MedienPädagogik
Zeitschrift für Theorie und Praxis der Medienbildung
www.medienpaed.com
ISSN 1424-3636

Themenheft Nr. 33: Medienpädagogik und Didaktik der Informatik.
Eine Momentaufnahme disziplinärer Bezüge und schulpraktischer Entwicklungen.
Herausgegeben von Torsten Brinda, Ira Diethelm, Sven Kommer und Klaus Rummler

Scalable Game Design Switzerland

Alexander Repenning, Anna Lamprou, Nicolas Fahrni und Nora Escherle

Zusammenfassung
Das Modul «Medien und Informatik» des Lehrplans 21 verlangt von Primarlehrpersonen, dass sie mit ihren Schülerinnen und Schülern verschiedene Kompetenzen und Inhalte im Bereich der Informatik erarbeiten. Für die Erfüllung dieses Auftrags benötigen die angehenden Lehrpersonen eine entsprechende Ausbildung, die sie mit dem relevanten Fachwissen und den erforderlichen Kompetenzen in der Informatik ausstattet. Um dies zu gewährleisten, hat die Professur für Informatische Bildung (IB) der PH FHNW im Herbst 2017 mit dem zweisemestrigen Modul «Informatische Bildung» eines der schweizweit ersten obligatorischen Veranstaltungen für alle Bachelorstudierenden des Instituts Primarstufe eingeführt. Das Modul ist die praktische Umsetzung der theoretischen Konzepte und fachdidaktischen Ansätze von Scalable Game Design Switzerland (SGD Switzerland). Bei SGD Switzerland handelt es sich um die auf Schweizer Erfordernisse angepasste Weiterentwicklung von SGD USA – ein jahrzehntelang erprobtes umfassendes Curriculum für die Vermittlung von informatischer Bildung. Grundlage und Essenz von SGD Switzerland und somit auch von dem Konzept des neuen Moduls ist das Verständnis von Denken mit dem Computer – das sogenannte Computational Thinking. Das Konzept des Moduls «Informatischen Bildung», bei welchem die Schulung des analytischen, lösungsorientierten Denkens mit dem Computer im Fokus steht, ist ein Novum schweizweit. Dieser Beitrag erläutert das Konzept des Moduls IB und präsentiert einen ersten Einblick in die Eindrücke der Studenten, die den Kurs besucht haben.

Scalable Game Design Switzerland

Abstract
The module «Media and Computer Science» of curriculum 21 requires primary teachers to develop various competences and contents in the field of Computer Science (CS) with their students. In order to fulfil this task, the prospective teachers need appropriate training. To ensure this, the Chair of Computer Science Education (IB) of the PH FHNW introduced the two-semester module «Computer Science Education» in autumn 2017, one of the first compulsory courses for all Bachelor students of the Institute of Primary Education in Switzerland. The module is the practical implementation of the theoretical concepts and didactic approaches of Scalable Game Design Switzerland (SGD Switzerland). SGD Switzerland is the further development of SGD USA adapted to Swiss requirements – a

Repenning, Alexander, Anna Lamprou, Nicolas Fahrni, und Nora Escherle. 2018. «Scalable Game Design Switzerland». *MedienPädagogik* 33, (Oktober), 27–52. https://doi.org/10.21240/mpaed/33/2018.10.31.X.

comprehensive curriculum for CS education that has been used and tested for decades. The basis and essence of SGD Switzerland and thus also of the course concept of the new module is the understanding of thinking with the computer – the so-called Computational Thinking. The course concept of the module «Computer Science Education» is a novelty throughout Switzerland. Currently, the IB professorship is teaching the first part of the module (CS Specialization). In this article the philosophy of the course, the tools used, the didactic considerations and the working methods are presented. Finally, first impressions of the students who took the course are presented.

Einleitung

Das Modul «Medien und Informatik» des Lehrplans 21 (LP 21) verlangt von Primarlehrpersonen, dass sie mit ihren Schülerinnen und Schülern verschiedene Kompetenzen und Inhalte im Bereich der Informatik erarbeiten. Für die Erfüllung dieses Auftrags benötigen die angehenden Lehrpersonen eine entsprechende Ausbildung, die sie mit dem relevanten Fachwissen und den erforderlichen Kompetenzen in der Informatik ausstattet. Um dies zu gewährleisten, hat die Professur für Informatische Bildung (IB) der Pädagogischen Hochschule Nordwest-Schweiz (PH FHNW) im Herbst 2017 mit dem zweisemestrigen Modul «Informatische Bildung» (Modul IB) eines der schweizweit ersten obligatorischen Veranstaltungen für alle Bachelorstudierenden des Instituts Primarstufe eingeführt. Dieses Modul, bestehend aus den Modulteilen Fachwissenschaft (1. Semester) und Fachdidaktik (2. Semester), ist die praktische Umsetzung der theoretischen Konzepte und fachdidaktischen Ansätze von *Scalable Game Design Switzerland* (SGD Switzerland). SGD Switzerland ist ein umfassendes Curriculum für die Vermittlung von informatischer Bildung. Bei SGD Switzerland handelt es sich um eine Weiterentwicklung von SGD USA, welches von Alexander Repenning ursprünglich in den USA entwickelt, jahrzehntelang erprobt und durch Forschungsprojekte begleitet und fortentwickelt wurde (Repenning et al. 2015; Repenning 2014). Mit der Einrichtung der Professur IB im Jahr 2014 wurde SGD Switzerland ins Leben gerufen und begann die Anpassung und Feinabstimmung auf die Erfordernisse der Schweizer Bildungslandschaft (Lamprou, Repenning, und Escherle 2017).

Grundlage und Essenz von SGD Switzerland und somit auch von dem Kurs-Konzept des neuen Moduls ist das Verständnis und die Vermittlung des sogenannten *Computational Thinking* (CT) – jenes analytisch-strategischen *Denkens mit dem Computer*, welches die Qualitäten des Menschen mit den Fähigkeiten des Computers kreativ vereint, um komplexe Probleme zu lösen. Entsprechend der Philosophie von SGD Switzerland fasst das Modul IB informatische Bildung auf der Primarstufe primär als die konzeptuelle Erarbeitung und die fächerübergreifende, transversale Anwendung von CT. Aus diesem Grund spielt CT eine zentrale Rolle bei der Vermittlung aller Inhalte, Konzepte und Methoden des Moduls IB und ist oberstes Kompetenzziel.

Das Kurskonzept des Moduls IB, welches die Schulung des analytischen, handlungsorientierten *Denkens mit dem Computer* in den Fokus nimmt, ist ein Novum schweizweit. Häufig stehen vielerorts noch immer vor allem das Unterrichten von Medienbildung und Anwendungskompetenzen, wie etwa das Setzen eines Passworts oder die Nutzung von Office-Anwendungen, im Zentrum der Ausbildungen und Weiterbildungsangebote für Primarlehrpersonen. Während dies bei uns klar nicht der Fall ist, integrieren wir aber durch unseren Ansatz durchaus auch Aufträge, welche das Anwenden des Computers und verschiedener Software beinhalten, und diskutieren in verschiedenen Kontexten relevante Fragen aus dem Gebiet der Medienbildung. Aktuell unterrichtet die Professur für Informatische Bildung den ersten Modulteil Fachwissenschaft. In diesem Beitrag werden die dem Modul IB zugrundeliegende Definition von CT erläutert, die Eckpfeiler des Kurskonzepts und die eingesetzten Werkzeuge beschrieben, welche für die konkrete Gestaltung der beiden Modulteile federführend sind. Abschliessend können erste Eindrücke der Studenten, die den Kurs besucht haben präsentiert werden.

Computational Thinking

CT ist eine entscheidende transversale Grundkompetenz, die eine mündige Teilhabe an der heutigen digitalisierten Informationsgesellschaft ermöglicht. Denn CT ist nicht nur für die Informatik wichtig. In nahezu sämtlichen Lebensbereichen und Fachbereichen ist das Digitale präsent und gewinnt das Verständnis informatischer Konzepte an Bedeutung. CT ist nicht das Gleiche wie Programmieren. Jede Programmiersprache ist durch eine eigene Syntax gekennzeichnet, die Benutzerinnen und Benutzer beherrschen müssen, um funktionierende Programme schreiben zu können. Zum Zweck einer allgemeinen Mündigkeit in der digitalen Welt und insbesondere auf der Primarstufe sind jedoch solche syntaktischen Details einzelner Programmiersprachen nicht so wichtig wie das Grundverständnis allgemeingültiger Konzepte der Informatik. Dem eingedenk entspricht CT einem grundlegenden Verständnis allgemeiner grundlegender Informatik- und somit auch Programmierkonzepte, die in diversen Kontexten und auch auf verschiedene Programmiersprachen anwendbar sind. CT umfasst diverse Einzelkompetenzen wie ausgeprägtes Abstraktionsvermögen (den sprichwörtlichen ‹Blick für das Wesentliche›), algorithmisches Denken, die Fähigkeit, komplexe Phänomene in überschaubare Einzelteile zu zerlegen, Mustererkennung, analytisch-kritisches Denken (auch im Sinne eines reflektierten Umgangs mit den digitalen Medien) und viele mehr.

Obgleich es bis heute keine allgemeingültige Definition von CT gibt, herrscht unter Forschenden und Lehrenden auf dem Gebiet der informatischen Bildung weitgehend Einigkeit sowohl über die wichtigsten Aspekte von CT als auch über den Status von CT als eine transversale Grundkompetenz für alle Menschen, die den Anspruch

haben, aktiv mitwirkende, mündige Bürger digitalisierter Informationsgesellschaften zu sein (Barr et al. 2011; Wing 2006; 2014; Lee et al. 2014; Yadav et al. 2014; Gretter und Yadav 2016; College Board 2017b; Repenning 2015). Ebenso fordern zahlreiche Autoren, dass CT in sämtlichen Klassenzimmern und möglichst auch bereits in den Primarschulen gelehrt werden sollte (Barr und Stephenson 2011; Lee et al. 2014; Repenning 2015; Yadav et al. 2014). Dies ist eine erfreuliche Ausgangslage für die baldige feste Verankerung von CT als zu vermittelnde Kernkompetenz auch und vor allem in Schweizer Schulen. Nicht zuletzt, weil mit der Vermittlung von CT viele Elemente des LP21-Moduls «Medien und Informatik» abgedeckt werden können. Während man im englischsprachigen Raum bereits weiter ist[1], fehlen im deutschsprachigen Raum leider nach wie vor allgemein anerkannte didaktische Konzepte oder gar konkrete Unterrichtsszenarien, wie CT gezielt gelehrt und somit gelernt werden kann. Diese Situation gilt es zu ändern, um die Schweizer Bildungslandschaft fit für die Anforderungen einer zunehmend digitalisierten Zukunft zu machen. Das in diesem Beitrag beschriebene Kurskonzept des Moduls IB wurde zu eben diesem Zweck entwickelt.

