, um sich Schritt für Schritt vom Anwender zum Programmierer zu entwickeln.

Raspbotics ist besonders für Einsteiger geeignet, denn Ausgangspunkt ist die kindgerechte Programmierumgebung SCRATCH, mit der sämtliche Raspbotics-Boards programmiert werden können. Selbst Kindern und Jugendlichen ohne Vorkenntnisse eröffnet sich die Möglichkeit, erste spannende Programme zu schreiben.

In die Programmierumgebung werden vorgegebene Anweisungsblöcke wie z.B. „falls-dann" oder „warte 1 Sekunde" einfach hineingezogen, ohne dass man sich über die benötigte Syntax wie die richtige Klammerwahl, Groß-und Kleinschreibung usw. Gedanken machen muss. Mit Grafiken, Klängen und der Verwendung von Sensoren lassen sich dadurch innerhalb kürzester Zeit kreative Ideen am PC umsetzen. Die Software steht sowohl online als auch offline für alle Plattformen für einen Gratisdownload zur Verfügung.



Mit Scratch programmierbar (https://scratch.mit.edu/)

Selbstverständlich können aber auch Fortgeschrittene mit einer konventionellen Programmiersprache, wie z.B. Python viele Boards ansprechen, sodass ein Umstieg in weiterführende Schulen mit Programmierschwerpunkt bedeutend erleichtert wird. Einige Boards lassen sich sogar aus der Arduino-Programmierumgebung heraus programmieren, sodass mit einem einzigen System ein breites Anwendungsspektrum abgedeckt wird.

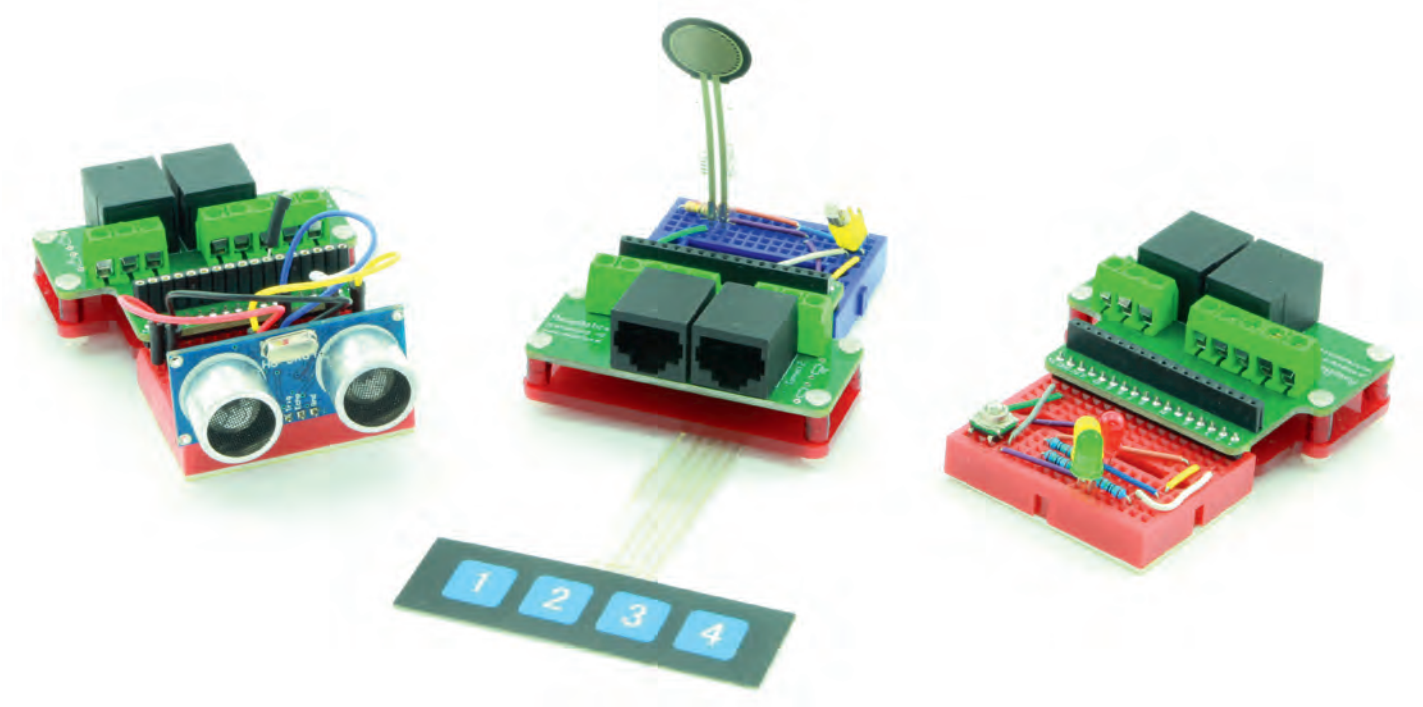
Die unterschiedlichen Raspbotics- Boards verfügen über Taster, Joysticks, Displays und zahlreiche Sensoren. Damit können Temperatur, Licht, Druck sowie Farben und Distanzen exakt gemessen werden. Schon ist der Abstandswarner, der beim Einparken so wunderbare Dienste leistet, programmiert. Wäre dies eine Anwendung für den Physikunterricht, so lassen sich für nahezu alle Unterrichtsfächer alltagsnahe und kreative Projekte kinderleicht umsetzen.

Oder sollten wir besser sagen lehrerleicht? Denn für alle Boards stehen Aufgaben und Lösungswege als Videotutorial auf www.raspbotics.at zur Verfügung. Diese Tutorials sind nach Schwierigkeitsgrad gruppiert, sodass für alle Schulstufen Aufgaben und Lösungen vorhanden sind. Somit wird auch LehrerInnen, deren SchülerInnen in Sachen Anwendung längst enteilt sind, eine Hilfestellung zu ersten Programmiererfahrungen geboten. Raspbotics-Workshops ermöglichen KollegInnen, erste Eindrücke zu sammeln und gegebenenfalls Unsicherheiten zu beseitigen.

Auch mit dem Protoboard lassen sich spannende Unterrichtsstunden quasi aus dem Ärmel schütteln. Über ein Steckboard besteht die Möglichkeit, einfache Schaltungen aufzubauen – schon ist der Sachunterricht, Physik- oder Elektronikunterricht gerettet. Individuelle und vor allem kostengünstige Anwendungen mit LEDs, Taster und Sensoren garantieren einen hoch interessanten Unterricht, der die vollständige Induktion zur halben Sache mutieren lässt.

