Programmieren lernen mit dem genetischen Ansatz

Braune, G; Mühling, A.
Leibniz Institut für Pädagogik der Naturwissenschaften und Mathematik & Universität Kiel
DOI: 10.18420/ibis-02-01-08

Zusammenfassung

Durch die Ausweitung des Informatikunterrichts und insbesondere durch die Einführung von verpflichtendem Unterricht in immer mehr Bundesländern steigt der Bedarf an konkretem Material zur Umsetzung der Fachanforderungen und Curricula. Dieser Beitrag beschreibt ein Konzept für den Anfangsunterricht im Programmieren sowie das auf der Grundlage des Konzeptes erstellte Unterrichtsmaterial. Das Material zielt besonders auf das Pflichtfach Informatik ab, ist aber auch für Wahlkurse und Wahlpflichtkurse geeignet. Es betont besonders das *informatische Denken*, die *Themenorientierung* und die *prozessbezogenen Kompetenzen*. Als didaktische Leitlinie wird ein *genetischer Ansatz* verwendet. Das Unterrichtsmaterial besteht aus einer Folge von Arbeitsbögen und weiteren medialen Hilfsmitteln, die den gesamten Unterrichtsverlauf eines halben Jahres abdecken. Es steht online zur Verfügung.

Einleitung

Die Einführung eines Pflichtfachs Informatik an deutschen Schulen schreitet langsam, aber stetig voran (Meisner 2023). Dadurch steigt auch der Bedarf an Konzepten und Materialien, mit denen die Lehrkräfte in die Lage versetzt werden, den Fachanforderungen und Curricula entsprechend zu unterrichten.

In Schleswig-Holstein finden in den Schuljahren 2022-2024 landesweit Pilotprojekte statt, die die Einführung des Pflichtfaches Informatik vorbereiten sollen (IQSH - Institut für Qualitätsentwicklung an Schulen Schleswig-Holstein 2022). Der vorliegende Beitrag stellt ein Konzept mit zugehörigem Unterrichtsmaterial vor, das auf der Grundlage der Fachanforderungen Informatik (Land Schleswig-Holstein 2021) für eine der Pilotschulen entwickelt wurde. Alle Schülerinnen und Schüler des siebten Jahrgangs eines Gymnasiums erlernten über ein halbes Schuljahr hinweg nach Maßgabe des Konzeptes grundlegende Kompetenzen im Bereich des Programmierens. Dabei wurde besonderer Wert auf die in den Fachanforderungen besonders betonten Bereiche *Informatisches Denken*, *Themenorientierung* und *Prozessbezogene Kompetenzen* gelegt. Der Unterricht verbindet etablierte didaktische Theorien mit modernen Methoden und folgt dem *genetischen Ansatz*. Aufgrund ähnlicher curricularer Anforderungen im Bereich des Programmierens ist das Material ganz oder teilweise auch in verschiedenen Schularten anderer Bundesländer und in unterschiedlichen Anwendungskontexten, zum Beispiel in Wahl- und Wahlpflichtkursen, einsetzbar.

Leitgedanken des Unterrichtskonzeptes

Genetischer Ansatz

Die Grundidee des genetischen Ansatzes ist es, die Schülerinnen und Schüler an der Entwicklung fachlicher Konzepte teilhaben zu lassen, statt ihnen die Konzepte als feststehendes Wissen lediglich zu präsentieren. Für den Bereich der Informatik formuliert Schuster (2011) als „Arbeitsdefiniton“: „Genetischer Informatikunterricht fasst informatische Begriffe als ‚gewordene‘ auf und will ihr mögliches ‚Werden‘ im Lernprozess nachvollziehen lassen“. Tragendes Element des genetischen Unterrichts ist immer eine konkrete Fragestellung, die einen „sinnstiftenden Kontext“ schafft (Leuders et al. 2011). Dadurch wird echtes Verständnis gefördert und das isolierte Anhäufen von Wissen vermieden.

Das „genetische Prinzip“ als didaktischer Begriff ist facettenreich und daher hier nur kurz skizziert. Eine detaillierte Beschreibung und Analyse der verschiedenen Perspektiven findet sich z.B. bei Möller (2001).

Informatisches Denken (Computational Thinking)

Die bekannte Charakterisierung informatischen Denkens von Jeanette Wing (2012) lautet: “Computational Thinking is the thought processes involved in formulating problems and their solutions so that the solutions are represented in a form that can be effectively carried out by an information-processing agent”. Mithilfe informatischen Denkens werden also Probleme formuliert und gelöst - die Lösung ist aber ein Prozess. Dieser Prozess wird von einem informationsverarbeitenden „Agenten“ ausgeführt. Das kann ein Computer, aber auch ein nicht-kreativ handelnder Mensch sein, der Anweisungen mechanisch ausführt. Die Erkenntnis, dass Computer Probleme nicht lösen, sondern von Menschen erdachte Lösungen rezeptartig ausführen, ist von grundlegender Bedeutung für das Verständnis des Programmierens und auch in Zeiten der künstlichen Intelligenz weiterhin gültig. Sie wird in der vorliegenden Unterrichtseinheit an mehreren Stellen aufgegriffen.

Themenorientierung bzw. Kontextualisierung

Der genetische Ansatz führt fast automatisch dazu, dass man für den Unterricht eine von den Schülerinnen und Schülern als sinnvoll wahrgenommene thematische Fragestellung aufgreift und aus dieser Fragestellung heraus allgemeine (informatische) Konzepte entwickelt. In diesem Sinn ist der Unterricht themenorientiert, wie von den Fachanforderungen in Schleswig-Holstein gefordert – genauso aber auch aus der Perspektive der Phänomen- (Humbert & Puhlmann 2004) oder Kontextorientierung (Koubek et al. 2009) zu verstehen.