Die Definition von CT, die dem Modul IB zugrunde liegt, orientiert sich primär an derjenigen der renommierten Informatikerin Jeannette Wing. Wing fasst CT als einen Gedankenprozess, der sowohl die Formulierung eines Problems als auch die Repräsentation der Problemlösung so darstellt, dass sie von Menschen oder durch Maschinen ausgeführt werden könnte (2014). Unsere, auf Wings aufbauende, Definition orientiert sich stark an den Bedürfnissen der Primarschule, ihren Lehrpersonen und ihren Schülerinnen und Schülern (Repenning 2015). In diesem Sinne wird CT nach Seymour Papert im konstruktionistischen Sinn als handlungsorientiertes, transversales *Denken mit dem Computer* definiert (1996). Dabei fungiert der Computer als Hilfsmittel, das den menschlichen Denkprozess bei der Entwicklung von Problemlösungsstrategien unterstützt und dabei hilft, die Konsequenzen des eigenen Denkens zu visualisieren und damit potenziell mentale oder digitale Artefakte wie beispielsweise Computerprogramme zu generieren.

Ein entscheidender weiterer Aspekt ist der dezidiert iterative Charakter von CT, den bereits die Bezeichnung von CT als *Prozess* andeutet und auf den Wing an zahlreichen Stellen verweist (Wing 2006; 2014). Hier wird CT beschrieben als iterativer Prozess in Analogie zu bewährten wissenschaftlichen Methoden, die Wissen generieren, indem sie Hypothesen zur potenziell sinnvollen oder richtigen Vorgehensweise bei der Lösung eines Problems oder der Beantwortung einer Frage aufstellen, die gewählte Vorgehensweise zur Lösung des Problems anwenden, die dadurch

1 Im englischsprachigen Raum existieren zahlreiche Ideen und Unterrichtsmaterialien für die Lehre von CT, beispielsweise das das «AP Computer Science Principles Curriculum» (College Board 2017b), das UK «Computing at School»-Projekt (http://community.computingatschool.org.uk/resources/252/single) mit einer speziellen Abteilung «Barefoot» (https://barefootcas.org.uk) für die Primarstufe, oder «Exploring Computational Thinking» von Google (https://edu.google.com/resources/programs/exploring-computational-thinking/index.html#!home).

generierten Ergebnisse anschliessend hinsichtlich der gewünschten Problemlösung einer Analyse unterziehen und diesen Prozess so häufig wiederholen, bis die Ergebnisse zufriedenstellend sind (Wing 2008). In Anlehnung an Wings Definition wird CT im Rahmen des Moduls IB als iterativer Prozess gefasst und in drei Phasen gegliedert (Repenning 2015) (Abb. 1):

1. ***Abstraktion*** (Problemformulierung): In der einfachsten Form die Formulierung einer präzisen Fragestellung basierend auf einer Problemanalyse.
2. ***Automation*** (Repräsentation einer Lösung): Die exakte, unzweideutige Darstellung eines Lösungsweges basierend auf einer Kombination von Text und Diagrammen, zum Beispiel in Form eines Computerprogramms.
3. ***Analyse*** (Ausführung und Bewertung der Lösungsrepräsentation): Nach Ausführung eines Lösungsweges, beispielsweise in Form eines Computerprogramms, erfolgt die Evaluation von dessen Qualität.

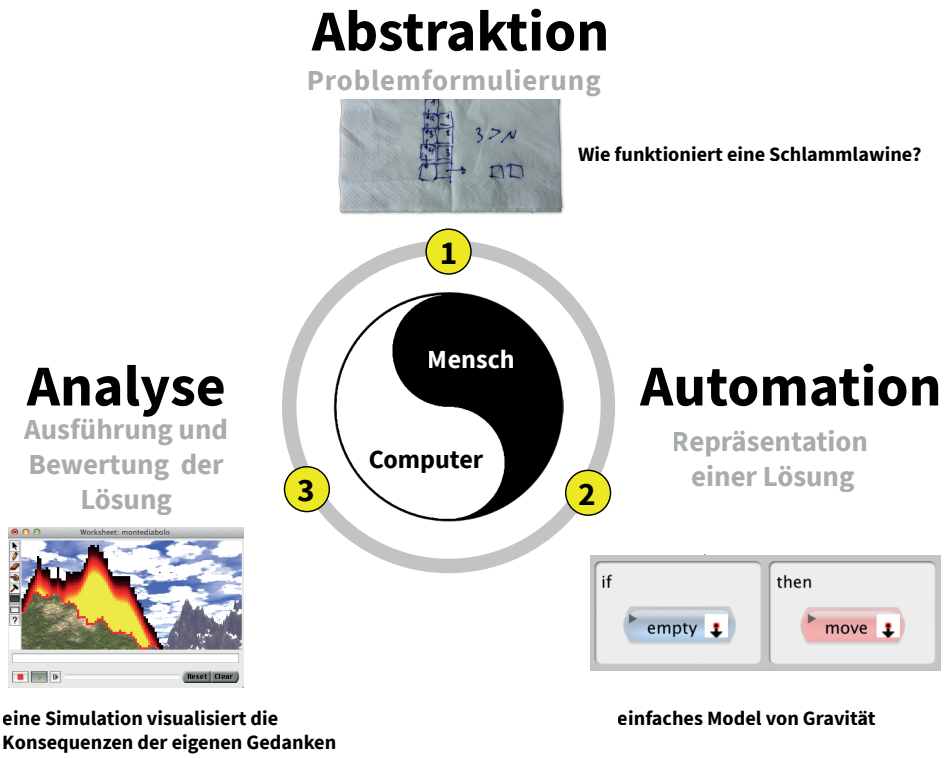

Abb. 1.: Die drei Stufen der Computational Thinking Prozesses (Repenning, Basawapatna, und Escherle 2016, 219).

Die zentrale Vision der beiden Veranstaltungen des Moduls IB es, zukünftige Primarlehrpersonen in der Kernkompetenz des CT zu schulen und sie dazu zu befähigen, diese Kernkompetenz in ihren Schülerinnen und Schülern aufzubauen und zu fördern.

Kurskonzept

Um diese zentrale Vision zu verwirklichen, unternimmt das Modul IB die praktische Umsetzung der theoretischen Konzepte und fachdidaktischen Ansätze von SGD Switzerland, indem es sich hinsichtlich der wichtigsten Inhalte und Kompetenzen orientiert am LP 21, der Informatik und Medien als eigene, fächerübergreifende Kompetenzbereiche definiert. Im Modul IB wird der Kompetenzbereich Informatik mit den Kompetenzzielen hinsichtlich Datenstrukturen, Algorithmen und Informationssystemen prioritär behandelt. Diese Setzung beruht wesentlich auf zwei eng verknüpften Faktoren: Zum einen erwarten wir seitens der Studierenden in diesem Bereich nur wenig Vorwissen und gehen davon aus, dass dieses zunächst aufgebaut werden muss. Zum anderen ist aber die Informatik ein hoch anspruchsvolles Fachgebiet, in welchem selbst innerhalb von zwei Semestern nur rudimentäre Kompetenzen aufgebaut werden können. Dazu gehört nebst grundlegenden Programmierkenntnissen und Informatikwissen primär das Verständnis, was kreative Prozesse ausmacht und wie diese nachhaltig angeregt werden können. Die Studierenden benötigen für den Kurs keinerlei Vorwissen der Informatik. Wichtig ist nur, dass Sie interessiert sind, wie sie mit ihren zukünftigen Schülerinnen und Schülern dank dem Computer kreative Prozesse anregen können. Dazu gehört nebst grundlegenden Programmierkenntnissen und Informatikwissen primär das Verständnis, was kreative Prozesse ausmacht und wie diese nachhaltig angeregt werden können. Eine geeignete Strategie zur Vermittlung von CT kombiniert projektorientierten Unterricht mit einem Fokus auf das Motivationsmodell und die Lernstrategien des SGD-Curriculums, nutzt stufengerechte und anwendungsfreundliche Programmier- und Kreativitätswerkzeuge und basiert inhaltlich wie strukturell auf allgemein als fundamental anerkannten Informatik-Prinzipien, die eine sinnvolle Einbindung zentraler Themen der Medienbildung gewährleisten. In Abstimmung auf das oberste Kompetenzziel des CT und dessen Verknüpfung mit den Kompetenzen des LP 21 hat das Kurskonzept des Moduls IB drei Eckpfeiler:

1. das Motivationsmodell und die Lernstrategien des SGD-Curriculums,
2. Werkzeuge, die speziell zur Schulung und Unterstützung von CT in der Schule konzipiert wurden – sogenannte *Computational Thinking Tools* (CT Tools), und
3. die sieben grossen Themen der Informatik – in Analogie zu den «7 Big Ideas» des «AP Computer Science Principles»-Kurses (College Board 2017b).

Diese drei Eckpfeiler werden im folgenden Abschnitt detailliert beschrieben.

SGD Switzerland: Die drei Eckpfeiler des Moduls Informatische Bildung

Die drei Eckpfeiler sind die grundsätzlichen Komponenten des Moduls IB Kurses, um CT auf eine motivierende, konstruktionistische und handlungsorientierte Weise vermitteln zu können. Sie werden im Folgenden genauer erläutert.

Motivationsmodell und Lernstrategie des Scalable Game Design-Curriculums

Das Motivationsmodell und die Lernstrategie des SGD-Curriculums wurden ursprünglich in den USA erarbeitet und dort über viele Jahre hinweg auf der Grundlage von diversen umfassenden analytisch-evaluativen Durchführungen im Rahmen von Forschungsprojekten weiterentwickelt (Basawapatna et al. 2013). Die Konzepte und Methoden des SGD-Curriculums wurden mittlerweile in unterschiedlichsten Ländern und Kulturen inklusive Mexiko, Brasilien und Japan für die Vermittlung von CT und informatischer Bildung eingesetzt. Die bei diesen Durchführungen erhobenen Daten bezeugen nicht nur eine grosse Begeisterung bei den teilnehmenden Schülerinnen und Schülern, sondern belegen auch, dass die Lehrpersonen oft überrascht sind über ein bisher unbekanntes Mass an Motivation und Lernbereitschaft bei ihren Schülerinnen und Schülern (Repenning et al. 2015). Auch Erfahrungen bei einer ersten umfassenden Umsetzung des SGD-Curriculums im Schweizer Bildungskontext im Kanton Solothurn zeigen diese grosse Motivation sowie nachweisliche Lernfortschritte bei Schülerinnen und Schülern auf (Lamprou, Repenning, und Escherle 2017).