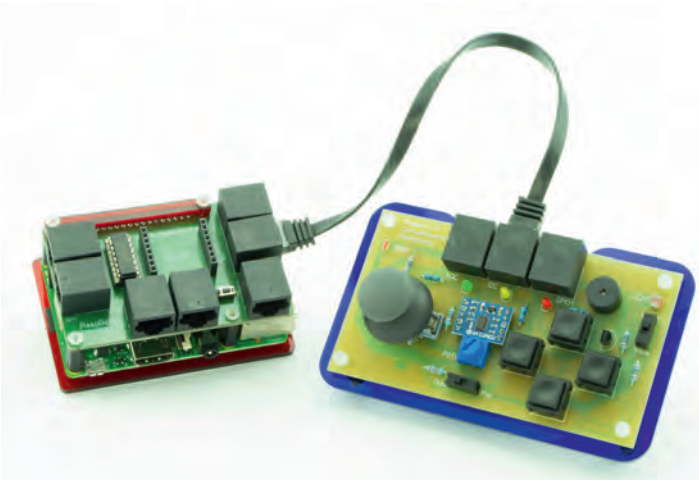
Die ersten Schritte mit Scratch können schon VolksschülerInnen machen. Man glaubt gar nicht, was in den Kiddies steckt, wie schnell sie spielerisch in die Programmiersprache hineinfinden. Im Hauptschulalter kann mit motivierten SchülerInnen auch mit einer konventionellen Programmiersprache begonnen werden. Ich selbst programmiere mit meinen 14jährigen SchülerInnen Mikrocontroller in C und nach bereits einem halben Jahr sind sie imstande, Taster und Leuchtdioden zu bedienen bzw. Werte aus diversen Sensoren auszulesen und zu verarbeiten. Allein die Tatsache, dass der Nachwuchs sich mit Begeisterung ans Programmieren macht und dafür auf Zeit vor dem Fernsehschirm verzichtet, bestätigt, dass wir mit Raspbotics genau den richtigen Mix aus Spiel und geistiger Herausforderung gefunden haben. Viele Experten sind sich einig, dass Programmieren schon im Volksschulalter erlernt werden kann, da sich Coding ähnlich verhält wie Lesen, Schreiben und Rechnen.

Und wer früh mit den unterschiedlichen Komponenten des Raspbotics-Systems in Berührung kommt,



profitiert davon im Alltag. Das Erkennen und Formulieren eines Gesamtproblems, dessen Zerlegen in Teilaufgaben sowie die zielorientierte Recherche sind Fähigkeiten, die während der Ausbildung und im Berufsleben immer wieder gefordert werden. Das Testen von Lösungswegen, die durchaus auch mal als Sackgasse enden, erhöht die Frustrationsgrenze, das heißt bei Fehlschlägen werden Wege raus aus der Sackgasse gesucht.

Die Komponenten Raspberry Pi + Robotics sind allerdings so kompakt, dass sie nur schwer zu treffen sind. Und sie geben den Nutzern auch keinen Grund dazu. Voraussetzung für dieses System ist ein Raspberry Pi (ab Version Pi2), auf dem das Raspbotics Basisboard einfach aufgesteckt wird. Je nach Bedarf können anschließend alle weiteren Raspbotics-Boards durch einfaches Verbinden verwendet werden.



„Hardware nennt man die Teile eines Computers, die man treten kann." (Jeff Pesis)

Die Kombination mit dem 40 Euro teuren Raspberry Pi bietet sich besonders für Schulen an, da nicht immer ein Computersaal zur Verfügung steht. Da sich das Betriebssystem und die zusätzlichen Programme auf einer SD-Karte befinden, lässt sich ein Klonen der Software für die gesamte Klasse mit einfachem Kopieren der SD-Karte bewerkstelligen. Die Software für den Raspberry Pi steht gratis zum Download bereit. Ach ja, und das Raspbotics-System mit allen verfügbaren Komponenten und Zubehör wiegt gerade mal 300 Gramm, oder anders ausgedrückt 0,0003 Tonnen.

Nähere Informationen unter: www.raspbotics.at

Autor

Claus Zöchling

Claus Zöchling, 45 Jahre, Lehrer an der PTS18.

Aktuell: Abhaltung von Workshops (Programmieren von Mikrocontrollern, Mechatronik). Momentan berufsbegleitendes Elektronik-Studium am Technikum Wien.

Ziel: Mit Raspbotics vielen Kindern und Jugendlichen einen raschen und spannenden Einstieg in die Welt der Technik und des Programmierens zu ermöglichen.

E-Mail: zoechling@matheonline.com

Spieleprogrammierung

Vorwort

Computer wurden dazu erschaffen, um so aufregende Dinge wie Raketenflugbahnen und Mondlandungen zu berechnen oder Geheimcodes zu knacken. Verwendet wird der durchschnittliche Computer für so schnöde Dinge wie Textverarbeitung.

Glücklicherweise kann Informatikunterricht aufregender sein als das Auswendiglernen von Office-Software-Klickstrecken.

Computer wurden zwar genau so wenig für Computerspiele erfunden wie für Textverarbeitung, werden aber genau dazu – zum Spielen – gerne verwendet.

Um ein kleines Computerspiel zu schreiben, braucht man weder großartige Programmierkenntnisse noch teure 3D-Engines. Die Fähigkeit, buchstabengetreu abzuschreiben, etwas Fantasie und ein gewisses Interesse am Genre der Roguelikes-Spiele reicht durchaus aus, um einen Workshop oder eine (Doppel)Stunde zum Thema abzuhalten.

Bei meiner Arbeit mit meinen eigenen Programmierkursen und bei Workshops in Schulen (http://spielend-programmieren.at/blog/20151116_schul-workshop.html) ist mir aufgefallen, dass gerade jüngere Kinder nicht unbedingt mit bunten Farben, Musik und Soundeffekten zum Programmieren „gelockt" werden müssen… einen Buchstaben (traditionell das @-Zeichen) durch ein ein- oder gar zwei-dimensionales Spielfeld aus farblosen Satzzeichen zu bewegen, bietet Aufregung genug…

Erfahrungen

Folgende Erfahrungen habe ich beim Unterrichten mit Python und Roguelikes gemacht, mit Kindern zwischen 10 und 13 Jahren:

- Kinder brauchen nicht unbedingt Farben, Grafik- und Soundeffekte, sondern können sich sehr gut in eine nur von Textzeichen dargestellte Spielwelt „hineinfühlen".
- Kinder ab 10 Jahren können Programmcode abtippen, den sie noch nicht verstehen.
- Jüngere Kinder können Programmcode ändern, den sie noch nicht verstehen.
- Schwierigkeiten beim Abtippen, speziell von Python-Programmen, sind:
  - Einrückungen links einhalten
  - Funktion des Zeilenwechsels
  - Unterschied zwischen Tabulatortaste und Leertaste
  - Unterschied zwischen Entfernen- und Rücklöschtaste
  - Unterschied zwischen Überschreib- und Einfügemodus
  - Erzeugung von Sonderzeichen mit Hilfe der Alt-Gr-Taste: eckige und geschweifte Klammern, Unterstriche, Backslash
  - Verstehen der englischen Fehlermeldungen des Python-Interpreters
- Programmieren lernen bedeutet den Umgang mit der Tastatur zu lernen.
- Kinder verbringen sehr viel Zeit mit dem Gestalten von Labyrinthen im Textmodus, sobald sie das Spielprinzip eines Roguelikes (Bewegung, Monster) verstanden haben.
- Kinder lernen sehr schnell einzelne Teile des Python-Programms zu modifizieren (z.B. Bewegungstasten), ohne dabei den Rest des Programmes zu zerstören.
- Kinder entwickeln sehr schnell eigene Verbesserungswünsche (Heiltränke, Magie, Monster), um das Rogulike anderen, ihnen bekannten Fantasy-Spielen, anzupassen.
- Der Lernerfolg steigt enorm, wenn ich als Kursleiter anstatt erklärend zu unterrichten („Das ist ein String, das ist eine Schleife") eine passivere Rolle einnehme und warte, bis die Kinder ein weiteres „Feature" (z.B. bewegliche Monster) einfordern und ihnen dann nur die dafür notwendigen Programmbefehle zeige. Das Verständnis dafür, was ein Python-Befehl bewirkt, wird wesentlich erleichtert, wenn der Python-Befehl sichtbar ein „Feature Request" des Schülers erfüllt.
- Viel Programmierwissen kann implizit vermittelt werden, durch die Aufforderung, kleine Änderungen an einem Programm vorzunehmen (z.B. `dx += 2` anstatt `dx += 1`) und die Auswirkungen auf das Spiel zu beachten.