Prozessbezogene Kompetenzen

Franz E. Weinert (2001) definiert Kompetenz als „die bei Individuen verfügbaren oder durch sie erlernbaren kognitiven Fähigkeiten und Fertigkeiten, um bestimmte Probleme zu lösen [...]“.

Eine Kompetenz zu haben, bedeutet also nicht nur, etwas zu *wissen*, sondern auch, auf der Grundlage des Wissens *handeln* zu können. Es liegt daher nahe, bei Kompetenzbeschreibungen zwischen *inhaltsbezogenen* und *prozessbezogenen* Kompetenzen zu unterscheiden.

Die Fachanforderungen in Schleswig-Holstein (Land Schleswig-Holstein 2021) betonen dabei informatische Entwicklungsprozesse und folgen erkennbar den amerikanischen K12-Rahmenrichtlinien (K-12 Computer Science Framework Steering Committee 2016). Abbildung 1 stellt die für Schleswig-Holstein gültigen prozessbezogenen Kompetenzen dar. Der äußere Ring beschreibt das typische iterative Vorgehen der praktischen Informatik: Ein Artefakt erstellen, indem man ein Problem beschreibt, analysiert, löst und dann die Lösung implementiert, evaluiert und ggf. verbessert (Buttke et al. 2011, Best et al. 2019). Dies wird in der vorliegenden Unterrichtseinheit als eine direkte Leitlinie zur Gestaltung des Unterrichts gesehen. Während man das Problem schrittweise löst, ergibt sich die Notwendigkeit, die Inhalte und Vorgänge zu benennen, also durch Abstraktion Konzepte zu gewinnen. Dies wiederum spiegelt die Intention des genetischen Ansatzes wider. Grundsätzlich passt das gewählte Vorgehen auch zu den eher generisch formulierten Prozessbereichen der GI-Bildungsstandards (GI 2008).

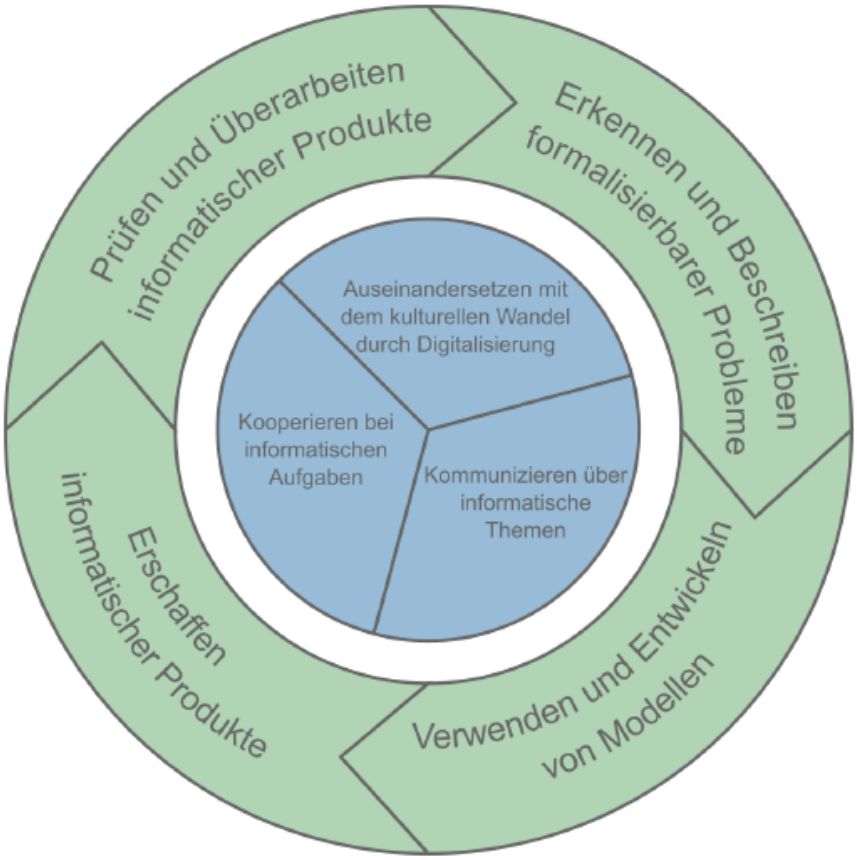


Abbildung 1: Prozessbezogene Kompetenzen in den schleswig-holsteinischen Fachanforderungen (Land Schleswig-Holstein 2021).

Methodische Aspekte

Programmierumgebung

In der vorliegenden Unterrichtseinheit wird als Programmierwerkzeug *Scratch* verwendet. Scratch ist (nicht nur) an schleswig-holsteinischen Schulen sehr weit verbreitet (Braune und Mühling 2020), was sicherlich mit der niedrigen Einstiegsschwelle und der Vermeidung von Syntaxproblemen zusammenhängt (Resnick et al. 2009, Grover 2020). Die schleswig-holsteinischen Fachanforderungen empfehlen auch explizit den Beginn mit einer blockbasierten Programmierumgebung.

In der hier präsentierten Unterrichtseinheit soll Scratch nicht zum Erstellen von interaktiven Geschichten oder Spielen verwendet werden, sondern als Werkzeug zum zielgerichteten, planvollen Lösen von Problemen. Die Unterrichtseinheit ist grundsätzlich auch auf andere (blockbasierte) Programmierumgebungen, etwa Snap!, portierbar, da sie nicht speziell auf Scratch-spezifische Merkmale angewiesen ist.