Motivationsmodell: Die «Zones of Proximal Flow»-Theorie

Das Motivationsmodell des SGD-Curriculums basiert auf der Theorie der «Zones of Proximal Flow» (ZPF) (Basawapatna et al. 2013) und implementiert diese auf praxisnahe und projektorientierte Weise. Die ZPF-Theorie kombiniert bildungsrelevante Aspekte der «Flow»-Theorie (Csíkszentmihályi 1997) mit denen der «Zone of Proximal Development»-Theorie (Vygotsky 1978) und bereichert diese wiederum mit didaktischen Prinzipien des Informatikdidaktikers Seymour Papert (1996) wie beispielsweise den Konstruktionismus. «Flow» bezeichnet den mentalen Zustand von völliger Konzentration und restlosem Aufgehen in einer Tätigkeit. Dieser Zustand wird lokalisiert in dem Kontext, wo Herausforderungen proportional sind zu bestehenden Kompetenzen (grüner Bereich in Abb. 2). Sind die Herausforderungen zu klein im Vergleich zu Kompetenzen, stellt sich Langeweile als Zustand ein (blau-grauer Bereich in Abb. 2). Wenn hingegen die Herausforderungen die Kompetenzen weit übersteigen, resultiert dies in einem Zustand von Angst (roter Bereich in Abb. 2). Im Gegensatz zu Csíkszentmihályi hat sich Vygotsky vor allem für Lernprozesse interessiert. Er sieht einen weiteren Zustand bzw. eine weitere Zone, die «Zone proximaler Entwicklung» (orangefarbener Bereich in Abb. 2), als idealen Zustand für Lernerfolge. In dieser Zone werden Schülerinnen und Schüler durch Herausforderungen bewusst an die Grenze der eigenen Kompetenzen gebracht, um dann, dank gezielter Unterstützung («Scaffolding») durch die Bereitstellung notwendiger neuer Konzepte und geeigneter Methoden zum richtigen Moment einen Lernfortschritt leisten zu können. Diese Art von sozialem Lernen kann sowohl im Rahmen traditioneller Lehrperson/Lernende-Modellen erfolgen, ist aber auch möglich in Konstellationen, wo Lernende

unterschiedlicher Niveaus einander unterstützen. Die ZPF-Theorie beschreibt, wie Lernende dank gezieltem Scaffolding den Herausforderungen-Kompetenzen Raum weitgehend selbstständig navigieren können (Basawapatna et al. 2013).

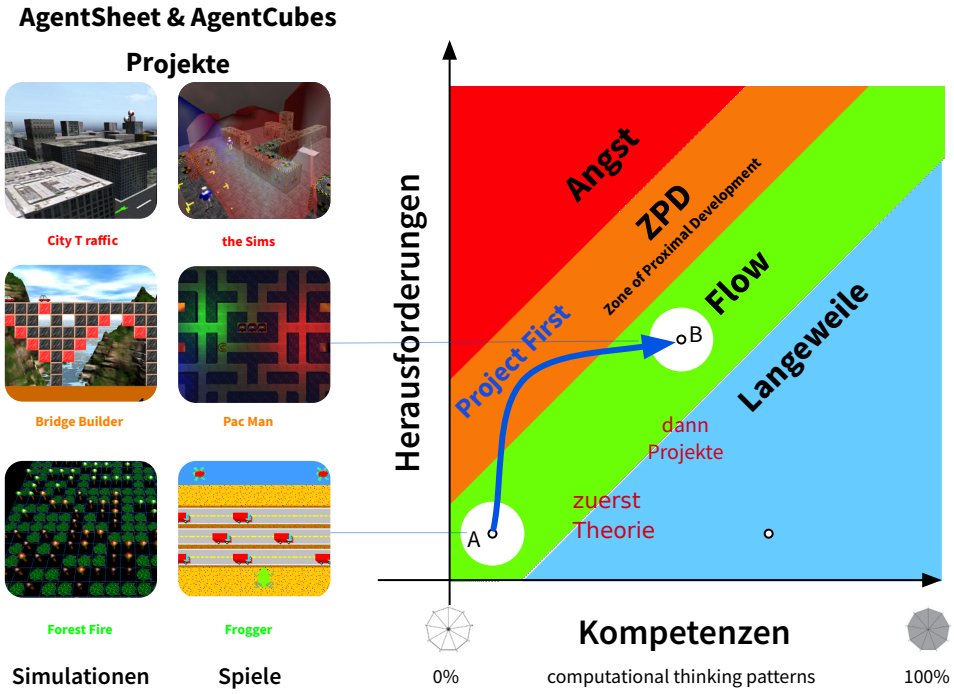

Abb. 2.: Das «Zones of Proximal Flow»-Modell (Basawapatna et al. 2013, 71).

Das Motivationsmodell und die Lernstrategie des SGD-Curriculums sind eng miteinander verschränkt. Traditionell wird oft angenommen, dass Kompetenzen zu Motivation führen. Neuere Modelle haben eher eine gegensätzliche Perspektive, welche Kompetenzen als das Resultat, gewissermassen als Nebenprodukt, von Motivation sieht (Webb, Repenning, und Koh 2012). In diesem Sinne ist Motivation sehr wichtig für Lernprozesse und ein erklärtes Ziel des SGD-Curriculums. Die Forschung hat gezeigt, dass Game Design unabhängig vom Geschlecht motivierend wirkt auf Kinder. Der motivationale Aspekt stützt sich stark auf die Möglichkeit, etwas Eigenes zu schaffen, indem Schülerinnen und Schüler Objekte und Welten in 2D (Repenning and Ambach 1996) oder 3D (Repenning et al. 2012) kreieren, die für sie persönlich interessant und relevant sind. Die Idee, selbst 2D/3D-Objekte zu kreieren und diese dann durch Programmieren zum Leben zu erwecken, ist für eine Vielzahl von Kindern sehr überzeugend. Wichtig ist, dass der Fokus auf Motivation nicht als notwendiges Übel erscheint. Der Erwerb von Kompetenzen wird in motivationsorientierten Ansätzen nicht vernachlässigt oder als irrelevant erachtet. Im Gegenteil. Die grosse Motivation durch Game Design führt beispielsweise dazu, dass Kinder Kompetenzen aufbauen,

die laut traditionellen entwicklungspsychologischen Modellen scheinbar ausserhalb von deren Reichweite liegen. So erlernen beispielsweise Sechstklässler eine projektorientierte Umsetzung anspruchsvoller Diffusionsgleichungen, da sie diese benötigen, um damit die von ihnen selbst erstellten Spiele weiterzuentwickeln.

Lernstrategie: Computational Thinking Patterns

Die Lernstrategie von Scalable Game Design beinhaltet die Verwendung von graduell in ihrer Komplexität und ihrem kognitiven Anspruch steigenden (daher der Begriff «scaling») Computational Thinking Patterns (CTPs). CTPs sind Objekt-Interaktionsmuster, die Schülerinnen und Schüler zunächst im Spieldesign lernen, dann aber auf die Erstellung von MINT-Simulationen übertragen (Koh et al. 2010). Diese CTPs sind abstrahierende Denkmodelle von fundamentalen, häufig wiederkehrenden Interaktionen zwischen verschiedenen Objekten von Computersimulationen oder -spielen bzw. von Benutzern mit diesen Objekten. Da die CTPs mehr oder minder komplexe Abstraktionen sind, ist deren konkrete Umsetzung unabhängig von Programmiersprache oder von Anwendung. Diese Abstraktionen sind ein Teil des CT Prozesses. Es handelt sich um Konstellationen einander zugeordneter Bedingungen und Aktionen, die einem oder mehreren Objekten bei Interaktionen als Verhaltensregeln bzw. -programme in Spielen zugewiesen werden und so deren Verhalten bei Interaktionen bestimmen. Diese Konstellationen von aufeinander abgestimmten Verhaltensregeln sind aufgrund ihres hohen Grades der Abstraktion unabhängig von spezifischen Projekten einsetzbar und können deshalb ohne weiteres zum Erstellen von Objekt-Interaktionen in Simulationen verwendet werden. CTPs können als die Übertragungseinheiten zwischen Spieldesign und wissenschaftlichen Simulationen angesehen werden. Das SGD-Curriculum baut die CT-Fähigkeiten der Schülerinnen und Schüler auf, indem sie von einfacheren Spielen (wie Frogger) an anspruchsvollere Spiele herangeführt werden, die zusätzliche, komplexere CTPs erfordern. Auf ähnliche Weise bewegen sich erfahrene Schülerinnen und Schüler von diesen Spielen weiter hin zu Simulationen, die wiederum von einfacheren Simulationen mit weniger CTPs wie Infizierung durch einen Grippe-Virus bis hin zu komplexeren Simulationen wie der Simulation eines Waldbrandes reichen.

CT steht in enger Beziehung zur Programmierung, weshalb die Schlussfolgerung nahe liegt, dass CT durch Programmier-Unterricht vermittelt werden kann. Die Forschung zeigt jedoch, dass dies ein Trugschluss ist. So fasste beispielsweise Duncan eine Pilotstudie mit neuseeländischen Grundschülern folgendermassen zusammen: «Wir hatten gehofft, dass CT-Fertigkeiten indirekt durch Programmieren und andere Themen im Computerunterricht gelehrt werden, aber von unseren anfänglichen Beobachtungen aus kann dies nicht der Fall sein» (Duncan und Bell 2015). Anders formuliert: Würden Lehrpersonen irrtümlicherweise «erwarten, dass [sie] Schülerinnen

und Schüler in Computational Thinkers verwandeln, indem sie ihnen Code beibringen, ist so, als würden sie erwarten, dass Menschen zu Architekten werden, indem sie IKEA-Möbel zusammenzubauen», sagt Repenning. In der Tat ist CT viel mehr als das Programmieren, und das Unterrichten von CT erfordert den Einsatz spezifischer Ansätze und Werkzeuge. SGDs Strategie beginnt mit einem IKEA-Ansatz, bewegt sich aber darüber hinaus. Durch erste Projektansätze und Gerüsttechniken unterstützt es die Schülerinnen und Schüler beim Erlernen von CTPs. CTPs sind leistungsstarke Abstraktionen, da sie sprachunabhängig sind und es Benutzern ermöglichen, jedes gewünschte Spiel zu erstellen, selbst wenn sie keine IKEA-ähnliche Schritt-für Schritt-Anleitung haben (Abb. 3).