Roguelikes

Rogue (englisch für „Schurke, Räuber") ist ein Computerspiel, welches bereits im Computer-Mittelalter (1980) auf Uni-Rechnern gespielt wurde. Da die damaligen Computer im Wesentlichen nur Text in einem Terminal darstellen konnten, wurde genau dies zum Spielen verwendet: Man sieht den Spieler von oben durch ein Labyrinth (Dungeon) wandern, wobei er Schätze und Nahrung aufsammelt, Gegner bekämpft, Fallen ausweicht und Stiegen findet, um noch tiefere, noch gefährlichere Dungeons zu erforschen.

Moderne Computerspiele wie z.B. Diablo sind Nachfahren von Rogue und werden als Roguelikes

(https://en.wikipedia.org/wiki/Diablo_(video_game)) oder "Dungeon Crawler" bezeichnet.

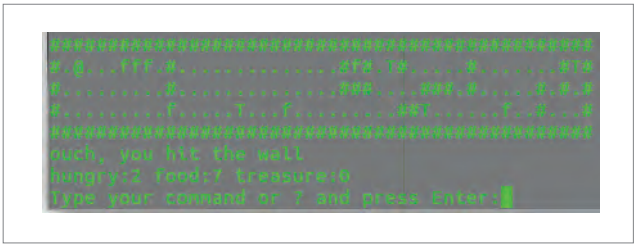
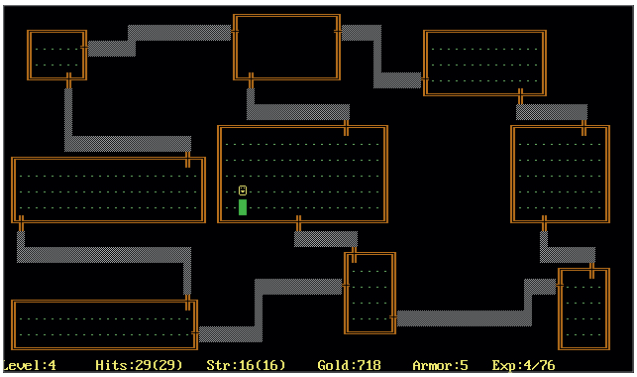


Abb.: "Der Spieler (@) bewegt sich in einem 2-dimensionalem Labyrinth und sammelt Essen (f) und Schätze (T)"



Python

Ich verwende gerne die Programmiersprache Python (Version 3.5), welche als freie Software (https://www.gnu.org/philosophy/free-software-for-freedom.de.html) für Linux, Mac und Windows unter http://python.org zum Download bereitsteht. Am RaspberryPi und auf vielen Linux-Computern ist schon Python in Version 2 vorinstalliert, ich empfehle aber Python ab Version 3 zu benutzen: Dadurch lassen sich Sonderzeichen und Umlaute problemlos darstellen. Das abgebildete Python-Programm funktioniert notfalls auch mit Python Version 2, sofern man in Zeile 24 den Befehl **`input`** durch **`raw_input`** ersetzt .Falls das Installieren von Python im eigenen PC-Saal undurchführbar ist, lässt sich das Beispiel auch online ausführen auf Webseiten, welche Python-im-Browser (https://repl.it/languages/python3) anbieten.

Arbeitsablauf

Bei meinen eigenen Workshops handle ich stets nach der Devise „erst abtippen, Erklärungen nur nach Bedarf". Die Minimalvariante besteht darin, den abgebildeten Python-Code abzutippen und entweder auf einer Python-im-Browser-Webseite oder in einem Programmeditor (Bei Python mitinstalliert wird der Editor Idle, welcher im interaktiven Modus (Shell) startet. Dieser Modus ist nur für winzige (einzeilige) Programme gedacht, deshalb mittels „File -> New File" ein echtes Programmierfenster öffnen. Das Programm startet man mittels „Run -> Run Module". Davor wird man aufgefordert, das Programm abzuspeichern, wobei die Dateiendung ".py" verwendet werden muss, z.B. „textrogue.py". Im Internet findet man eine große Auswahl von Python-Editoren.) auszuführen und zu spielen. Der Spieler (dargestellt als @) wird mit den aus Ego-Shootern vertrauten Tasten **`w`**, **`a`**, **`s`** und **`d`** bewegt, wobei nach jeder Befehlseingabe die Enter-Taste gedrückt werden muss; „help" zeigt einen Hilfstext an, der alle Befehle erklärt.

Auch ohne Programmierkenntnisse ist es ab diesem Zeitpunkt möglich, die Spielfigur zu ändern (Zeile 9) und das Labyrinth (Zeile 3 bis 7) zu erweitern oder zu verändern. Zu beachten ist dabei, dass die Umfassungsmauer (#) lückenlos den äußeren Rand bilden muss.

```
1  # legend: #=rock .=floor f=food  T=treasure
2  dungeon = """
3  ##############################################
4  #.....fff.#..............#f#.T#.....#.......#T#
5  #.........#..............###....###.#....#.#.#
6  #.........f.....T...f.........##T.....f..#...#
7  ##############################################
8  """  # add more line to the dungeon!
9  player, msg = "@", "welcome @, move with w,a,s,d"
10 px, py, dx, dy = 1, 1, 0, 0
11 lines = dungeon.split()
12 length = len(lines[1])
13 hunger, treasure, food = 0, 0, 7
14 prompt = "Type your command or ? and press Enter:"
15 while hunger < 100:
16     for y, line in enumerate(lines):
17         # y is the line number starting with 0
18         if y == py:
19             print(line[0:px]+player+line[px+1:])
20         else:
21             print(line)
22     s = "hungry:{} food:{} treasure:{}\n".format(
23          hunger, food, treasure)
24     command = input(msg+"\n"+s+prompt)
25     dx, dy, msg = 0, 0, ""
26     if command == "help" or command == "?":
27         msg = "movement: w,a,s,d\n"
28         msg += "eat: e\nquit: exit or quit or q"
29     if command in ["quit", "exit", "q"]:
30         break # exit the game
31     if command == "e": # eat
32         if food > 0:
33             msg = "you eat food"
34             food -=1
35             hunger -= 11
36         else:
37             msg = "You have no food!"
38     hunger += 1 # getting more hungry
39     if command == "a":
40         dx = -1  # go left
41     elif command == "d":
42         dx = 1   # go right
43     elif command == "w":
44         dy = -1  # go up
45     elif command == "s":
46         dy = 1   # go down
47     target = lines[py+dy][px+dx]
48     if target == "#":
49         dx, dy = 0, 0 # running into wall
50         msg = "ouch, you hit the wall"
51     if target in ["f","T"]: # food or treasure
52         lines[py+dy] = lines[py+dy][:px+dx]+ \
53                        "."+lines[py+dy][px+dx+1:]
54         if target == "f":
55             msg = "you found food!"
56             food += 1
57         if target == "T":
58             msg = "you found treasure!"
59             treasure += 1
60     px += dx #  movement x
61     py += dy #  movement y
62 print("Game Over")
```

Dies ist ein guter Zeitpunkt, um die Bedeutung der „Einfg"-Taste (Insert) zu erklären, welche in den meisten Code-Editoren zwischen dem Einfügen- und dem Überschreiben-Modus wechselt.