Unterrichtsmethoden

Zur Realisierung des genetischen Ansatzes werden speziell die folgenden typischen Methoden des Informatikunterrichts benutzt:

- *Computer Science Unplugged*[1], also die Idee, dass sich die eigentliche Problemlösung in den Köpfen der Menschen abspielt und dass es daher das Verständnis fördert, Vorgehensweisen und Lösungskonzepte zunächst abseits des Computers zu entwickeln (Curzon 2013, Bell und Vahrenhold 2018).
- *Cognitive Apprenticeship*, um die Unterstützung durch die Lehrkraft nach und nach zurückzufahren („faded scaffolding“) und dabei typische Praktiken des Fachs Informatik zu modellieren (Lave und Wenger 2011).
- *PRIMM*[2] *(Predict-Run-Investigate-Modify-Make)*: Die aus der Einsicht, dass Lernende nicht von Anfang an selbständig programmieren können, folgende Idee, zunächst gegebene funktionsfähige Programme analysieren und modifizieren zu lassen (Sentance et al. 2019).

Es wird davon ausgegangen, dass es nicht *die eine* Methode gibt, die auf alle Situationen erfolgversprechend anwendbar ist. Stattdessen kommen je nach Erfordernis unterschiedliche Vorgehensweisen zum Tragen.

Aufbau der Unterrichtseinheit

In Anwendung des Ansatzes *Cognitive Apprenticeship* besteht die Unterrichtseinheit aus einem eher lehrkraftzentrierten *Einführungsteil* und einem eher durch selbständige Arbeit der

[1] https://www.csunplugged.org/de/

[2] https://primmportal.com/

Schülerinnen und Schüler charakterisierten *Projektteil*. Im Pilotprojekt dauerte der Einführungsteil rund neun Wochen und der Projektteil rund 10 Wochen (bei einer Doppelstunde Unterricht pro Woche) – grundsätzlich lassen sich die jeweiligen Teile aber in ihrer Dauer auch etwas anpassen.

Unterrichtsmaterial

Überblick

Das Unterrichtsmaterial ist online zugänglich[3] . Es besteht aus drei Teilen:

- einer Einführung in das Unterrichtskonzept und die Nutzung des Materials,
- einer Folge von Arbeitsbögen, denen weitere Unterlagen (z.B. Programme, Präsentationen, Videos) zugeordnet sind und die den gesamten Unterrichtsverlauf des Pilotprojektes widerspiegelt, und
- einigen Ergänzungen und Erweiterungen, darunter Anregungen für weitere Themenorientierungen und noch nicht erprobtes Material für ältere Schülerinnen und Schüler (Ende der Mittelstufe und Oberstufe).

Jeder Haupt-Abschnitt wird durch Informationen für Lehrkräfte eingeleitet. Sie enthalten Empfehlungen für den Unterricht, die auf Erfahrungen aus dem Pilotprojekt basieren.

Die beiden folgenden Abschnitte vermitteln einen Eindruck von dem Material, indem sie beispielhaft zwei darin enthaltene Fragestellungen vorstellen.

Beispiel 1: „Wie speichert das Handy Bilder?“

Diese Fragestellung stammt aus dem eher lehrkraftzentrierten Einführungsteil. Der Unterricht vermittelt hier inhaltsbezogene Kompetenzen aus den Bereichen *Daten und Informationen* und *Algorithmen und Programmierung* (vgl. (Land Schleswig-Holstein 2021) und folgt den in den prozessbezogenen Kompetenzen beschriebenen Schritten (ebd.) der Problemlösung. Die Schülerinnen und Schüler haben im vorangehenden Unterricht erste Erfahrungen mit grundlegenden Kontrollstrukturen gemacht, in Bezug auf das Thema aber keine Vorkenntnisse. Dem genetischen Ansatz entsprechend beginnt der Unterricht mit einer konkreten, am Beispiel orientierten Arbeit an der Fragestellung und entwickelt aus dieser konkreten Arbeit heraus allgemeine Konzepte, die vor dem Hintergrund der Fragestellung „Sinn machen“ (Leuders et al. 2011), zum Beispiel „Rastergrafik“ und „Algorithmus“.

Um die Beschreibung und Lösung des Problems klar von der Implementation der Lösung zu trennen, wird die Fragestellung zunächst ohne Computer untersucht (*Computer Science Unplugged*, siehe oben). Schülergruppen, die in verschiedenen Räumen sitzen, sollen einfache Zeichnungen anfertigen und sich gegenseitig die Zeichnungen „per E-Mail“, also irgendwie

[3] https://oer.ipn.uni-kiel.de/edu-sharing/components/collections?id=3d505196-248c-4404-9e5a-1029f141aa33

zeichenweise, übermitteln, wobei die Lehrkraft den Transport der „Mails“ übernimmt. (Abbildung 2 zeigt einen Ausschnitt aus dem zugehörigen Arbeitsbogen). Anschließend vergleichen die Gruppen ihre Vorschläge und Ergebnisse. Die Erfahrung zeigt, dass neben vielen nicht brauchbaren Ansätzen (über die man dann gut diskutieren kann) oft Ideen geäußert werden, die im Kern mit Raster- oder Vektorgrafiken zu tun haben. Es erfolgt nun eine Beschränkung auf Rastergrafiken. In den folgenden Stunden geht es – den inhaltsbezogenen Kompetenzbereichen entsprechend – um zwei Fragen: Wie lassen sich Bilder als Rastergrafiken repräsentieren? Und: Wie kann man die Bilder in den Computer bzw. das Handy „bringen“ und wieder herausholen? Die erste Frage berührt die verwendete *Datenstruktur*, die zweite diejenigen *Algorithmen*, die auf der Datenstruktur arbeiten.