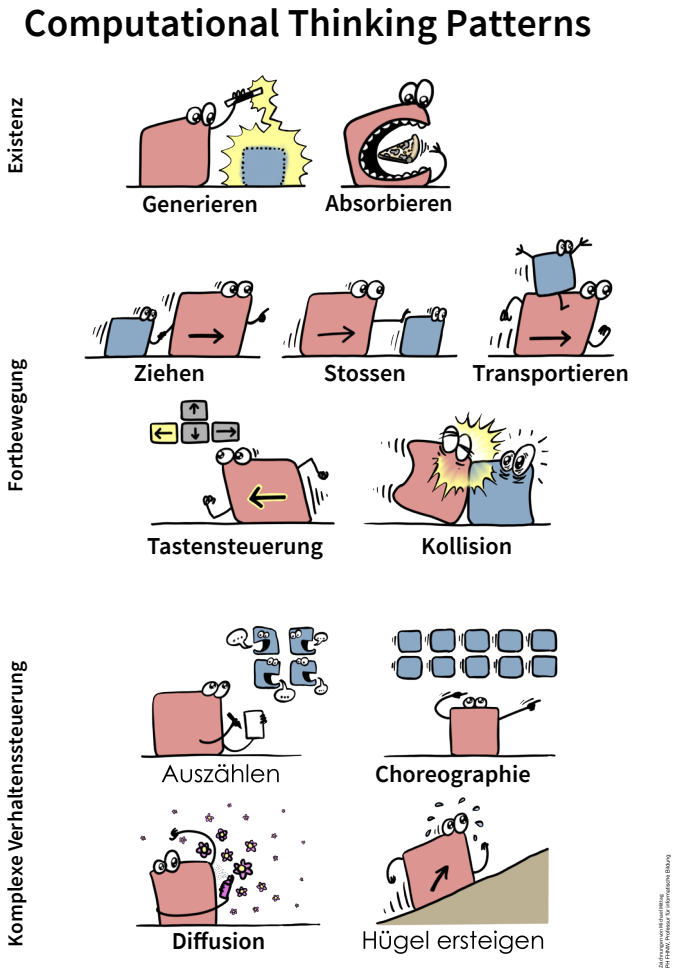

Abb. 3.: CTPs repräsentieren Abstraktionen, die sowohl kognitiv als auch programmiertechnisch gut fassbar sind (Zeichnungen von Michael Mittag, PH FHNW, Professur für informatische Bildung).

Die CTPs, die Schülerinnen und Schüler verwenden, um Spiele zu implementieren, entsprechen den Mustern, die sie verwenden würden, um eine MINT-Simulation zu erschaffen. Der eigentliche Unterschied besteht darin, dass sich der Kontext und die Objekte, die die Interaktionen in der Simulation ausführen, geändert haben. Ein Beispiel einer solchen Abstraktion ist die Kollision von zwei Objekten, die in zahlreichen Kontexten erfolgen und eine Vielzahl von Konsequenzen nach sich ziehen kann. In einem Spiel wie Frogger 3D (Repenning et al. 2015) muss der Benutzer einen Frosch über eine befahrene Strasse manövrieren. In diesem Kontext beschreibt die Kollision die für den Frosch unvorteilhafte Berührung mit einem fahrenden Auto. Ein völlig anderer Kontext ist die Kollision zweier Moleküle in einer Simulation. Die Programme, die geschrieben werden müssen, um die Abstraktion einer Kollision umzusetzen, z.B., in AgentCubes online oder in Processing, können sehr unterschiedlich sein. Aber das Konzept der Kollision bleibt dasselbe. Momentan ist eine kleine Liste von CTPs (Abb. 3) ausreichend, um eine sehr grosse Anzahl von Projekten abzudecken.

Werkzeuge: Computational Thinking Tools

Wie bereits ausgeführt, ist CT nicht das gleiche wie Programmieren. Es befähigt vielmehr zum konzeptuellen Verständnis fundamentaler Informatik-Konzepte und deren Anwendung in verschiedensten Kontexten und Lebensbereichen – auch ausserhalb der Informatik. Forschungsergebnisse, wie z.B. die von Duncan (Duncan und Bell 2015), zeigen, dass spezifische Programmiersprachen und -umgebungen ineffizient sein können für die Vermittlung von CT. Das ist nicht im Sinne der Informatischen Bildung in Primarschulen. CT Tools richten sich, im Gegensatz von allgemein gebräuchlichen Programmiersprachen und -umgebungen, nicht an zukünftige Programmiererinnen, sondern an sogenannte Computational Thinkers. Dank ihrer auf den Bildungskontext ausgerichteten Beschaffenheit und ihrer Einbindung entsprechender Funktionalitäten bieten CT Tools Schülerinnen und Schülern eine maximale Unterstützung bieten, damit diese mit geringem Aufwand eigene Spiele und Simulation erstellen können und dabei ihre CT Kompetenzen schulen.

Viele Schülerinnen und Schüler, vor allem Frauen, nehmen Programmieren als schwierig und langweilig wahr (2014). «Schwierig» ist eine kognitive Dimension und «langweilig» eine affektive Dimension dieser grundsätzlichen Herausforderung. Diese persistenten Bedenken können als zweidimensionaler Raum interpretiert werden, der Raum kognitiver / affektiver Herausforderungen genannt wird (Abb. 4) (Repenning et al. 2012). CT Tools adressieren diesen zwei fundamentalen Herausforderungen.

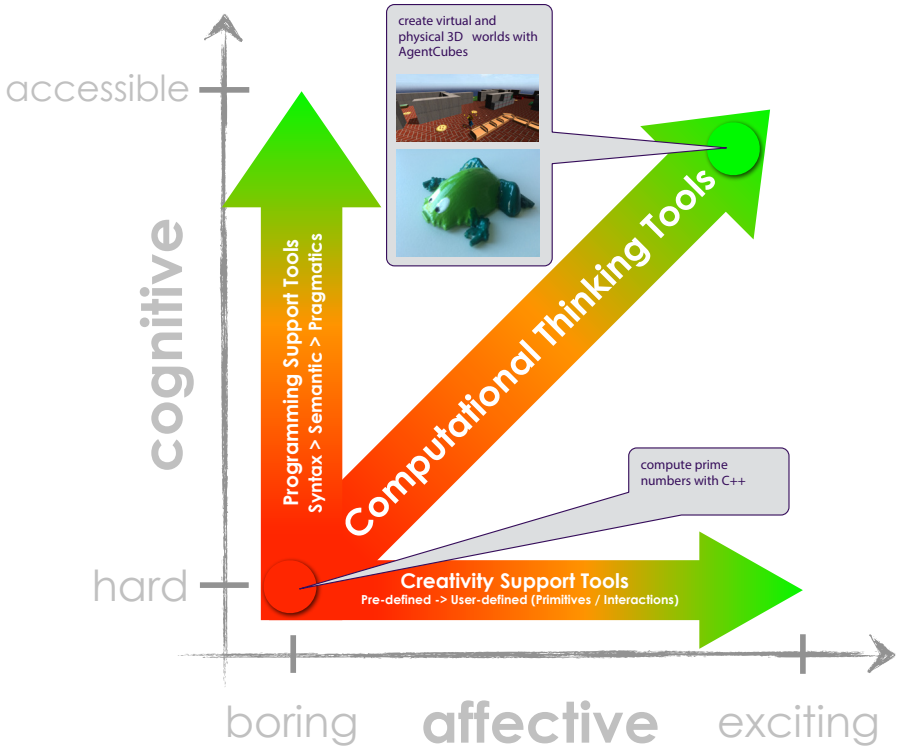

Abb. 4.: Der Raum kognitiver / affektiver Herausforderungen (Repenning 2018).

Der «schwierige» Teil, die kognitive Herausforderung, erfordert einerseits von Programmierumgebungen, dass diese leichter bedienbar und somit zugänglicher zu werden, und andererseits von Programmier-Projekten, dass diese niedrigschwellig sind. Der «langweilige» Teil, die affektive Herausforderung, erfordert von Programmierprojekten, dass diese interessanter sind für die Lernenden und ihnen somit eine grössere Motivation ermöglichen. Die zentrale Frage ist somit, wie man «schwierig und langweilig» in «zugänglich und interessant» umwandelt. Um Programmieren zugänglicher und interessanter zu machen, ist es notwendig, die komplexen Wechselwirkungen zwischen affektiven und kognitiven Herausforderungen zu verstehen. Kinder können sehr interessiert daran sein, ein Spiel oder eine Simulation zu erstellen oder einen Roboter zu programmieren. Wenn aber die Werkzeuge in ihrer Anwendung zu kompliziert sind, besteht eine gute Chance, dass sie schnell aufgeben, weil der ‹return on investment› nicht klar ist. Gute CT Tools kombinieren die wichtigen Aspekte von «Programming Support Tools», welche die Programmierung zugänglich machen und unnötige Komplexität vermeiden, mit denen von «Creativity Support Tools», welche die Programmierung interessant machen (Abb. 4).