Für weitergehende Änderungswünsche sind minimale Programmierkenntnisse erforderlich. Ich gehe im weiteren Artikel davon aus, dass solche vorhanden sind, z.B. durch Beschäftigung mit Scratch, PocketCode oder ähnlichen Projekten.

Python-Besonderheiten

Wer erstmals mit Python arbeitet, muss aufpassen: Nicht nur ist die Sprache case-sensitive (Groß- und Kleinschreibung spielt eine Rolle), es ist auch sehr wichtig, wie weit eine Zeile links eingerückt ist und ob dazu Leerzeichen oder die Tabulatortaste verwendet wurde. Prinzipiell signalisiert eine Einrückung (die nur nach einer Zeile, welche mit einem Doppelpunkt endet) einen Programmblock. Zum Beispiel einen Block, der nur nach einer bestimmten Bedingung ausgeführt oder öfter wiederholt wird. In anderen Sprachen werden diese Blöcke oft durch geschweifte Klammern eingeschlossen, bei visuellen Programmiersprachen wie z.B. Scratch werden die Blöcke von anderen, Schraubstock-förmigen Blöcken, umrahmt.

Um überlange Zeilen umzubrechen, gibt es die Möglichkeit, die Zeile mit einem \ zu beenden und in der nächsten Zeile (mit beliebiger Einrückung) fortzusetzten. Das ist beim Programmbeispiel in Zeile 52 und 53 der Fall, die eigentlich eine einzige, überlange Zeile bilden und nur aus Layoutgründen geteilt wurden. Lange Zeilen darf man auch aufteilen, sofern man sich innerhalb von Klammern befindet (Zeile 22 und 23). Die Raute (#) beginnt einen Kommentar. Strings (Zeichenketten) können durch einfache (') oder doppelte (") Anführungszeichen gekennzeichnet werden. Werden drei Anführungszeichen verwendet (Triple-Quotes), so darf der String mehrere Zeilen lang sein.

Schleifen, Verzweigung, Variablen

Wie in jeder prozeduralen Programmiersprache, gibt es auch in Python Schleifen (**`for, while, break`**), Verzweigungen (**`if, elif, else`**) und Variablen. Die Zuweisung eines Wertes zu einer Variablen erfolgt durch das =-Zeichen, wobei es möglich ist, gleich mehreren Variablen in einer einzigen Zeile Werte zuzuweisen (Zeile 13):

```
hunger, treasure, food = 0, 0, 7
```

Obige Zeile bedeutet dass die Variable "hunger" den Startwert 0, bekommt, ebenso die Variable "treasure", wohingegen die Variable "food" den Wert 7 bekommt (damit der Spieler nicht sofort verhungert). Wer hier den Wert für "hunger" auf 90 stellt, den Wert für "food" auf 0 und die Nahrungspakete (f) im Dungeon (Zeilen 3 bis 7) strategisch knapp verteilt, verwandelt das Roguelike in ein Denkspiel. Wie viel Hunger die Spielfigur aushält und wie sehr Essen sättigt, steht in Zeile 15 bzw. 35.

Weitere Änderungsvorschläge

Gewünscht wird von meinen Kursteilnehmern oft eine Erweiterung des (Bei)spiels um Fallen und Monster. Nichts einfacher als das, zumindest solange die Monster (vorerst) stationär sind: In den Dungeon-String (Zeile 3-7) werden ein paar Statuen eingebaut (S). Läuft der Spieler in diese hinein, zersplittern sie und der Spieler verliert Lebenspunkte (hitpoints). Dazu muss der Spieler erst einmal hitpoints haben.

Wichtig: Im Folgenden bitte immer die Original-Einrückung unverändert lassen!

Wir verändern Zeile 10 und setzen den Wert der Variable **`php`** (player hit points) auf 10:

```
px, py, dx, dy, php = 1, 1, 0, 0, 10
```

Die hitpoints müssen auch angezeigt werden, in Zeile 22 und 23 (hier zusammengefasst in eine überlange Zeile). **`s = „hungry:{} food:{} treasure:{} hitpoints:{}\n".format(hunger, food, treasure, php)`**

Und schlussendlich brauchen wir eine Kollisionserkennung. Diese gibt es schon, wir ergänzen Zeile 51:

```
if target in [„f","T","S"]:
```

Jetzt müssen wir nur noch den Hiptointverlust einbauen. Dazu am Anfang der Zeile 60 ein paarmal auf Enter drücken (Zeile 60 und folgende wandern weiter hinunter) und diesen Codeblock in die jetzt leere Zeile 60 einbauen:

```
if target == „S":
  msg = „you destroy a statue and loose
  hitpoints"
  php -= 5
```

Die Einrückung muss dabei genauso sein wie bei den beiden vorherigen if-Blöcken (Zeile 54 und 57).

Schlussendlich soll das Spiel aufhören, wenn der Spieler zu wenig hitpoints hat. Wir verändern Zeile 15:

```
while hunger < 100 and php >0:
```

Fertig!

Weiterführend bietet es sich an, eine Funktion mit Hilfe des Random-Moduls zu schreiben, um zufällige Schadenswerte erzeugen zu können bzw. bewegliche Monster zu bauen, wozu eine Beschäftigung mit Klassen (objektorientierte Programmierung) hilfreich wäre.

Um ein kleines Computerspiel zu schreiben, braucht man weder großartige Programmierkenntnisse noch teure 3D-Engines.

Spielfigur und Spielbrett

Den meisten Schülern meiner Erfahrung nach relativ egal (sofern der Code funktioniert) ist der Teil, welcher die Spielfigur darstellt. Das Labyrinth oder Spielbrett wird in der Variable dungeon (Zeile 2) gespeichert, dessen Wert ist ein mehrzeiliger String (Zeichenkette), erkennbar an den dreifachen Anführungszeichen (Triple-Quotes) am Anfang und am Ende. In Python ist jeder Buchstabe in einem String indiziert, das heißt, er hat eine Nummer. Die Zählung beginnt verwirrenderweise bei 0. Angenommen, ich habe einen ein-dimensionalen Dungeon (einen langen Gang), bestehend aus drei Bodenplatten (Punkten), einem Nahrungspaket (f), einem Schatz (T) und einer Mauer (#):

```
dungeon = '...fT#'
```

Das erste Zeichen im String (die erste Bodenplatte) hat den Index 0, das Nahrungspaket (das 4. Zeichen) hat den Index 3, der Schatz hat den Index 4 und die Mauer den Index 5.