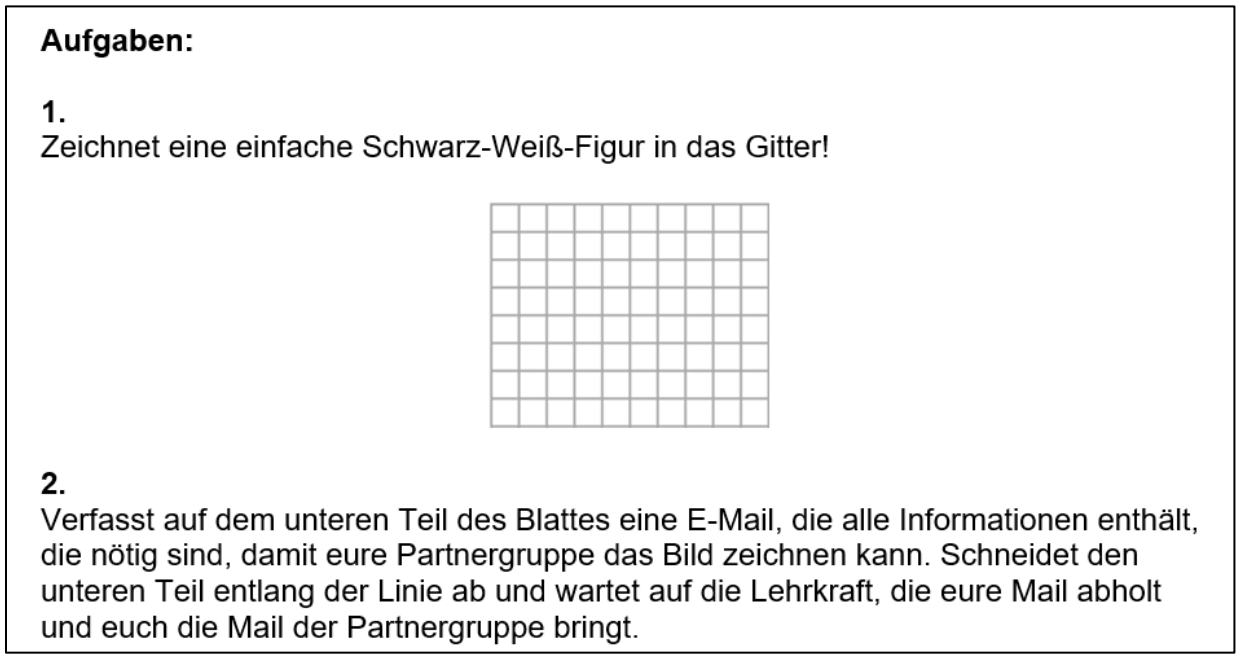


Abbildung 2: Bilder "unplugged" übermitteln

Dem genetischen Ansatz folgend werden diese fachlichen Konzepte jedoch erst eingeführt, wenn die Lösung des gegebenen Problems es erforderlich macht. In Bezug auf die zweite Frage bedeutet dies für den Unterricht Folgendes: Die Beschäftigung mit dem Scannen und Zeichnen von Bildern abseits des Computers („unplugged“) führt zu der Erkenntnis, dass zum Beispiel zum Einscannen eines Bildes in eine Rastergrafik eine bestimmte Folge von Anweisungen durchlaufen werden muss. Dabei wird diese Folge von Anweisungen „stur“ und rezeptartig abgearbeitet, das heißt, der oder die Ausführende muss die Anweisungen nicht inhaltlich verstehen, sondern sie lediglich abarbeiten. Um über eine solche Folge von Anweisungen, die schrittweise ausgeführt wird, gut sprechen zu können, wird ihr jetzt ein Name gegeben: *Algorithmus*. Dies ist natürlich nur eine erste Annäherung an das Konzept, eine genaue Definition kann erst im Laufe der Zeit entstehen, wenn wiederum bestimmte Fragestellungen dies nahelegen. (Zum Begriff des Algorithmus siehe z.B. Hubwieser et al. 2013.)

Die Schülerinnen und Schüler werden überwiegend nicht in der Lage sein, sich einen geeigneten Algorithmus selbst auszudenken und zu formulieren. Sie können aber gegebene Algorithmen gedanklich „nachspielen“ und dadurch *verstehen*, wie Algorithmen arbeiten. Abbildung 3 zeigt beispielhaft eine Aufgabe, die solche Aktivitäten verlangt.