Programming Support Tools

Das erste Schlüsselaspekt von CT Tools ist, dass sie durch *Programming Support Tools* unbeabsichtigte, vermeidbare Komplexität (y-Achse Abb. 4) so weit wie möglich reduzieren und so die kognitive Herausforderung adressieren. Die Elemente und konkreten Werkzeuge, welche eine Reduktion der unbeabsichtigten Komplexität ermöglichen, können drei verschiedenen Kategorien zugeordnet werden (Repenning 2017):

- ***Syntaktisch***: *Werkzeuge, die helfen, reine Syntaxfehler zu vermeiden wie z.B., drag and drop programming*. Auf der syntaktischen Ebene kann das Anzeigeformat eines Programms oder von Programmteilen durch Offenlegung der diesen zugrundeliegenden, mitunter komplexeren Elemente gesteuert werden. Diese Anzeigemechanismen können vor allem Anfängern dabei helfen, durch das Ausblenden verwirrender Details den notwendigen Überblick zu behalten und funktionierende Programme zu schreiben. Ein solcher Anzeigemechanismus ist beispielsweise das Dreieck-Symbol in AgentCubes. Es hat die Funktion eines Schalters, der den einfachen und schnellen Wechsel zwischen Anzeigeformaten unterschiedlicher Komplexität und zusätzlichen optionalen Auswahlmöglichkeiten erlaubt. Zum Beispiel können Programmblöcke in AgentCubes optionale Parameter (extern definierte Übergabewerte) haben. Ein Klick auf das Dreieck-Symbol öffnet oder schliesst gewissermassen ein Menü, das diese optionalen Parameter enthält. Der Anzeigemechanismus für Parameter ist für den Begriff der zufälligen Komplexität in dem Sinne relevant, dass optionale Parameter typischerweise gewählt werden, um weniger wichtige oder sogar qualitativ unterschiedliche Parameter zu erfassen, die für das Verständnis der Hauptfunktion eines Programms nicht zwingend nötig sind. In AgentCubes erlauben solche Dreieck-Symbole zudem den Wechsel zwischen verschiedenen Ansichtsformaten von Methoden (eigenständigen Unterprogrammen). So lässt sich die vollumfängliche Implementierung von Methoden mit all ihren Regeln (Kombination von Bedingung «if» und resultierender Aktion «then») ausblenden und nur die nachvollziehbare Dokumentation von deren Funktion innerhalb des Gesamtprogramms anzeigen. (Repenning 2017).
- ***Semantisch***: *Werkzeuge, die zur Erkundung und Überprüfung von Funktionen dienen wie z.B., integrierte Hilfefunktionen und Live Programming*. Auf der Ebene der Semantik ist die Domain-Orientierung (Fischer 1994) die Bereitstellung von Funktionen, die den Anforderungen spezifischer Anwendungsgebiete entsprechen. Der Hauptzweck der Domain-Orientierung im Sinne einer Verringerung der unbeabsichtigten Komplexität besteht darin, dass sie die ansonsten erforderlichen Funktionen eines Aufbaus von Grund auf eliminiert. Wenn beispielsweise eine Programmierumgebung häufig verwendet wird, um wissenschaftliche Visualisierungen zu erstellen, sollte sie eine Domain-Orientierung mit entsprechenden Funktionen enthalten, die für das Erstellen dieser Visualisierungen relevant und nützlich sind.

– ***Pragmatisch***: *Werkzeuge, welche die Untersuchung dessen, was Programme in bestimmten Situationen bedeuten, unterstützen*. Diese Werkzeuge sollten dem Nutzer praxisnah und effizient dabei helfen, Ideen umzusetzen oder über Probleme nachzudenken, indem sie sich auf konkrete Situationen und Kontexte beziehen anstatt auf allgemeine Theorien zu verweisen. *Conversational Programming* (Repenning 2013), ein pragmatisches Werkzeug integriert in AgentCubes, analysiert die spezifische Situation, in der sich selektierte Objekte eines Programms befinden, und zeigt dann den Benutzenden durch farbige Annotationen des Programms an, wie sich diese Objekte verhalten werden. In einem Frogger 3D-Spiel kann beispielsweise der Frosch, der über Strassen hüpfen soll, selektiert werden, um zu verstehen, was als nächstes mit dem Frosch passieren wird. Befindet sich der Frosch zum Beispiel unmittelbar neben einem auf ihn zufahrenden Auto, so signalisiert das Conversational Programming durch grüne Annotation der spezifischen Regel im Programm, dass gleich eine Kollision erfolgt, durch welche der Frosch 'stirbt' und das Spiel neu startet (Repenning 2017).

Creativity Support Tools

Der zweite Aspekt von CT Tools ist, dass sie Kreativität auf hohem Niveau (x-Achse Abb. 4) unterstützen. Es hat sich gezeigt (Webb, Repenning, und Koh 2012), dass Motivation stark davon abhängt, ob Benutzende ein persönlich bedeutsames, eigenes Artefakt erschaffen können. Für viele ist es beispielsweise wesentlich spannender, eine interaktive Welt zu bauen, als Primzahlen zu berechnen. Um diese affektive Herausforderung zu adressieren, sollten CT Tools so genannte *Creativity Support Tools* beinhalten.

Das Ziel von Creativity Support Tools ist es, *den kreativen Prozess zu fördern, zu beschleunigen und zu erleichtern* (NSF 2005). Creativity Support Tools sollten die Benutzer ermutigen, diverse Gestaltungsräume zu erkunden. Sie sollten einerseits Anfängern den Einstieg leicht machen (low threshold), aber andererseits auch Experten die Möglichkeit geben, an immer anspruchsvolleren Projekten (high ceiling) zu arbeiten (Meyers, Hudson, und Pausch 2000). Ausserdem sollten diese Werkzeuge eine Vielzahl an Herangehensweisen und Arbeitsstile unterstützen (high flexibility [Resnick 2009]), da die meisten kreativen Arbeiten in Teams durchgeführt werden und die Zusammenarbeit bestmöglich unterstützt werden sollte.

Konzeptionell kann man Creativity Support Tools mit LEGO Baukästen vergleichen. Ein solcher Baukasten besteht aus einer Sammlung von Bausteinen, die potenziell zu einer schier unendlichen Vielzahl von Artefakten zusammengebaut werden können und ermöglichen potenziell ein Maximum an kreativen Ausdrucksmöglichkeiten. Computergestützte Creativity Support Tools gehen weiter und ermöglichen das Erstellen nicht nur statischer Artefakte, sondern auch dynamischer, interaktiver

Artefakte (NSF 2005). Im speziellen sollen Nutzerinnen und Nutzer dabei unterstützt werden, eigene digitale Artefakte zu erschaffen wie zum Beispiel 3D-Welten, Spiele, Simulation, Roboter, oder Mobile Apps, die sie selbst interessieren. Verschiedene Werkzeuge lassen sich anhand folgender Kategorien unterscheiden und beurteilen (Abb.5).

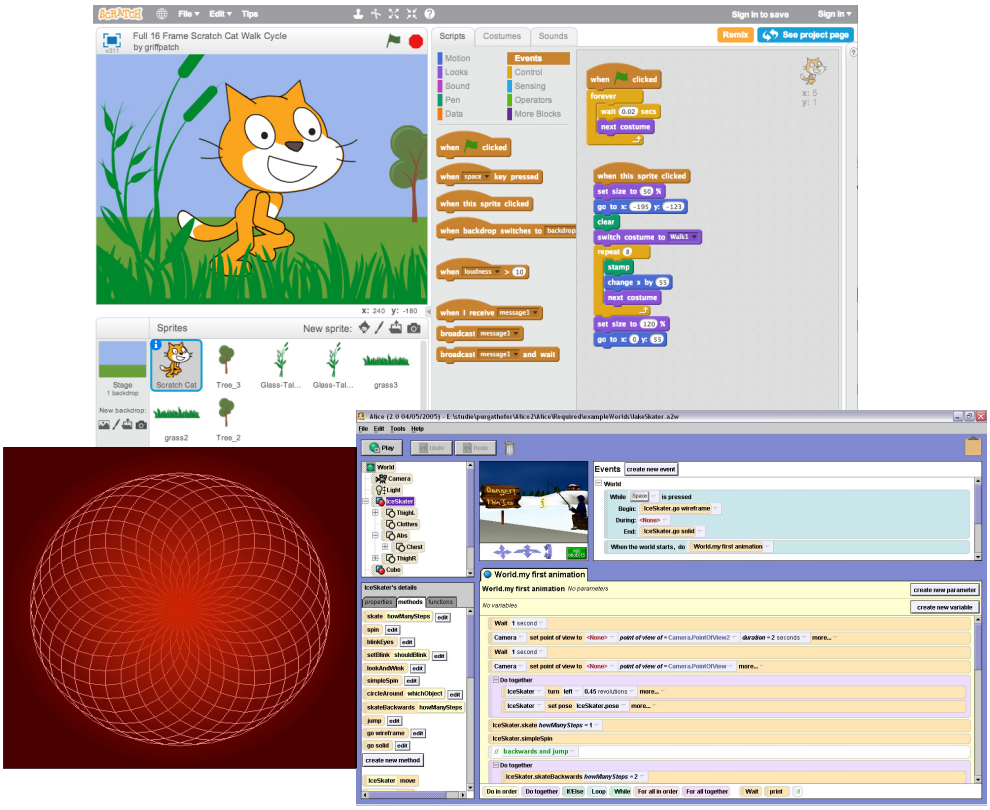

Abb. 5.: Beispiele von Werkzeugen um 2D und 3D Objekte zu kreieren. Scratch (oben) bietet sowohl vordefinierte 2D-Objekte als auch benutzerdefinierte Interaktionen. Alice (unten rechts) bietet nur vordefinierte 2D und 3D Objekte und benutzerdefinierte Interaktionen. Logo (unten links) bietet benutzerdefinierte 2D-Objekte und benutzerdefinierte Interaktionen.

- ***Vordefinierte Objekte***: Eine existierende Sammlung von 2D- oder 3D-Objekten kann, ähnlich wie die Bausteine in LEGO, zu 2D- und 3D-Artefakten zusammengesetzt werden.
- ***Selbstkreierte Objekte***: Im Gegensatz zu physikalischen LEGO Bausteinen, welche Nutzerinnen und Nutzer schwierig selbst herstellen könnten, ermöglichen zahlreiche digitale Creativity Support Tools das selbständige Erstellen eigener, neuer 2D- und 3D-Objekte. Es konnte belegt werden, dass insbesondere die Möglichkeit,

eigene Objekte zu gestalten, grosse Motivation sowie Lernbereitschaft in Lernenden generiert und somit das Potenzial eines langfristigen Lernerfolg steigert (Basawapatna et al. 2018).

– ***Interaktionen***: Damit dynamische digitale Artefakte entstehen, müssen diverse Interaktionen zwischen den virtuellen Objekten aber auch Interaktionen zwischen virtuellen Objekten und den Benutzenden stattfinden. Diese Interaktionen sind in einfachen Fällen vordefiniert. In komplexeren Fällen können weitere, neue Interaktionen durch die Benutzenden selbst anhand von sogenanntem End-User Programming (Lieberman, Paternò, und Wulf 2006) definiert werden.

AgentCubes online

Im Rahmen des Moduls IB ist die Programmierumgebung AgentCubes online das primär verwendete CT-Tool, da es über sämtliche wichtigen Elemente sowohl von Programming Support Tools als auch von Creativity Support Tools verfügt. Im Hinblick auf die Hilfestellung bei der Programmierung bietet es beispielsweise einfache Drag-and-Drop-Programmierung, eine farbliche Annotation fehlerhafter Programmteile (*conversational programming*) sowie ein eingebautes 3D-Raster, das sich extrem vielseitig nutzen lässt. Hinsichtlich der Kreativität gibt es Benutzenden die grosse Freiheit, sowohl die Objekte (einzeln und deren Zusammensetzung zu einer Spielwelt) als auch die Interaktionen (sowohl von Objekten untereinander als auch zwischen Objekten und Nutzerinnen und Nutzern) selbst zu erstellen und somit eine potentiell unbegrenzte Vielfalt unterschiedlicher Projekte zu erschaffen. Dies reicht von einfachen Computerspielen bis hin zu komplexen MINT-Simulationen (Abb. 6).