Möchte ich in diesen Dungeon einen Spieler (@) zeichnen, der auf der dritten Bodenplatte (links vom Nahrungspaket) steht, überdeckt der Spieler auf Index 2 die Bodenplatte. Da sich der Spieler später weiterbewegt, verändere ich nicht direkt den Wert der Variable Dungeon, sondern sage Python, es soll den Dungeon-Teil links vom Spieler (die ersten beiden Bodenplatten, Index 0 bis 1) darstellen, dann den Spieler selbst (auf Index 2) und danach den Dungeon-Teil rechts vom Spieler (Index 3 bis 5).

Die Position vom Spieler speichere ich in der Variablen px (x-position of player) und gebe ihr den Wert 2. Der Python-Code würde so ausschauen:

```
dungeon = '...fT#'
px = 2
print(dungeon[:px]+"@"+dungeon[
px+1:])
```

Das Ergebnis:

```
..@fT#
```

Python kann einen Teilstring anzeigen, indem man nach dem Namen des Strings in eckigen Klammern den Index schreibt (sofern man nur ein einziges Zeichen schreiben will) oder in die eckigen Klammern den Startindex, einen Doppelpunkt und den Stopp-Index schreibt. Der Stopp-Index ist dabei jenes Zeichen, das nicht mehr gedruckt werden soll. Wird die Zahl links vom Doppelpunkt weggelassen, bedeutet das „vom ersten Zeichen an", wird die Zahl rechts vom Doppelpunkt weggelassen, bedeutet das „bis zum letzten Zeichen".


Bildquelle: shutterstock [welcomia]

Der print-Befehl sorgt für die Ausgabe am Bildschirm. Obiges Beispiel bedeutet: "Drucke die Zeichenkette, welche in der Variablen dungeon gespeichert ist, vom ersten (Index 0) Zeichen an bis zum zweiten Zeichen (Index 1), dann drucke einen Klammeraffen (@), dann drucke vom 4. Zeichen (2+1=3) bis inklusive zum letzten Zeichen. Im Beispiel-Programm passiert genau das in Zeile 19, allerdings nur wenn in der gerade dargestellten Zeile (mit Index y) sich auch der Spieler befindet (Variable py). Ansonsten wird die dungeon-Zeile so gedruckt wie sie ist, ohne verdeckende Spielfigur (Zeile 21).

Sammelt die Spielfigur Gegenstände (Nahrung, Schätze) aus dem Labyrinth auf, muss der Wert der Variablen dungeon dauerhaft verändert werden.

Zur Erklärung: der mehrzeilige dungeon wird in eine Liste einzelner Zeilen zerlegt, welche in der Variablen *lines* gespeichert werden (Zeile 11).

Diese Zeilen haben natürlich einen Index, wiederum beginnend bei 0. lines[0] ist die erste Zeile, lines[1] die zweite Zeile usw.

Um zum Beispiel ein aufgesammeltes Nahrungspaket durch eine Bodenplatte zu ersetzten, muss die Variable dungeon (genauer gesagt: die entsprechende Linie in der Liste lines) verändert werden.

Dies geschieht, indem man ein 2-dimensionales Array anspricht, mit zwei eckigen Klammerpaaren hintereinander.

Die erste Dimension bildet die Linie in der verändert werden soll (y). Die zweite Dimension ist das Zeichen innerhalb dieser Linie (x).

Dies geschieht in dem Augenblick im Programm, bevor der Spieler die neue Position erreicht hat (er könnte ja auch in eine Mauer laufen). Seine Laufrichtung wird in den Variablen dx und dy (für delta-x, delta-y) gespeichert. Seine zukünftige Position ergibt sich aus px (wo er gerade steht) und dx (Laufrichtung) bzw. py + dy.

Zuerst wird die Linie gesucht, in der sich der Spieler befinden wird:

```
lines[py+dy]
```

Diese Linie ist ein großer Textstring. Sie wird zerlegt in den Teil links vom Spieler [:px+dx] und in den Teil rechts vom Spieler [px+dx+1:]. Dazwischen kommt das Bodenplattenzeichen des Dungeons, ein Punkt. Die komplette Zeile (52+53, hier zusammengefasst) lautet:

```
lines[py+dy] = lines[py+dy][:px+dx]+
„.“+lines[py+dy][px+dx+1:]
```

Was bedeutet das enumerate in Zeile 16?

```
for y, line in enumerate(lines):
```

Dies ist ein Trick, der es erspart eine Variable y zu definieren, auf 0 zu setzten und nach jeder Linie von *lines* um eins zu erhöhen. **`enumerate(lines)`** numeriert die Linien (natürlich pythonesk mit 0 beginnend) für uns durch. Die for Schleife „Schleift" über die Variable y und die dazugehörende *line* von *lines* gleichermaßen.

Einen ausführlicheren Blogartikel sowie Quellcode, Links und Bilder finden Sie unter http://spielend-programmieren.at/blog/20160718_python.html

Autor

Horst Jens

Horst Jens ist Gründer der Firma spielend-programmieren und bietet seit 2006 im Raum Wien Programmierkurse für Kinder und Jugendliche an sowie Workshops für Erwachsene und in Schulen.

Er verwendet gerne die Programmiersprache Python und interessiert sich vor allem für freie Software (free/libre/open-source) im Bereich Programmiersprachen und Spieleprogrammierung.

Homepage, Kontakt:
http://spielend-programmieren.at/de:kontakt
E-Mail: horstjens@gmail.com

Pocket Code – freier Online-Kurs für Kinder

Die Anzahl der Smartphone-User ist in den letzten Jahren rasant gestiegen und ein Ende ist nicht in Sicht. Schon Studien von 2014 haben gezeigt, dass mehr oder weniger alle 10- bis 18-jährigen ein eigenes Smartphone besitzen (Feierabend et al, 2015) (Nagler et al, 2016). Selbst bei den 6- bis 7-jährigen wird der Griff zum Smartphone immer häufiger (Grimus & Ebner, 2014). Es werden Spiele gespielt, im Internet gesurft und Apps verwendet, doch wie ein Smartphone und diese täglich genutzten Apps funktionieren, tritt dabei zunehmend in den Hintergrund. Im Gegenteil ist Programmieren für viele Kinder und Jugendliche schwer durchschaubar, da es kaum in den Lehrplänen der Sekundarstufe verankert ist. Die Stärkung dieses MINT-Fachs wird in vielen Ländern bereits als zentral erachtet und umgesetzt. Insbesondere werden in den letzten Jahren in einigen Schulen bereits digitale Werkzeuge eingesetzt, um kreatives digitales Gestalten zu ermöglichen und zu unterstützen. Diese Aktivitäten werden im deutschsprachigen Raum als „Making" bezeichnet (von engl. „to make" für „machen"). Making sind Aktivitäten, bei denen jede/r selbst aktiv wird und ein Produkt, ggf. auch digital, entwickelt, adaptiert, gestaltet und produziert, und dabei (auch) digitale Technologien zum Einsatz kommen (Schön et al, 2014).