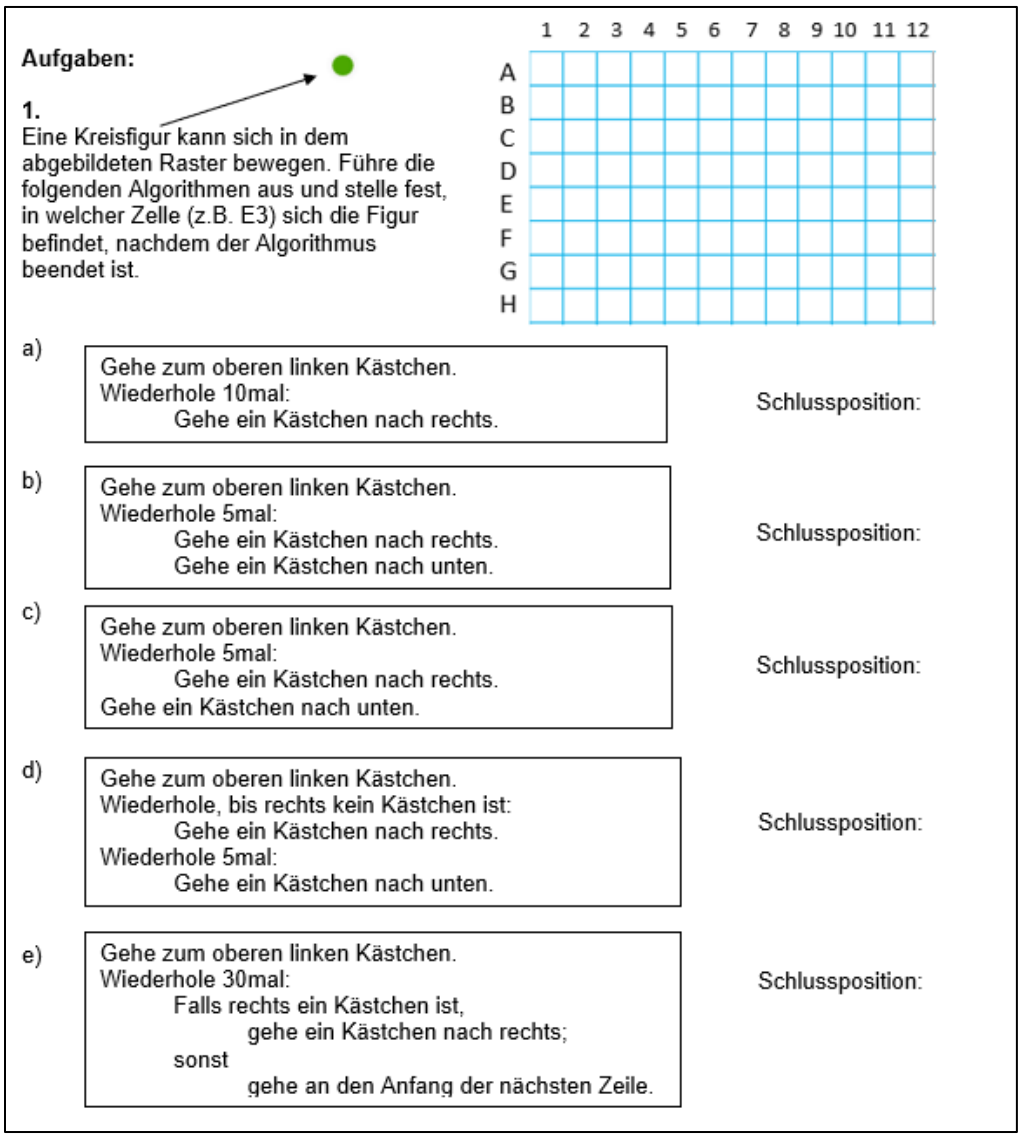
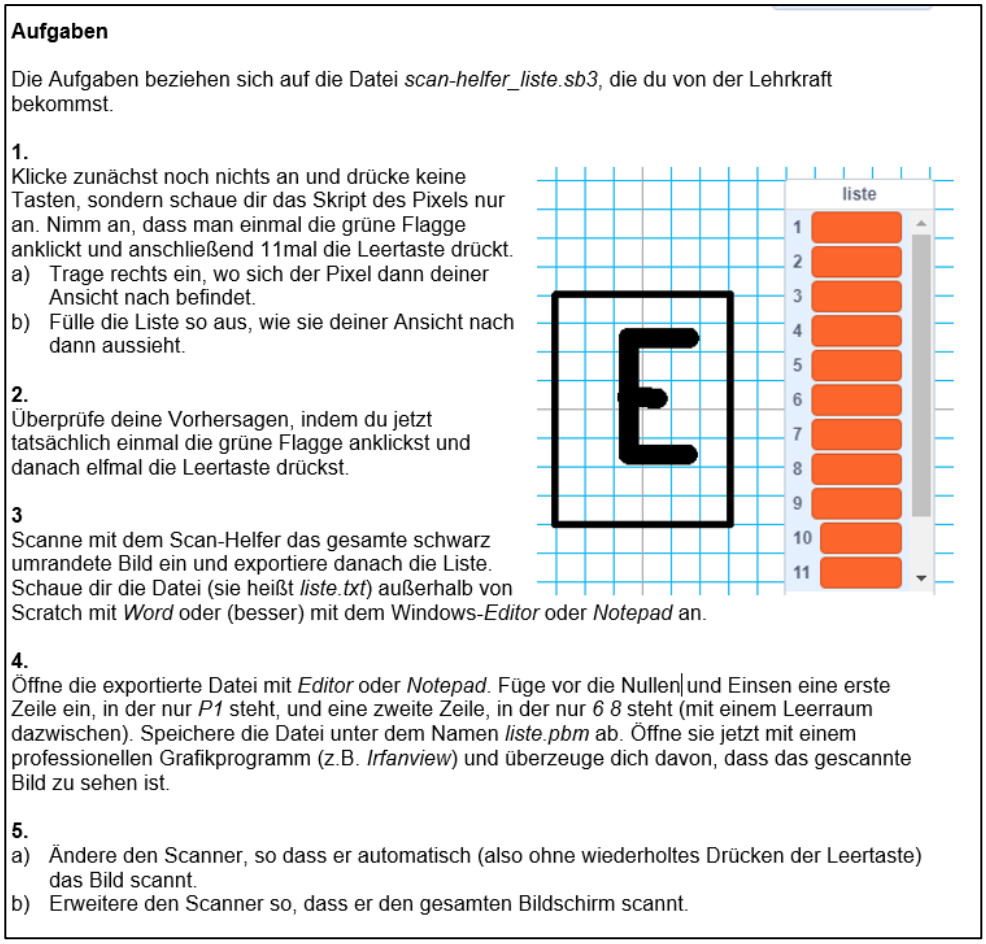


