

Vertiefungsarbeit – Kommentar

programmier abc

Mathias Nater
Breitistrasse 13
8953 Dietikon
mathias.nater@stud.phzh.ch

April 2006

Betreut durch Roland Fischer

Vorwort

Es wurde lange und viel diskutiert, ob und wann man Schülerinnen und Schülern programmieren lehren soll.

Am einen Ende des Spektrums stehen die Befürworter, allen voran Seymour Papert. Sie treten für eine möglichst frühe Einführung in die Informatik ein, weil der geschickte Umgang mit dem Computer das mathematisch-logische Denken fördere. Sie kreieren interaktive Lernumgebungen und spezielle, kindergerechte Programmiersprachen.

Sie werfen der Schule eine zu anwenderorientierte Informatikausbildung vor, die sich auf die Bedienung von womöglich schon bald veralteten Programmen beschränkt. Im Gegenzug fordern sie eine mehr grundlagenorientierte Informatik, die sich mehr mit den Problemen der theoretischen Informatik und der Logik beschäftigt. Die Kinder sollen so ein nachhaltigeres Wissen zum Umgang mit Computern erwerben können.

Auf der anderen Seite scharen sich um Clifford Stoll die Gegner der „Hightech-Kinderzimmer“. Das folgende Zitat von Clifford Stoll aus „LogOut“ (Vgl. STOLL (1999), Seite 78f) fasst ihren Standpunkt am besten zusammen:

Kinder begreifen die Logik eines Programms sehr schnell. Das hat wohl Papert so beeindruckt, dass er um diese Erkenntnis herum ein komplettes Lernsystem gebaut hat. Aber die trockene und vorhersagbare Logik der Computer – sei es nun in Gestalt von Myst, Logo oder Nintendo – ist ganz anders als die Logik, die ein Kind braucht, um sich mit anderen Kindern auseinander zu setzen. So bieten die Computerprogramme viel Vorbereitung für die Welt der Booleschen Algebra – und eine völlig unzureichende Einführung in das Zusammenleben mit anderen Menschen. Computer liefern im Überfluss abstrakte Symbole, aber sie vermitteln nur dürftige Erfahrungen. Brauchen unsere Kinder noch mehr Icons, Markenzeichen und ausgefeilte Schrifttypen – oder sollten sie mehr klettern, herumrennen oder herausfinden, wie man miteinander gut auskommen kann?

Meinen eigenen Standpunkt siedle ich irgendwo dazwischen ein. Ich bin durchaus der Meinung, dass der *Computer als Werkzeug* seinen Stellenwert in der Schule haben muss. Auch ist es unabdingbar, dass die Schülerinnen und Schüler ein Textverarbeitungsprogramm bedienen und ein Tabellenkalkulationsprogramm nutzen können, sowie Informationen im Internet suchen und kritisch bewerten können. Diese Fertigkeiten werden auch

schon als „vierte Kulturtechnik“ (vgl. BUCHER ET AL (2004), S. 3) bezeichnet.

Der aktuelle Umgang mit Computern ist im Bezug auf die Informatik sehr *anwendungsorientiert*. Probleme werden nicht mehr gelöst, in dem man sich selbst überlegt, wie, sondern indem man im Internet eine Lösung sucht.

Ich trete hier für einen vermehrt *produktiv-kreativen Umgang* mit dem Computer ein, in den Bereichen, für die er eigentlich geschaffen wurde: für die Unterstützung unseres Denkens dort, wo unser Hirn zu langsam ist oder zu ungeduldig, um immer wiederkehrende Dinge zu tun. Dazu muss man aber ein wenig programmieren können.

Das heisst nun nicht, dass alle Schülerinnen und Schüler programmieren lernen müssen! Programmieren lernen verlangt viel Geduld und intrinsische Motivation. Deshalb sollten nur Schülerinnen und Schüler damit konfrontiert werden, die sich wirklich dafür interessieren und vielleicht im Rahmen der Berufskunde Erfahrungen im Bereich Informatik sammeln wollen.

Damit bewege ich mich auch im Rahmen der aktuellen Deutschschweizer Lehrpläne, die Programmieren nie als explizites Ziel nennen.¹

Diese Arbeit begleitet eine eLearning-Sequenz, mit der sich interessierte Schülerinnen und Schüler der Sekundarstufe I in die Grundlagen der Programmierung einarbeiten können. Das erste Kapitel widmet sich den Fragen der Programmier-Didaktik, das zweite den Aspekten der eLearning-Didaktik, beides Bereiche, mit denen ich mich im Laufe der theoretischen Vorbereitung für das Verfassen der eLearning-Einheit intensiv befasst habe.

Im dritten Kapitel wird erläutert, wie ich diese theoretischen Erkenntnisse in meinem eLearning-Angebot, dem „ProgrammierABC“ praktisch umgesetzt habe.