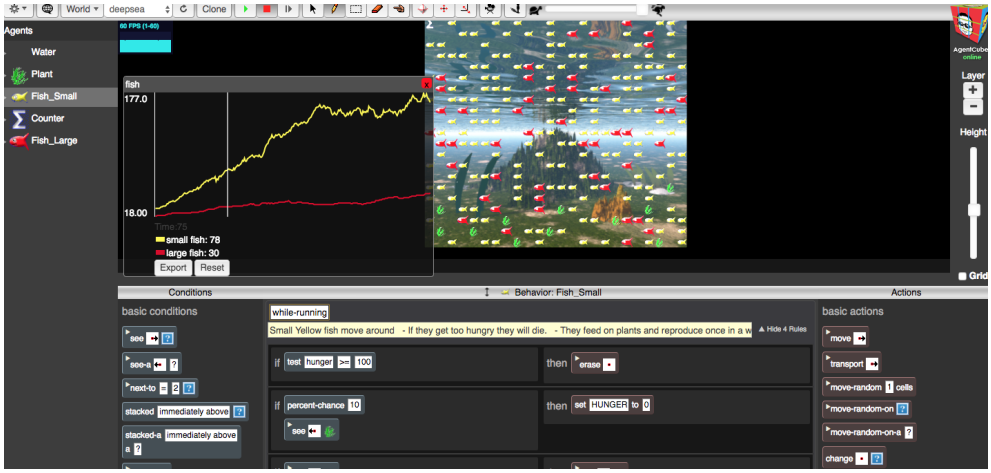

Abb. 6.: Eine Fischteich-Simulation, die mit AgentCubes online erstellt wurde. Anhand von Formeln können Daten erhoben, visualisiert und zur weiteren Untersuchung in Excel- oder Google-Tabellen exportiert werden.

Mit AgentCubes online können Benutzende nicht nur ihre eigenen 3D-Objekte («Agenten») erstellen, sondern diese auch mit 3D-Druckern drucken (Abb. 7).

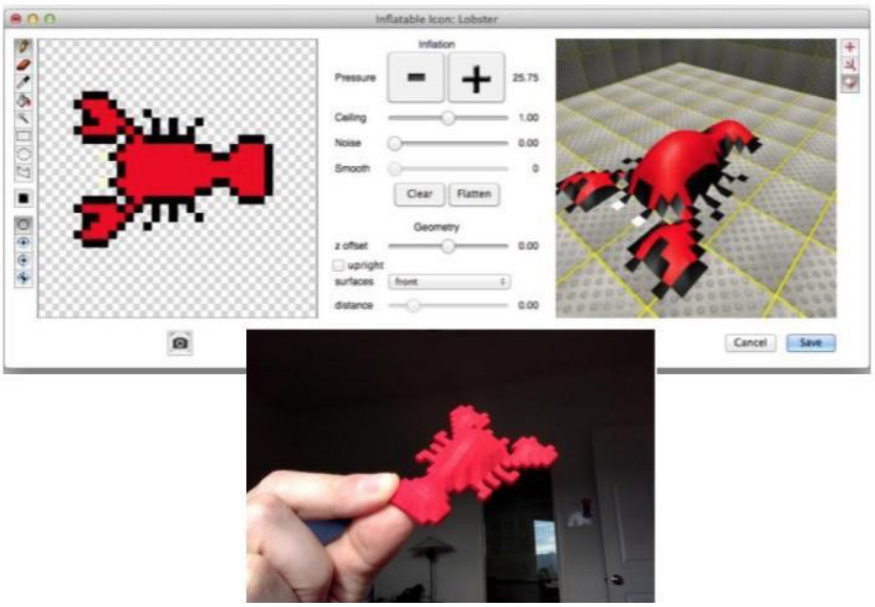

Abb. 7.: Ein Objekt («Agent»), das mit AgentCubes online entworfen wurde: In 2D-Format (oben links), in 3D-Format (oben rechts) und als 3D-Druck (unten Mitte).

Im Rahmen seiner Verwendung als primäres CT-Tool zur Umsetzung des SGD-Curriculums wurde konnte nachgewiesen werden, dass sich AgentCubes online hervorragend eignet für die Vermittlung von CT in der Primarschule (Repenning 2017, Repenning et al. 2015). Dies gründet unter anderem auch darin, dass es einem Low-Threshold-High-Ceiling-Ansatz folgt. Damit ermöglicht es einerseits Personen ohne Vorkenntnisse, sofort mit sehr einfachen Programmier-Aktivitäten zu beginnen, bietet aber andererseits auch fortgeschrittenen Benutzerinnen und Benutzern die Möglichkeit, zunehmend anspruchsvolle und komplexe Projekte zu erstellen.

AgentCubes online eignet sich für Lernende ab Zyklus 2, um durch das Erstellen eigener Projekte viele Grundkonzepte der Programmierung wie Variablen, Parameter, Methoden oder Schleifen kennenzulernen und auszuprobieren. Die Einsatzmöglichkeiten im Unterricht sind vielfältig und reichen von MINT-Fächern über Musik und Gestalten bis hin zu Sprachen. Denn insbesondere das *digital storytellin*g sowie das Erstellen von Spielen und Simulationen ermöglicht den handlungsorientierten, anschaulichen Aufbau von Kompetenzen verschiedener Fachbereiche im Zusammenspiel mit denen der Informatik. Das Erstellen eigener Projekte bietet viel Raum für Kreativität und generiert dadurch ein hohes Mass an Eigenmotivation. Letztere unterstützt auf zentrale Weise die Bereitschaft der Lernenden für die Schulung ihres analytisch-logischen Denken sowie das explorative Erkunden verschiedener Problemlösungsstrategien beim Programmieren eigener Projekte.

Zentrale Inhalte und Konzepte: Die* 7 Big Ideas *der Informatik

Inhaltlich orientiert sich das Modul IB am Kompetenzprofil des Moduls «Medien und Informatik» des LP21. Das Modul IB fokussiert im Sinne seines obersten Kompetenzziels Computational Thinking auf den LP21-Bereich Informatik mit seinen drei Teilbereichen Datenstrukturen, Algorithmen und Informatiksysteme, bindet aber insbesondere wichtige Themen der Medienbildung ein. Denn die in diesem Modul vermittelten Kompetenzen sollen CT zwar primär im Sinne einer grösstmöglichen Selbstwirksamkeit und Handlungsorientierung fördern und zu der selbständigen Erschaffung digitaler Artefakte befähigen. Doch sie sollen ebenso den reflektierte Umgang mit und die produktive Anwendung digitaler Medien ermöglichen. Ein wichtiger Aspekt des Moduls ist dementsprechend auch die kritische Reflexion des digitalen Wandels und der Informatik auf ihre gesellschaftliche Bedeutung hin.

Um auf die Inhalte und Kompetenzziele des LP 21-Moduls «Medien und Informatik» umzusetzen und im Sinne des CT zu lehren, bedient sich das Modul IB zahlreicher Elemente des «AP Computer Science Principles»-Kurses (AP CSP-Kurses) des College Boards (2017b). Der AP CSP-Kurs ist eine Einführung in die Grundlagen der Informatik mit dem Schwerpunkt darauf, wie und mit welchen Auswirkungen Computer die Welt antreiben. Er basiert auf den sechs CT-Praktiken «Connecting Computing», «Creating Computational Artifacts», «Abstracting», «Analyzing Problems and Artifacts», «Communicating» und «Collaborating» sowie den «7 Big Ideas» Kreativität, Abstraktion, Daten, Algorithmen, Programmieren, Internet und Globale Auswirkungen, die er als für die informatische Bildung unerlässlich erachtet (College Board 2017a). Durch den Aufbau an Kompetenzen in den sieben Themenbereichen lernen die Studierenden nicht nur, die Funktionsweise digitaler Technologien zu verstehen, digitale Medien und Daten kritisch zu analysieren und kompetent zu nutzen, sondern auch eigenständig digitale Artefakte zu erschaffen und dadurch selbst aktiv auf die digitalen Welt einzuwirken. Zudem erlangen sie ein umfassenderes Verständnis davon, wie die Informatik die Menschen und die Gesellschaft als Ganzes beeinflusst (College Board 2017a). Dank dieser umfassenden Einbeziehung kritischer Reflexion digitaler Medien sowie der Digitalisierung allgemein und deren sozio-kulturellen Auswirkungen bietet das Konzept der informatischen Bildung des AP CSP-Kurses eine grosse Schnittmenge mit dem Bereich Medienbildung. Durch diese konzeptuelle Öffnung der informatischen Bildung im Sinne eines transversalen CT ermöglicht der AP CSP-Kurs mit seinen sieben Themen eine sinnvolle Einbindung zahlreicher elementarer Themen der Medienbildung (Gretter und Yadav 2016). Das Modul IB übernimmt zentrale inhaltliche, konzeptuelle sowie strukturelle Elemente des AP CSP-Kurses und übersetzt sie für die Ausbildung von Primarlehrpersonen, um eine optimale, dem Kontext angemessene Vermittlung von CT zu gewährleisten. Im Fokus stehen dabei die sieben grossen Themen:

Kreativität

Das Thema Kreativität beleuchtet die Informatik bzw. den Computer als Sammlung von Werkzeugen, die es den Menschen ermöglichen, ihr kreatives Potenzial auszuschöpfen. Dies bezieht sich sowohl auf den ursprünglichen Wortsinn von Kreativität als schöpferische, neues gestaltende Kraft als auch auf deren Bedeutung im Sinne eines unkonventionellen Ideenreichtums. Egal, ob digitales Klangdesign, Animationen, Websites oder Programme – das kreative Potenzial der Digitalisierung ist enorm. Wie in Abschnitt «Werkzeuge: Computational Thinking Tools» dieses Kapitels besprochen, unterstützen die im Modul IB verwendeten CT-Tools die Kreativität auf einem sehr hohen Niveau. Kreativität ist ein elementarer Aspekt eines stark handlungsorientierten CT-Prozesses und bildet daher einer der Schwerpunkte des Moduls IB. Hinsichtlich der Medienbildung werden durch Lernaufgaben insbesondere Themen wie Urheberrechte und der ethische Umgang mit digitalen Inhalten und Artefakten thematisiert.