Abbildung 3: Algorithmen lesen und gedanklich abarbeiten

Nachdem das Problem *gedanklich* gelöst ist, geht es darum, die Rolle des oder der Ausführenden dem Computer zuzuweisen. Das bedeutet, dass der Algorithmus in eine Sprache übertragen werden muss, deren Anweisungen der Computer ausführen kann, mit anderen Worten: Er wird *implementiert*. Eine solche Implementierung können die Schülerinnen und Schüler zu diesem Zeitpunkt noch nicht selbständig leisten, sie können aber gegebene (Teil-)Lösungen analysieren, nachvollziehen, ändern und erweitern. Für diesen Zweck bietet der PRIMM-Ansatz (siehe oben) einen effektiv einsetzbaren methodischen Rahmen: Die Schülerinnen und Schüler bekommen funktionsfähige Programme, die Teile des gegebenen Problems lösen, aber noch

keineswegs „perfekt“ sind. Sie lesen die Programme und sagen die vermutete Wirkung vorher („Predict“). Dann lassen sie die Programme ausführen („Run“) und vergleichen die Vorhersagen mit der Realität. Anschließend untersuchen sie die Programme genauer auf bestimmte Aspekte hin („Investigate“) und modifizieren sie in Teilen, um sie zu verbessern („Modify“). Schließlich implementieren sie gewisse Programmteile selbständig („Make“). Die Abbildung 4 zeigt beispielhaft einen nach der PRIMM-Methode erstellten Arbeitsbogen.



Aufgaben

Die Aufgaben beziehen sich auf die Datei *scan-helfer_liste.sb3*, die du von der Lehrkraft bekommst.

1.
Klicke zunächst noch nichts an und drücke keine Tasten, sondern schaue dir das Skript des Pixels nur an. Nimm an, dass man einmal die grüne Flagge anklickt und anschließend 11mal die Leertaste drückt.
a) Trage rechts ein, wo sich der Pixel dann deiner Ansicht nach befindet.
b) Fülle die Liste so aus, wie sie deiner Ansicht nach dann aussieht.

2.
Überprüfe deine Vorhersagen, indem du jetzt tatsächlich einmal die grüne Flagge anklickst und danach elfmal die Leertaste drückst.

3
Scanne mit dem Scan-Helfer das gesamte schwarz umrandete Bild ein und exportiere danach die Liste. Schaue dir die Datei (sie heißt *liste.txt*) außerhalb von Scratch mit *Word* oder (besser) mit dem Windows-*Editor* oder *Notepad* an.

4.
Öffne die exportierte Datei mit *Editor* oder *Notepad*. Füge vor die Nullen und Einsen eine erste Zeile ein, in der nur *P1* steht, und eine zweite Zeile, in der nur *6 8* steht (mit einem Leerraum dazwischen). Speichere die Datei unter dem Namen *liste.pbm* ab. Öffne sie jetzt mit einem professionellen Grafikprogramm (z.B. *Irfanview*) und überzeuge dich davon, dass das gescannte Bild zu sehen ist.

5.
a) Ändere den Scanner, so dass er automatisch (also ohne wiederholtes Drücken der Leertaste) das Bild scannt.
b) Erweitere den Scanner so, dass er den gesamten Bildschirm scannt.

Abbildung 4: Mit der PRIMM-Methode erstellter Arbeitsbogen

Am Ende des Unterrichtsabschnitts verfügen die Schülerinnen und Schüler über funktionsfähige Scan- und Zeichenprogramme, die Rastergrafiken erzeugen und auswerten können. Die Rastergrafiken werden zudem in einem Format erstellt, das auch professionelle Grafikprogramme erkennen. Es handelt sich also um „richtige“ Grafiken, die Schülerinnen und Schüler verlassen damit die ansonsten künstlich abgegrenzte Welt von Scratch.

Beispiel 2: „Wie verläuft eine Epidemie?“

Dieses Beispiel stammt aus dem Projektteil der Unterrichtseinheit. Die mit dem genetischen Ansatz gewonnenen Konzepte werden jetzt in anderem Kontext *angewendet*. Die Schülerinnen und Schüler bearbeiten im Sinne des *Cognitive Apprenticeship*-Ansatzes (siehe oben) nun

weitgehend selbständig ein Projekt, erhalten jedoch bei Bedarf eine dem Leistungsvermögen angepasste Unterstützung. Es kann zwischen sechs Projektthemen gewählt werden, die alle eine außerschulische Fragestellung beinhalten. Abbildung 5 zeigt die Projektaufgabe zum Thema „Epidemien“.