Um dabei Kindern und Jugendlichen erste Schritte bei der Programmierung zu zeigen bzw. erste Erfahrungen sammeln zu lassen, wurde an der TU Graz Pocket Code, eine gratis App des dort beheimateten Non-Profit Projektes "Catrobat", entwickelt. Basierende auf einem Lego-artigen Bausteine-Prinzip sollen typische Probleme wegfallen, welche in traditionellen Programmiersprachen auftreten (Abb. 1). Kinder können sich aufgrund dieser einfachen und visuellen Benutzeroberfläche voll und ganz auf die Umsetzung ihrer eigenen kreativen Ideen konzentrieren. Gerade dem oft geglaubtem Vorurteil „So eine App zu erstellen ist doch viel zu schwer für mich" soll so mit Pocket Code entgegnet werden.

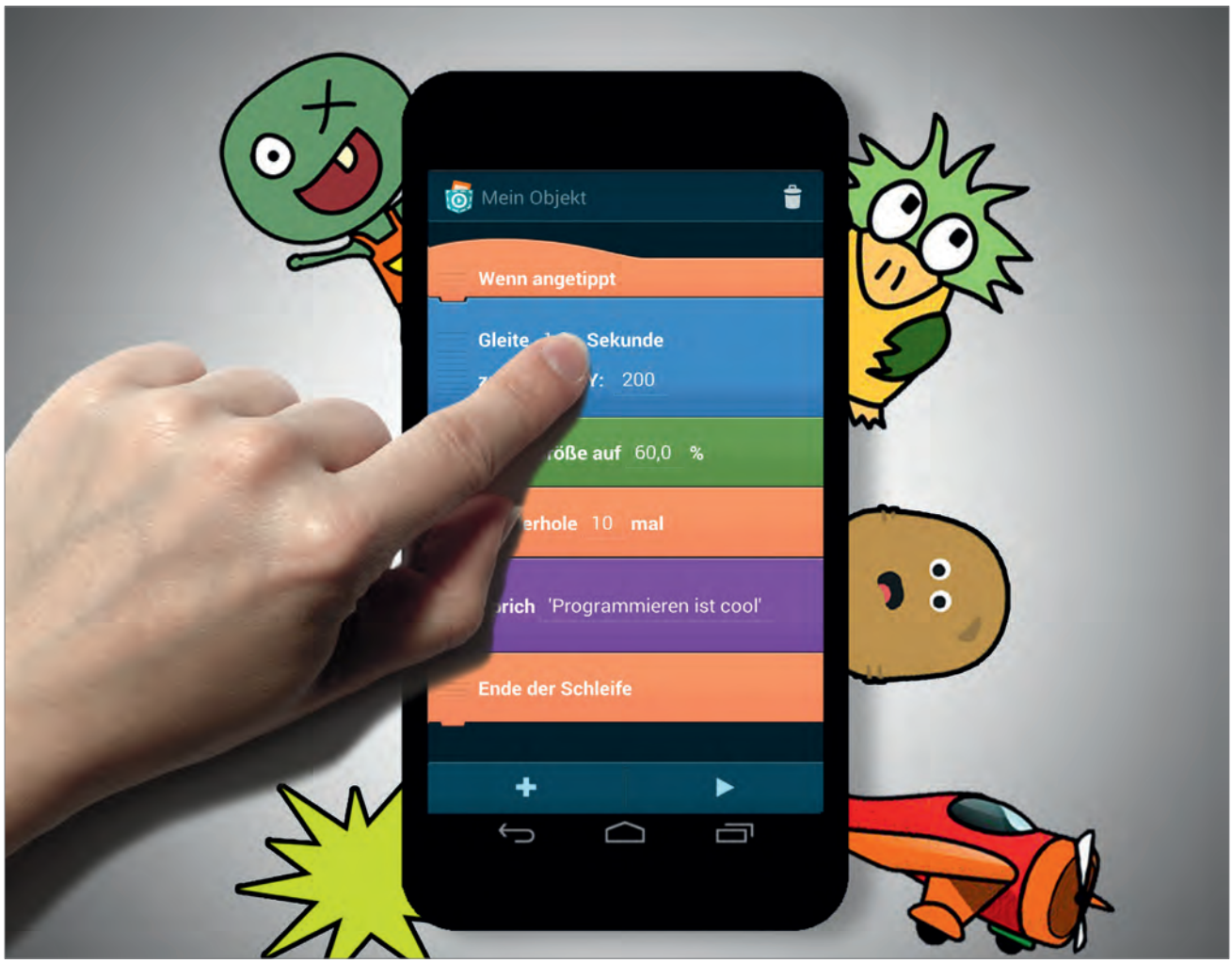


Abbildung 1: Lego-artiges Bausteine-Prinzip von Pocket Code

Um Kindern und Jugendlichen erste Schritte bei der Programmierung zu zeigen bzw. erste Erfahrungen sammeln zu lassen wurde an der TU Graz Pocket Code, eine gratis App des dort beheimateten Non-Profit Projektes "Catrobat", entwickelt.

Zusätzlich kann die selbsterstellte App anschließend ganz einfach auf der dazugehörigen Internetplattform www.pocketcode.org hochgeladen und so mit Freunden und anderen Nutzer/innen weltweit geteilt werden. Der Einblick in von anderen Usern erstellte Programme fördert das Erlernen dieser grafischen und intuitiven Programmiersprache. Mehr als 10.000 Apps wurden bis Juni 2016 hochgeladen, und die Tendenz ist ungebrochen weiter ansteigend.

Neben der Fähigkeit zu programmieren steht vor allem auch das logische und strukturierte Denken sowie ein Verständnis für täglich benutzte Technologien im Vordergrund, um für die zunehmend digitale Zukunft vorbereitet zu sein. In einigen Schulen in und außerhalb Europas wird Pocket Code daher bereits nicht nur im Informatikunterricht, sondern auch fächerübergreifend in anderen Unterrichtsfächern verwendet. Das Resultat sind von Kindern entwickelte interdisziplinäre Apps, wie zum Beispiel eine Weltkarte die beim Tippen auf ein Land dessen Namen in einer anderen Sprache verrät, oder eine Animation eines grafischen Beweises des Satzes von Pythagoras.

Doch wie kann man Kinder dazu bringen sich eigenständig mit Technologie und Programmieren zu beschäftigen? Eine Lösung bietet das eingangs erwähnte „Maker Movement", das derzeit einen großen Aufschwung erlebt. Das Aufkommen von FabLabs und Makerspaces, immer günstiger werdender Hardware, zunehmender Digitalisierung der Gesellschaft und freie Online-Materialien lassen einen vermehrten „Do It Yourself"-Gedanken zu. Diese Bewegung des „Einfach machen" ist mittlerweile auch in der Kinder- und Jugendarbeit angekommen. So wurde die Veranstaltung „Maker Days for Kids", die im letzten Jahr im oberbayrischen Bad Reichenhall stattgefunden hat, von rund 70 Kindern besucht (Schön et al, 2016a). Bei den Aktivitäten rund um 3D Druck, elektronische Bausätze, Roboter, virtuelle Realität, Lightpainting und vielem mehr zeigten die Kinder großes Interesse. Daraus entstand schlussendlich auch das erste deutschsprachige Handbuch (Schön et al, 2016b).