Abstraktion

Menschen abstrahieren andauernd, um die Komplexität der Realität zu reduzieren. In der Informatik ist die Abstraktion zentraler Teil von Problemlösestrategien. Es ist die erste Stufe des CT-Prozesses, wie es in Abschnitt 1.1 beschrieben wird. Abstraktion beinhaltet sowohl die Strategie, den Prozess, wie auch das Resultat von Komplexitätsreduktion mit dem Ziel, sich auf die relevanten Aspekte zu fokussieren, um komplexe Phänomene zu verstehen, in ihre Einzelbestandteile zu zerlegen und Probleme zu lösen. Im Rahmen des Moduls IB lernen die Studierenden verschiedene Abstraktionsformen und deren Funktionen kennen: Abstraktion von Daten (z.B. Parameter und Variablen), Abstraktion von Abläufen (z.B. Methoden) und Abstraktion von Objekt-Interaktionen (z.B. die CTPs).

Daten

Der Computer ermöglicht und begünstigt neue Methoden der Informationsverarbeitung. Dies führt zu enormen Veränderungen in verschiedenen Bereichen (z.B. Kunst, Wirtschaft und Wissenschaft). Die Digitalisierung führt zu einer Flut von Daten. Das Sammeln und auch das Auswerten von Daten wird durch die Digitalisierung enorm vorangetrieben. Menschen verwenden den Computer für die Übersetzung, Auswertung und Visualisierung von Rohdaten, um so Informationen und schliesslich Wissen zu generieren. Die Informatik erleichtert und ermöglicht ein neues Verständnis von Daten und Informationen, welche das Wissen der Welt konstituieren. Die Studierenden lernen im Modul IB verschiedene Techniken und Werkzeuge kennen, um aus Daten Informationen abzuleiten. Die Überschneidungen mit wichtigen Aspekten der

Medienbildung sind bei diesem besonders Thema zahlreich. Anhand von diversen Lernaufgaben werden Themen wie Datenschutz und Passwortsicherheit handlungsorientiert eingeführt sowie ein kritisch-reflektierter Umgang mit digitalen Inhalten erarbeitet.

Algorithmen
Bei verschiedensten alltäglichen Aufgaben werden Algorithmen verwendet. Unsere Welt und die Gesellschaft werden heute wesentlich von Algorithmen in Software beeinflusst. Sie sind beispielsweise der Grund, weshalb heute Daten sicher und in grosser Menge übertragen werden können. Die Entwicklung, Verwendung und Analyse von Algorithmen gehören damit zu den fundamentalsten Kompetenzen der Informatik. Im Modul IB erarbeiten Studierende durch diverse Lernaufgaben und konkrete Beispiele, unter anderem durch CS unplugged-Aktivitäten, ein grundlegendes Verständnis von der Funktionsweise von Algorithmen und dem immensen gestalterischen Potenzial, dass diesen innewohnt.

Programmierung
Programmieren und Softwareentwicklung haben unser Leben verändert. Das Ergebnis von Programmieren ist Software und es erleichtert das Kreieren von Artefakten wie digitaler Musik, digitalen Bildern und Visualisierungen. Im Modul IB lernen die Studierenden durch das Erstellen eigener digitaler Artefakte Konzepte und Techniken kennen und anwenden, die sie beim Programmieren, beim Entwickeln von Software und bei der effektiven Nutzung von Software unterstützen.

Internet
Das Internet und Systeme, die auf dem Internet basieren, haben einen enormen Einfluss auf unsere Gesellschaft. Computernetzwerke unterstützen die Kooperation wie auch die Kommunikation zwischen verschiedensten Personen, Gruppen oder Systemen. Die Prinzipien der Systeme und der Netzwerke, die dem Internet zugrunde liegen, sind für die Entwicklung von Informatiklösungen entscheidend. Im Modul IB lernen die Studierenden, wie das Internet funktioniert, erfahren von den Charakteristiken des Internets und der Systeme, auf die es gebaut ist. In den dazu gestellten Lernaufgaben werden auch zentrale Themen der Medienbildung wie Datenschutz und Sicherheit (z.B. Verschlüsselungen, Sicherheitsprotokolle, Urheberrecht und Bildrechte) oder Netzneutralität erarbeitet.

Globale Auswirkungen

Die Informatik (Computation) hat die Denkweise der Menschen, die Arbeit, das Leben wie auch das Spiel verändert. Die Methoden der Kommunikation, der Zusammenarbeit, des Problemlösens wie auch der Handel wurden und werden durch die Digitalisierung verändert. Integraler Bestandteil vieler Innovationen ist der Computer bzw. ein Computerprogramm. Zahlreiche Innovationen in anderen Feldern haben direkt mit der digitalen Datenverarbeitung zu tun. Die Informatik und das Problemlösen mit dem Computer führte zu neuen Erkenntnissen, neuen Entdeckungen und neuen Disziplinen. Mittels anschaulicher Materialien und Lernaufgaben macht das Modul IB den Studierenden diverse dieser Innovationen vertraut und veranlasst sie im Sinne wichtiger Aspekte der Medienbildung zu einer kritischen Reflexion des Nutzens sowie der Gefahren dieser Innovationen einerseits und der eigenen Verantwortung als mündige und aktiv mitwirkende Bürgerinnen und Bürger der digitalen Welt andererseits (College Board 2017b).

Diskussion / Fazit: Die ersten Eindrücke aus dem Kurs

Während das SGD-Curriculum bereits in vielen Ländern erprobt und durchgeführt wurde, ist dessen Implementierung im Bildungskontext der Schweiz im Rahmen des Moduls IB neu. Es ist der erste obligatorische Kurs zur Lehre von CT im Rahmen der Ausbildung von Primarlehrpersonen. Dies ist ein wichtiger Unterschied und ein grosser Vorteil hinsichtlich herkömmlicher Kontexte, in denen Lehrpersonen CT anhand dem SGD-Curriculum kennengelernt haben. In den USA erhalten beispielsweise nur interessierte Lehrpersonen aus Eigeninitiative eine Einführung in das SGD-Curriculum, was im Rahmen von Weiterbildungsformaten wie eintägigen Workshops oder mehrtägigen Sommerschulen erfolgt. Eine obligatorische Ausbildung von Lehrpersonen in informatischer Bildung und CT, wie sie der LP21 erfordert und das Modul IB der PH FHNW umsetzt, existiert in den USA nicht. Während es die umfangreichen und wertvollen Erfahrungen aus den zahlreichen Umsetzungen des SGD-Curriculums in den USA und in anderen Ländern durchaus nutzen kann, muss sich das Modul IB auf die beträchtlichen Anforderungen einer grösseren und auch anderen, viel heterogeneren Zielgruppe einstellen: Denn im Bachelor Primarstufe besteht der Auftrag und somit der Anspruch, alle Studierenden und zukünftigen Lehrpersonen zu erreichen und nicht nur solche, die ohnehin an der Thematik interessiert sind.

Während dieses Kapitel geschrieben wurde, wurde der Modulteil Fachwissenschaft bereits unterrichtet. Erste Rückmeldungen von den Studierenden und den Dozierenden zeigen, dass der Kurs sehr gut ankommt. Dieser Kurs bietet als erster seiner Art in der Schweiz eine wertvolle Lernerfahrung nicht nur für die Studierenden, sondern auch für das Dozierendenteam. Um diese Erfahrungen zu erfassen und auszuwerten, wird das Modul IB begleitet von einem Forschungsprojekt, das

wichtige Beiträge im Bereich der Forschung zur informatische Bildung leisten will. Die Datenerhebung läuft und die Analyse der Ausgangsdaten ist noch unvollständig. Wir möchten jedoch dieses Kapitel mit ersten Eindrücken abschliessen, die aus den Kommentaren von Studierenden zweier Dozierender stammen.

Während IB eine völlig neue Lernerfahrung für zukünftige Primarlehrpersonen ist und normalerweise in der Vergangenheit negative Reaktionen ausgelöst hat, scheinen unsere Studierenden die Notwendigkeit des neuen Faches zumindest akzeptiert zu haben und zeigen überwiegend grosse Motivation und Lernbereitschaft. Viele Studierende verwendeten das Wort «Spass» in ihren Kommentaren, auch wenn sie realisieren, dass Programmieren nicht ihr Interessengebiet ist:

> «Sehr motivierend, auch wenn es im Grunde nicht wirklich mein Gebiet ist. Macht Spass und ich lerne etwas :)»

Andere haben zwar auch Spass, äussern aber doch Bedenken:

> «Die Spiele zu Programmieren macht zwar Spass. Nach dem ersten Mal habe ich es jedoch verstanden wie es funktioniert und hätte mir daher eher etwas Neues gewünscht. Allerdings ist es noch ganz am Anfang des Kurses und somit ist es verständlich dass ihr das so macht.»

Obwohl viele Studierende daran interessiert sind, programmieren zu lernen, äusserten einige ihre Frustration über die noch nicht vorhandene Ersichtlichkeit der Verbindung zwischen dem, was sie in diesem Kurs lernen, und der Nutzbarkeit dieses Wissens in der Zukunft:

> «Das Programmieren macht zwar Spass, denke jedoch nicht, dass wir das als zukünftige Lehrer brauchen. Finde den Umgang mit dem Office viel wichtiger, da viele Lehrpersonen nicht wissen, wie man mit diesen Programmen genau umgeht. Das Programmieren finde ich sinnvoll, wenn man Informatik unterrichtet.»

Sie scheinen auch zu glauben, dass sie aufgrund ihres Mangels an Vorkenntnissen in Informatik im Rahmen des Kurses von zwei Semesterwochenstunden nicht ausreichend Zeit finden würden, um die Lücken zu schliessen:

> «Im Allgemeinen: Die meisten von uns haben kaum Vorkenntnisse in der Informatik, da es in der Schule auf das Tastaturschreiben und den Gebrauch der Programme wie Word und Excel beschränkt war (wenn überhaupt!.) Ein Kurs ein Semester lang ist viel zu wenig, da das Vorwissen kaum da ist (im Gegensatz zu anderen Fächern wie Mathe etc.). Man bräuchte mehr Zeit, um sich einarbeiten zu können und sich an die Denkweise gewöhnen zu können.»