Aufgaben zum Thema 1: Epidemien

Poster

Aufgabe:
Erstelle ein Poster zu *einem* der folgenden Themen:

- Verlauf der Covid-19-Epidemie (z.B.: Wo kam das Virus her? Wie hat es sich verbreitet? Welche Schutzmaßnahmen gab es? Wie ist die Situation heute?)
- Ausbreitung der Pest im Mittelalter (z.B.: Wie wird Pest verursacht und übertragen? Welche Pestwellen gab es in Europa? Wie wurde die Pest besiegt?)
- Viren (z.B.: Was sind Viren? Wie verbreiten sie sich? Wie kann man die Verbreitung verhindern?)
- Mein Leben während der Covid-Epidemie (z.B.: Persönliche Erfahrungen, Einschränkungen, Umgang mit Schutzmaßnahmen)
- Selbstgewähltes Thema aus dem Bereich „Epidemien“ (mit der Lehrkraft absprechen)

Programme

Den Verlauf einer Epidemie kann man mit dem Computer *simulieren*, das heißt, in vereinfachter Form nachbilden. Die Abbildung deutet eine solche Simulation an: Objekte bewegen sich in einem begrenzten Raum. Grüne Objekte stellen gesunde, rote Objekte kranke Menschen dar. Trifft ein grünes Objekt ein rotes, wird es selbst rot

Grundaufgabe:
Entwickle mit Scratch eine solche Simulation. Dabei soll gelten:

- „Kranke“ Objekte bleiben krank, es gibt keine Heilung.
- Die Geschwindigkeit der Objekte soll einstellbar sein.

Erweiterungsaufgaben:
Erweitere dein Programm, zum Beispiel so:

- Die Anzahl der Objekte in dem Raum soll einstellbar sein.
- Kranke werden nach einer bestimmten Zeit wieder gesund.
- Geheilte sind für eine bestimmte Zeit immun und können erst nach Ablauf dieser Zeit wieder angesteckt werden.
- Denk dir selbst eine Erweiterung aus und besprich sie mit der Lehrkraft.

Abbildung 5: Projektaufgabe zum Thema "Epidemien"

Für jedes Projekt fertigen die Schülerinnen und Schüler zunächst *Poster* an, mit denen der Bezug zur außerschulischen Wirklichkeit hergestellt wird. Die erstellten Poster können am Ende des Projektes zum Beispiel im Rahmen eines *Museumsrundgangs* (Klippert 2023) präsentiert werden.

Danach wird eine *Programmieraufgabe* gestellt, und zwar in Form einer Grundaufgabe mit Erweiterungsmöglichkeiten, die je nach Interessen und Fähigkeiten in Angriff genommen werden können. Im Falle der Epidemie-Themas geht es darum, den Verlauf einer Epidemie zu *simulieren*. Die Schülerinnen und Schüler können selbständig arbeiten oder aber in zwei Stufen *Hilfezettel* verwenden, die sie bei der Lösung des Problems unterstützen. Durch diese Differenzierung soll sowohl Über- als auch Unterforderung vermieden werden. Die erstellten Programme können zum Beispiel auch in die abschließende Ausstellung eingehen oder im Rahmen einer Präsentation gezeigt werden.

Ein Angebot zum Ausprobieren

Wie eingangs erwähnt, orientiert sich der Unterrichtsvorschlag zwar an Vorgaben aus Schleswig-Holstein, fügt sich jedoch in die Lehrpläne anderer Bundesländer zwanglos ein, da die blockbasierte Programmierung und das informatische Problemlösen ein typischer Bestandteil von modernem Informatikunterricht sind.

Wir hoffen, dass dieser Beitrag dazu anregen kann, das Material teilweise oder im Ganzen auszuprobieren, und würden uns über Rückmeldungen freuen.

Link zum Unterrichtsmaterial

Das Material findet man unter folgender URL:

https://oer.ipn.uni-kiel.de/edu-sharing/components/collections?_viewType=1&id=3d505196-248c-4404-9e5a-1029f141aa33

Quellen

Arbeitskreis »Bildungsstandards« der Gesellschaft für Informatik e. V. (2008): Grundsätze und Standards für die Informatik in der Schule. Bildungsstandards Informatik für die Sekundarstufe I. In: Log in 28 (150/151).

Bell, Tim; Vahrenhold, Jan (2018): CS Unplugged--How Is It Used, and Does It Work? In: Hans-Joachim and Komm Dennis and Unger Walter Böckenhauer (Hg.): Adventures Between Lower Bounds and Higher Altitudes: Essays Dedicated to Juraj Hromkovič on the Occasion of His 60th Birthday. Cham: Springer International Publishing, S. 497–521. Online verfügbar unter https://doi.org/10.1007/978-3-319-98355-4_29, zuletzt geprüft am 20.2.2024.

Best, Alexander; Borowski, Christian; Büttner, Katrin; Freudenberg, Rita; Fricke, Martin; Haselmeier, Kathrin et al. (2019): Kompetenzen für informatische Bildung im Primarbereich. Bonn: Gesellschaft für Informatik e.V.

Braune, Gert; Mühling, Andreas (2020): Learning to program. In: Proceedings of the 15th Workshop on Primary and Secondary Computing Education. New York, NY, USA: Association for Computing Machinery (WiPSCE '20).

Buttke, Robby; Engelmann, Lutz; Forman, Franz X. (Hg.) (2011): Informatik S I. Informatische Grundbildung; [Klassen 7 - 10]. 1. Aufl., 3. Dr. Berlin: Duden-Paetec-Schulbuchverl. (Duden).

Curzon, Paul (2013): cs4fn and computational thinking unplugged. In: Michael E. Caspersen, Maria Knobelsdorf und Ralf Romeike (Hg.): Proceedings of the 8th Workshop in Primary and Secondary Computing Education. the 8th Workshop in Primary and Secondary Computing Education. Aarhus, Denmark. New York, NY: ACM, S. 47–50.