Das eigene Produkt am Ende in den Händen zu halten war für viele dabei ein besonderer Ansporn und führte zu strahlenden Gesichtern. Zusätzlich entsteht durch das vordergründige kreative Gestalten ein geschlechtsneutraler Zugang, welcher sich positiv auf die Stärkung von MINT-Fächern auswirkt. Das Endprodukt muss jedoch keineswegs immer „anfassbar" sein, so gab es im „DevLab" auch eine Station mit Laptops und Phablets, wo die Kinder eigene Programme mit der Programmiersprache Scratch programmieren konnten. „Making" bezieht sich somit nicht nur auf „Hardware", sondern hält auch im Software-Bereich Einzug. Wesentlich ist dabei nur, dass die Kinder frei nach ihren eigenen Vorstellungen Kreatives schaffen können.

Auch bei Pocket Code steht vor allem das eigene Tun und Machen im Vordergrund. Gerade die Möglichkeit eigene Apps direkt am Smartphone zu erstellen und auch zu teilen, kann ganz im Sinne des Making-Gedankens zu großer Motivation führen. Die Kinder werden in ihrer Lebenswelt berührt und können somit einen Bezug zum Programmieren aufbauen.

Darüber hinaus arbeitet das Team von Catrobat ständig daran, die App speziell auf die Bedürfnisse von Jugendlichen und Kinder anzupassen und weiter zu entwickeln. So wurden zum Beispiel eigene Bausteine entwickelt, welche die Steuerung eines Lego Mindstorms NXT Roboters ermöglichen. Weiterentwicklungen zur Steuerung einer Drohne, der Einbindung der bekannten Raspberry Pi, sowie eine eigene iOS Version sind bereits auf der Zielgeraden. Kinder sollen so von passiven zu aktiven Nutzerinnen und Nutzern werden.

Da dies mit Hilfe leichter geht, wird dazu im Herbst 2016 eine kostenloser Online-Kurs zum Thema „Learning to code – Programmieren mit Pocket Code" angeboten, dessen Zielgruppe Kinder im Alter von 10-14 Jahren sind. Dieser Kurs wird auf der MOOC-Plattform iMooX (http://imoox.at) zur Verfügung gestellt (Kopp & Ebner, 2015) und soll Kindern die Möglichkeit geben, erste kleine Apps selber zu gestalten. Darüber hinaus kann der Kurs auch von Lehrerinnen und Lehrern als Material für den Informatik bzw. Programmierunterricht herangezogen werden.

Der freie Online-Kurs wird über einen Zeitraum von sechs Wochen den Schülerinnen und Schülern das Programmieren mit Pocket Code näherbringen. In jeder Woche wird es zumindest ein Video mit theoretischem Inhalt geben, das die grundlegenden Funktionen sowie Tipps und Tricks von Pocket Code erläutert. Das Hauptaugenmerk des Kurses wird wieder auf das „making" – das eigene Tun und Machen gerichtet. Dazu werden unterschiedliche Aufgaben gestellt, welche die Kinder individuell bearbeiten und lösen. Im Hintergrund werden dabei Konzepte erarbeitet, die in der Informatik eine wichtige Rolle spielen, zum Beispiel Bedingungen, Variablen, Ereignisse oder Parallelismus. Am Ende des Prozesses steht wie bereits erwähnt eine eigene App.

Zusätzlich zu den erscheinenden Videos gibt es Begleitmaterialien in Form von „Pocket Karten" (Abb. 2), welche einen Einblick in die Funktionsweise der

Abbildung 2: Pocket Karte

verschiedenen Blöcke geben und somit als weitere Hilfe auch im Unterricht genutzt werden können. Der erfolgreiche Abschluss des Kurses wird schließlich mittels einer Bestätigung oder zertifizierten Badges bescheinigt.

Die Anmeldung zum Kurs (http://catrob.at/mooc) ist jederzeit möglich. Wir möchten ganz besonders Lehrerinnen und Lehrer dazu einladen sich anzumelden, um den Kurs an ihre Schülerinnen und Schüler weiterzuempfehlen. Falls Sie auch Interesse an Kooperationen haben, schreiben Sie uns bitte gerne an imoox@tugraz.at.

Kinder lernen Programmieren.

Literatur:

Feierabend, S., Plankenhorn, T., Rathgeb, T. (2015) JIM 2015. Jugend, Information, (Multi-) Media. Basisstudie zum Medienumgang 12- bis 19-Jähriger in Deutschland. Medienpädagogischer Forschungsverbund Südwest. http://www.mpfs.de/fileadmin/JIM-pdf15/JIM_2015.pdf (last access 2016-04-21)

Grimus, M. & Ebner, M. (2014) Learning with Mobile Devices Perceptions of Students and Teachers at Lower Secondary Schools in Austria. In Proceedings of World Conference on Educational Multimedia, Hypermedia and Telecommunications 2014 (S. 1600-1609). Chesapeake, VA: AACE

Kopp, M., Ebner, M. (2015) iMooX - Publikationen rund um das Pionierprojekt. Verlag Mayer. Weinitzen

Nagler, W., Ebner, M., Schön, M. (2015) R.I.P. E-Mail * 1965 - 2015. In Proceedings of World Conference on Educational Multimedia, Hypermedia and Telecommunications 2015. accepted, in print. Chesapeake, VA: AACE.

Schön, S., Ebner, M., Kumar, S. (2014) The Maker Movement. Implications of new digital gadgets, fabrication tools and spaces for creative learning and teaching. In: eLearning Papers, 39, Juli 2014, S. 14-25.

Schön, S., Ebner, M., & Reip, I. (2016a) Kreative digitale Arbeit mit Kindern in einer viertägigen offenen Werkstatt. Medienimpulse

Schön, S., Ebner, M., Narr, K. (2016b) Making-Aktivitäten mit Kindern und Jugendlichen: Handbuch zum kreativen digitalen Gestalten, Book on Demand, Norderstedt

Autoren

Stefan Janisch

Stefan Janisch hat den Master in Sportwissenschaft und absolviert derzeit sein Diplomstudium Informatik und Sport Lehramt auf der TU Graz bzw. KFU Graz.
E-Mail: stefan.janisch@student.tugraz.at

Martin Ebner

Univ.-Doz. Dr. Martin Ebner leitet die Organisationseinheit Lehr- und Lerntechnologien an der Technischen Universität Graz und zeichnet für alle E-Learning Belange der Universität verantwortlich. Als solches ist das Thema Making mit Kindern und Jugendlichen ein wichtiges Thema für ihn. Mehr können sie auf seinem Weblog nachlesen: elearningblog.tugraz.at
E-Mail: martin.ebner@tugraz.at

Wolfgang Slany

Univ.-Prof. Wolfgang Slany ist am Institut für Softwaretechnologie an der Technischen Universität Graz tätig und für Pocket Code verantwortlich, welches im Rahmen des Catrobat Projekts entwickelt wird.
www.ist.tugraz.at/wolfgang_slany
E-Mail: wolfgang.slany@tugraz.at

Mobiles Klassenzimmer - didaktisches Konzept

Beim mobilen Klassenzimmer bringen wir Kindern bei, wie sie Spiele und andere Apps, z.B. Animationen von Geschichten oder eigene Musikvideos, direkt auf ihren Handys selbst erstellen können. Dabei verwenden wir eine intuitiv verständliche, Lego-artige visuelle Programmiersprache, die in Graz seit 2010 in Zusammenarbeit mit dem Scratch Team des MIT Media Labs sowie Freiwilligen aus 50 Ländern entwickelt und mit vielen Kindern speziell für kleine Handy-Bildschirme optimiert wurde. Unsere Gratis-Apps, die schon in über 40 Sprachen übersetzt wurden, sind fast 500.000 Male aus der ganzen Welt auf Smartphones installiert worden.