Insbesondere dieser letzte Kommentar zeigt die enorme Relevanz von Veranstaltungen wie denen des Moduls IB im Hinblick auf die Umsetzung des LP21 und die Bedürfnisse der zukünftigen Schweizer Informationsgesellschaft. Zwar befinden wir uns noch immer in der Anfangsphase, doch wir sind auf einem guten Weg zu lernen, wie wir zukünftige Primarlehrpersonen erfolgreich unterrichten können, damit sie CT unterrichten können. Es ist noch zu früh, die langfristigen Auswirkungen des Moduls IB zu bewerten, da die ersten unserer Studierenden frühestens in zwei Jahren in der Primarschullehre tätig sein werden. Doch eines steht fest: Das Modul IB hat das Potenzial, die Zukunft von IB positiv zu beeinflussen und die Schweiz zu einem Vorbild für andere Länder machen.

Literatur

Barr, David, John Harrison, und Leslie Conery. 2011. «Computational thinking: A digital age skill for everyone». *Learning & Leading with Technology* 38 (6): 20–23. https://www.iste.org/docs/learning-and-leading-docs/march-2011-computational-thinking-ll386.pdf.

Barr, Valerie, und Chris Stephenson. 2011. «Bringing Computational Thinking to K-12: What Is Involved and What Is the Role of the Computer Science Education Community?» *ACM Inroads* 2 (1): 48. https://doi.org/10.1145/1929887.1929905.

Basawapatna, Ashok R., Alexander Repenning, Kyu Han Koh, und Hilarie Nickerson. 2013. «The Zones of Proximal Flow: Guiding Students through a Space of Computational Thinking Skills and Challenges». In *ICER '13 Proceedings of the Ninth Annual International ACM Conference on International Computing Education Research*, 67. ACM Press. https://doi.org/10.1145/2493394.2493404.

Basawapatna, Ashok, Alexander Repenning, Mark Savignano, Josiane Manera, Nora Escherle, und Lorenzo Repenning. 2018. «Is Drawing Video Game Characters in an Hour of Code Activity a Waste of Time?» In *ITiCSE 2018 Proceedings of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education*, 93–98. ACM Press. https://doi.org/10.1145/3197091.3197136.

College Board. 2017a. «AP Computer Science Principles». College Board. https://apstudent.collegeboard.org/apcourse/ap-computer-science-principles/course-details.

College Board. 2017b. «AP Computer Science Principles. Course and Exam description». College Board. https://secure-media.collegeboard.org/digitalServices/pdf/ap/ap-computer-science-principles-course-and-exam-description.pdf.

Conway, Matthew, Steve Audia, Tommy Burnette, Dennis Cosgrove, und Kevin Christiansen. 2000. «Alice: Lessons Learned from Building a 3D System for Novices». In *CHI '00 Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 486–93. ACM Press. https://doi.org/10.1145/332040.332481.

Csikszentmihalyi, Mihaly. 1997. *Finding flow: the psychology of engagement with everyday life*. 1st ed. MasterMinds. New York: BasicBooks.

Duncan, Caitlin, und Tim Bell. 2015. «A Pilot Computer Science and Programming Course for Primary School Students». In *Proceedings of the Workshop in Primary and Secondary Computing Education (WIPSCE)*, 39–48. ACM Press. https://doi.org/10.1145/2818314.2818328.

Fischer, Gerhard. 1994. «Domain-Oriented Design Environments». *Automated Software Engineering* 1 (2): 177–203. https://doi.org/10.1007/BF00872289.

Gretter, Sarah, und Aman Yadav. 2016. «Computational Thinking and Media & Information Literacy: An Integrated Approach to Teaching Twenty-First Century Skills». *TechTrends* 60 (5): 510–16. https://doi.org/10.1007/s11528-016-0098-4.

Koh, Kyu Han, Ashok Basawapatna, Vicki Bennett, und Alexander Repenning. 2010. «Towards the Automatic Recognition of Computational Thinking for Adaptive Visual Language Learning». In *VLHCC '10 Proceedings of the 2010 IEEE Symposium on Visual Languages and Human-Centric Computing*, 59–66. Washington, DC, USA: IEEE Computer Society. https://doi.org/10.1109/VLHCC.2010.17.

Lamprou, Anna, Alexander Repenning, und Nora A. Escherle. 2017. «The Solothurn Project: Bringing Computer Science Education to Primary Schools in Switzerland». In *ITiCSE '17 Proceedings of the 2017 ACM Conference on Innovation and Technology in Computer Science Education*, 218–23. ACM Press. https://doi.org/10.1145/3059009.3059017.

Lee, Irene, Fred Martin, und Katie Apone. 2014. «Integrating Computational Thinking across the K-8 Curriculum». *ACM Inroads* 5 (4): 64–71. https://doi.org/10.1145/2684721.2684736.

Lieberman, Henry, Fabio Paternò, Markus Klann, und Volker Wulf. 2006. «End-User Development: An Emerging Paradigm». In *End User Development*, herausgegeben von Henry Lieberman, Fabio Paternò, und Volker Wulf, 9:1–8. Dordrecht: Springer Netherlands. https://doi.org/10.1007/1-4020-5386-X_1.

Luthiger, Herbert, und Susanne Wildhirt. 2018. «LUKAS – Ein Modell zur Entwicklung kompetenzfördernder Aufgabensets. Theoretische Grundlagen». In *Kompetenzförderung mit Aufgabensets. Theorie–Konzept–Praxis*, herausgegeben von Herbert Luthiger, Markus Wilhelm, Claudia Wespi, und Susanne Wildhirt, 7:19–78. Bern: hep Verlag.

Maue, Jens. 2017. «Analyse und Klassifikation der Aufgaben des Lehrmittels ‹Programmieren mit LOGO›». http://www.abz.inf.ethz.ch/wp-content/uploads/2017/07/fv2_maue-jens_lehrmittelanalyse_web.pdf.

Myers, Brad, Scott E. Hudson, und Randy Pausch. 2000. «Past, present, and future of user interface software tools». *ACM Transactions on Computer-Human Interaction (TOCHI)* 7 (1): 3–28. https://doi.org/10.1145/344949.344959.

NSF - National Science Foundation. 2005. «Creativity Support Tools». Report. The National Science Foundation. http://www.cs.umd.edu/hcil/CST/creativitybook_final.pdf.

Papert, Seymour. 1996. «An Exploration in the Space of Mathematics Educations». *International Journal of Computers for Mathematical Learning* 1 (1): 95–123. https://doi.org/10.1007/BF00191473.

PH Bern. 2015. «Ideenset Robotik». IdeenSets für den Unterricht. 2015. https://www.phbern.ch/ideenset-robotik/uebersicht.html.

Repenning, Alexander. 2013. «Conversational Programming: Exploring Interactive Program Analysis». In *ACM International Symposium on New Ideas, New Paradigms, and Reflections on Programming & Software (SPLASH/Onward! 13)*, 63–74. Indianapolis, Indiana, USA: ACM Press. https://doi.org/10.1145/2509578.2509591.

Repenning, Alexander. 2014. «Scalable Game Design: Broadening Participation by Integrating Game Design and Science Simulation Building into Middle School Curricula». In *Future Directions in Computing Education, Summit Meeting, January 8-9, 2014*. Orlando, Florida. http://stelar.edc.org/publications/scalable-game-design-broadening-participation-integrating-game-design-and-science.

Repenning, Alexander. 2015. *Computational Thinking in der Lehrerbildung*. Zürich: Schriftenreihe der Hasler Stiftung. http://www.fit-in-it.ch/sites/default/files/downloads/schrift_repenning-1411-gzd_deutsch_0.pdf.

Repenning, Alexander. 2017. «Moving Beyond Syntax: Lessons from 20 Years of Blocks Programing in AgentSheets». *Journal of Visual Languages and Sentient Systems* 3 (1): 68–91. https://doi.org/10.18293/VLSS2017-010.

Repenning, Alexander. 2018. «Computational Thinking Tools». 2018. https://web.fhnw.ch/plattformen/scalablegamedesign/mediawiki/index.php/Computational_Thinking_Tools.

Repenning, Alexander, und James Ambach. 1996. «Tactile programming: a unified manipulation paradigm supporting program comprehension, composition and sharing». In 1996 *IEEE Symposium of Visual Languages*, 102–9. Boulder, CO: IEEE Comput. Soc. Press. https://doi.org/10.1109/VL.1996.545275.

Repenning, Alexander, Ashok Basawapatna, und Nora Escherle. 2016. «Computational thinking tools». In *2016 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, 218–22. IEEE. https://doi.org/10.1109/VLHCC.2016.7739688.

Repenning, Alexander, Ryan Grover, Kris Gutierrez, Nadia Repenning, David C. Webb, Kyu Han Koh, Hilarie Nickerson, u.a. 2015. «Scalable Game Design: A Strategy to Bring Systemic Computer Science Education to Schools through Game Design and Simulation Creation». *ACM Transactions on Computing Education* 15 (2): 1–31. https://doi.org/10.1145/2700517.

Repenning, Alexander, Robert B. Owen, Corrina Smith, und Nadia Repenning. 2012. «Agentcubes: Enabling 3d Creativity by Addressing Cognitive and Affective Programming Challenges». In *World Conference on Educational Media and Technology, EdMedia 2012, June 26-29*. Denver, Colorado, USA. https://sgd.cs.colorado.edu/wiki/images/e/ed/AgentCubesEdMedia2012.pdf.

Resnick, Mitchel, Brian Silverman, Yasmin Kafai, John Maloney, Andrés Monroy-Hernández, Natalie Rusk, Evelyn Eastmond, u.a. 2009. «Scratch: Programming for All». *Communications of the ACM* 52 (11): 60. https://doi.org/10.1145/1592761.1592779.

Vygotskij, Lev Semenovich. 1978. *Mind in Society: The Development of Higher Psychological Processes*. Cambridge (Mass.): Harvard University Press.

Webb, David C., Alexander Repenning, und Kyu Han Koh. 2012. «Toward an Emergent Theory of Broadening Participation in Computer Science Education». In *SIGCSE '12 Proceedings of the 43rd ACM Technical Symposium on Computer Science Education*, 173. ACM Press. https://doi.org/10.1145/2157136.2157191.

Wing, Jeannette M. 2006. «Computational Thinking». *Communications of the ACM* 49 (3): 33. https://doi.org/10.1145/1118178.1118215.

Wing, Jeannette M. 2014. «Computational Thinking Benefits Society». *40th Anniversary Blog - Social Issues in Computing, New York: Academic Press* (blog). 10. Januar 2014. http://socialissues.cs.toronto.edu/2014/01/computational-thinking/.

Yadav, Aman, Chris Mayfield, Ninger Zhou, Susanne Hambrusch, und John T. Korb. 2014. «Computational Thinking in Elementary and Secondary Teacher Education». *ACM Transactions on Computing Education* 14 (1): 1–16. https://doi.org/10.1145/2576872.