Grover, Shuchi (Hg.) (2020): Computer science in K-12. An A to Z handbook on teaching programming. Palo Alto, CA: Edfinity.

Hubwieser, Peter; Mühling, Andreas; Aiglstorfer, Gerd (2013): Fundamente der Informatik. Funktionale, imperative und objektorientierte Sicht, Algorithmen und Datenstrukturen. 2nd ed. Berlin/Boston: De Gruyter.

Humbert, Ludger; Puhlmann, Hermann (2004): Essential Ingredients of Literacy in Informatics. In: Johannes Magenheim und Sigrid Schubert (Hg.): Informatics and Student Assessment. Concepts of Empirical Research and Standardisations of Measurement in the Area of Didactics of Informatics : GI-Dagstuhl-Seminar, September 19 - 24, 2004, Schloß Dagstuhl, Germany. Bonn: German Informatics Soc. (GI) (GI-Edition Seminars, 1), S. 65–76.

IQSH - Institut für Qualitätsentwicklung an Schulen Schleswig-Holstein (2022): Pilotphase Pflichtfach Informatik. Online verfügbar unter https://fachportal.lernnetz.de/sh/faecher/informatik/aktuelles/pilotphase-pflichtfach.html, zuletzt geprüft am 20.2.2024.

K-12 Computer Science Framework Steering Committee (2016): K-12 Computer Science Framework. Online verfügbar unter http://www.k12cs.org, zuletzt geprüft am 20.2.2024.

Klippert, Heinz (2023): Methoden-Kartei für die Sekundarstufe. 48 Lernkarten mit Kurzbeschreibung und Bild zu jeder Methode. 3. Auflage. Augsburg: Klippert Verlag in der AAP Lehrerfachverlage GmbH.

Koubek, Jochen; Schulte, Carsten; Schulze, Peter; Witten, Helmut (2009): Informatik im Kontext (IniK) – Ein integratives Unterrichtskonzept für den Informatikunterricht. In: Bernhard Koerber (Hg.): Zukunft braucht Herkunft. 25 Jahre „INFOS - Informatik und Schule" ; INFOS 2009, 13. GI-Fachtagung „Informatik und Schule", 21. bis 24. September 2009 an der Freien Universität Berlin. Bonn: Ges. für Informatik (GI-Edition Proceedings, 156), S. 268–279.

Land Schleswig-Holstein (2021): Fachanforderungen Informatik. Online verfügbar unter https://fachportal.lernnetz.de/files/Fachanforderungen%20und%20Leitf%C3%A4den/Sek.%20I_II/Fachanforderungen/21-14520%20Fachanforderungen%20Informatik%20SEK_WEB_PDF%20UA.pdf, zuletzt geprüft am 20.2.2024.

Lave, Jean; Wenger, Etienne (2011): Situated learning. Legitimate peripheral participation. 24. print. Cambridge: Cambridge Univ. Press (Learning in doing).

Leuders, Timo; Hußmann, Stephan; Barzel, Bärbel; Prediger, Susanne (2011): "Das macht Sinn!". Sinnstiftung mit Kontexten und Kernideen. In: Praxis der Mathematik 53 (37), S. 2–9.

Meisner, Julia (2023): In Deutschland erhalten zu wenige Schülerinnen und Schüler Informatikunterricht. Hg. v. GI. Online verfügbar unter https://idw-online.de/en/news822386, zuletzt geprüft am 20.2.2024.

Möller, Kornelia (2001): Genetisches Lehren und Lernen - Facetten eines Begrifffs. In: Diethard Cech, Bernd Feige, Joachim Kahlert, Helmut Schreier, Hans-Joachim Schwier und Ute Stoltenberg (Hg.): Die Aktualität der Pädagogik Martin Wagenscheins für den Sachunterricht. Walter Köhnlein zum 65. Geburtstag. Bad Heilbrunn: Klinkhardt, S. 15–30.

Resnick, Mitchel; Maloney, John; Monroy-Hernández, Andrés; Rusk, Natalie; Eastmond, Evelyn; Brennan, Karen et al. (2009): Scratch. Programming for All. In: Commun. ACM 52 (11), S. 60–67. DOI: 10.1145/1592761.1592779.

Schuster, Jan (2011): Ein genetischer Zugang zum Programmieren mit CGI-Skripten in Python. In: Informatik in Bildung und Beruf – INFOS 2011 – 14. GI-Fachtagung Informatik und Schule. Bonn: Gesellschaft für Informatik e.V, S. 227–236.

Sentance, Sue; Waite, Jane; Kallia, Maria (2019): Teaching computer programming with PRIMM. A sociocultural perspective. In: Computer Science Education 29 (2-3), S. 136–176. DOI: 10.1080/08993408.2019.1608781.

Weinert, Franz E. (2001): Concept of competence: A conceptual clarification. In: Dominique Simone Rychen und Laura Hersh Salganik (Hg.): Defining and selecting key competencies. Seattle: Hogrefe & Huber.

Wing, Jeannette M. (2012): Computational Thinking. Microsoft Asia Faculty Summit. Zianjin, China. Online verfügbar unter https://www.microsoft.com/en-us/research/wp-content/uploads/2012/08/Jeannette_Wing.pdf, zuletzt geprüft am 20.2.2024.

Lizenz

Kontakt

Gert Braune und Andreas Mühling

Arbeitsgruppe Didaktik der Informatik
Leibniz Institut für Pädagogik der Naturwissenschaften und Mathematik & Universität Kiel

gert.braune@email.uni-kiel.de / muehling@leibniz-ipn.de