Apps programmieren kann natürlich beliebig aufwendig sein, aber wir machen es so einfach und vor allem so spannend und motivierend wie nur möglich. Schon nach wenigen Sekunden oder Minuten gibt es erste Erfolgserlebnisse, und nach oben hin existieren dank der Kreativität der Kinder keine Grenzen. Unser Catrobat-Projekt hat viele Erfolge vorzuweisen, unter anderem den österreichischen Staatspreis für Multimedia und e-Business im Bereich Innovation sowie die fünfjährige Unterstützung durch Google, und wir sind weiters das einzige nicht-US Projekt auf code.org.

Seit 2015 läuft unser "No One Left Behind" Horizon2020-Projekt, bei dem es um die Einführung unserer Apps im Schulunterricht inklusive Learning-Analytics geht. Unser Forschungsthema in Österreich dreht sich dabei zentral um die Erhöhung des Mädchenanteils in der IT, und gemeinsam mit unseren Partnern in Spanien, England und Deutschland passen wir unseren pädagogischen Zugang auch für Jugendliche mit Migrationshintergrund und mit kognitiven oder visuellen Behinderungen an. Wir arbeiten im EU Projekt mit neun Schulen und insgesamt 600 Jugendlichen über mehrere Jahre hinweg intensiv zusammen. Die daraus entstehenden Unterrichtskonzepte werden auf unserer Education-Webplattform https://edu.catrob.at/ (die unabhängig vom EU Projekt aufgebaut wird) der Allgemeinheit zur Verfügung gestellt. Wir arbeiten dabei mit


Bildquelle: shutterstock [racorn]

Kinder erstellen Apps mit Hilfe einer visuellen Programmiersprache.

LehrerInnen aller Unterrichtsfächer zusammen, da unser pädagogischer Ansatz sich in alle Projektarbeiten, z.B. im Sprach- oder Sportunterricht, perfekt integrieren lässt.

Das mobile Klassenzimmer werden wir dabei im Rahmen des EU Projekts intensiv wissenschaftlich begleiten und danach so aufbereiten, dass das Modell unter Berücksichtigung unserer Erfahrungen von interessierten LehrerInnen aufgegriffen und unabhängig durchgeführt werden kann – wir sind an einer nachhaltigen Nutzung unserer Unterrichtsmaterialien natürlich extrem interessiert. Ein MOOC für Lehrende mit 18 Videos über unsere Software, den wir im Jänner 2016 online gestellt haben, wurde so bereits mehr als 900 Mal konsumiert.

Prof. Slany hat an der TU Graz auch das sehr erfolgreiche Informatik Lehramtsstudium aufgebaut, wobei die Studierenden im Rahmen ihrer Fachdidaktikausbildung intensiv am Projekt mitarbeiten. *Lesen Sie weiter auf Seite 48.*

Impressum:
Verleger: CDA Verlags- und Handelsges.m.b.H, **Herausgeber:** MinR Dr. Anton Reiter, Abteilung II/8 – IT Didaktik und Digitale Medien, Minoritenplatz 5, 1010 Wien, **Redaktionsanschrift:** A-4341 Arbing, Bundesstr. 9, Tel.: (+43) 07269/60220, Fax: (+43) 07269/60220-44, **E-Mail:** redaktion@cda-verlag.com, **Internet:** http://www.cda-verlag.com

Manuskripte und Programme: Es wird keine Haftung für unverlangt eingesandte Manuskripte übernommen. Die Einsendung von Manuskripten jeder Art gilt als Zustimmung des Verfassers zum Abdruck in den vom Verlag herausgegebenen Publikationen. Der Verlag behält sich das Recht vor, eingesandte Manuskripte nicht zu veröffentlichen. Eine Gewähr für die Richtigkeit der Veröffentlichung kann nicht übernommen werden. Für den Inhalt der Anzeigen haftet ausschließlich der Inserent, eine Prüfung seitens des Verlags erfolgt nicht!

Die IT ist aus unserem Berufs- und Privatleben nicht mehr wegzudenken, und wir wollen daher Jugendlichen helfen, die digitalen Technologien der Gegenwart zu verstehen und damit die Zukunft selbst beeinflussen zu können. Wenn die Kinder dabei ihre eigenen Handys verwenden können, löst dies auch in Europa und Amerika das Infrastrukturproblem an Schulen (keine Informatik-Räume mehr nötig, kein Aufbewahren, Laden, Aktualisieren der Hardware und Software, keine Verantwortung für Verluste oder Schäden durch die Schulen), denn jedes Kind kann sich dann zu jeder Zeit mit unseren Apps auf ihren privaten Handys auseinandersetzen. Viele Länder haben diese Vorteile bereits erkannt, und auch in Europa setzt nun ein Umdenkprozess ein, wobei hier vor allem England seit 2015 eine Vorreiterrolle übernommen hat.

Ein Video über Einsatz von Pocket Code in einer Grazer Schule (wurde im Oktober 2013 von Google gedreht): http://catrob.at/gpfe

Weiterführende Links

- Deutsche Hauptseite: http://www.catrobat.org/de/ --- Englisch: http://www.catrobat.org/#
- Education Website: https://edu.catrob.at/ z.B. mit MOOC auf Englisch: https://edu.catrob.at/fundamental-online-workshop
- Video von 2014 über unser Projekt (Google Team war uns eine Woche in Graz besuchen): https://youtu.be/75i10o_uv0U
- Alice im Wunderland Game Jam während code.org Woche im Dezember 2015: www.alicegamejam.com
- Samsung Galaxy Game Jam bis 23. Oktober 2016: www.galaxygamejam.com
- Coding-for-Kids „mobiles Klassenzimmer" mit Samsung Juni - Oktober 2016: http://samsung.com/at/microsite/digitale-bildung/coding-for-kids.html
- Google+: http://catrob.at/plus oder Facebook: http://catrob.at/fb bzw. Twitter: http://catrob.at/pctwitter
- YouTube: http://catrob.at/youtube

Autor

Wolfgang Slany

Univ.-Prof. Dr. techn. Dipl.-Ing. Wolfgang SLANY, Institut für Softwaretechnologie, Technische Universität Graz
E-Mail: wolfgang.slany@tugraz.at