Erstellt wurde diese Arbeit als Vertiefungsarbeit (Diplomarbeit) an der Pädagogischen Hochschule Zürich.

¹Eine Ausnahme bildet die Lehrplanergänzung der BKZ zu ICT an der Volksschule, die Programmieren als möglichen Inhalt eines Informatik-Wahlfaches im 9. Schuljahr nennt.

Inhaltsverzeichnis

Vorwort	ii
1 Aspekte der Programmier-Didaktik	1
1.1 Einführung	1
1.2 Begriffe	1
1.3 Verschiedene Ansätze der Programmier-Didaktik	2
1.4 Programmieren in der Volksschule – warum, wann und wie?	3
1.5 Programmierparadigmen und -sprachen, Elemente einer Programmier- sprache	4
1.6 Programmieren in der Volksschule – Inhaltsanalyse	5
1.7 Programmier-Lernumgebungen	7
1.8 Zusammenfassung	11
2 Aspekte der eLearning-Didaktik	13
2.1 Einführung	13
2.2 Erstellen von eLearning-Angeboten	16
2.3 Einsatz von eLearning-Angeboten im schulischen Unterricht	20
2.4 Zusammenfassung	23
3 ProgrammierABC	24
3.1 Einführung	24
3.2 Zielgruppe(n)	24
3.3 Überlegungen rund ums ProgrammierABC	24
4 Schlussgedanken	31

1 Aspekte der Programmier-Didaktik

1.1 Einführung

Die Frage nach dem Wann und Wie man Schülerinnen und Schülern oder Studierenden das Programmieren beibringen soll wird schon lange diskutiert. Seit den Anfängen der Informatik wurden Umgebungen, Spiele und sogar eigens Sprachen kreiert, um den schwierigen Einstieg in die Welt der Programmierer zu erleichtern. Im Folgenden will ich verschiedene Ansätze kurz beleuchten.

Auch die Frage nach dem „Ob“ spielt eine wichtige Rolle. Allerdings wird diese Frage von den Lehrplänen klar beantwortet. Ich gehe deshalb davon aus, dass in der Volksschule nur interessierte Schülerinnen und Schüler mit dem Thema konfrontiert werden. Deshalb wird hier nicht mehr näher darauf eingegangen.

Will man „Programmieren“ unterrichten, sieht man sich mit einer Reihe von Fachbegriffen und fachlichen Fragen konfrontiert, die sich damit beschäftigen, was denn nun vermittelt werden soll. Diese Begriffe werden kurz erläutert und Gedanken zum Inhalt zusammengefasst.

Schliesslich werden einige bereits bestehende – zum Teil berühmte – Programmier-Lernumgebungen betrachtet.

1.2 Begriffe

1.2.1 Was ist ein Programm?

Vereinfacht gesagt ist ein Programm eine genaue Beschreibung von Ereignissen, die in einer bestimmten Reihenfolge ablaufen. Wir werden im Alltag mit vielen Programmen konfrontiert. Im weiteren Sinne sind also auch Gebrauchsanweisungen oder Kochrezepte Programme.

Im Zusammenhang mit Computern versteht man unter einem Programm eine genaue Abfolge von Befehlen, die der Computer – je nach Eingabe des Benutzers – durcharbeitet. Meist wird ein Algorithmus so umschrieben, dass er für den Computer verständlich ist (Implementierung).

1.2.2 Was bedeutet Programmieren?

Programmieren ist mehr als das Schreiben eines Programms. Programmieren kann man nach STOODLEY ET AL (2004) in fünf Ebenen einteilen:

Coding meint das einfache Eintippen von Programmcode oder das zusammenfügen von Befehlen.

Thinking differently beschreibt die Art und Weise zu „denken“, wie der Computer. Das heisst, Handlungen so zu beschreiben, dass sie aus der Logik des Computers zu verstehen sind.

Problem solving bedeutet, dass ein Problem mit einem Programm gelöst werden kann.

Participating bedeutet, an der Gemeinschaft der Programmierer teil zu haben und Programme so zu schreiben, dass sie auch für andere Programmierer verständlich sind.

Satisfying Clients umfasst alle weiteren Aspekte, die auch zum Programmieren gehören, wie das Erfüllen von Kundenvorgaben und das Vermarkten der Programme.

Die ersten zwei Ebenen sind die anspruchsvollsten, was den Einstieg ins Programmieren alles andere als erleichtert.

1.3 Verschiedene Ansätze der Programmier-Didaktik

Ich unterscheide hier drei verschiedene Ansätze oder Ziele der Programmier-Didaktik: Im „akademischen Ansatz“ wird das Programmieren um der Informatik willen gelehrt. Die Lernenden sollen anhand verschiedener Programmiersysteme allgemein gültige Erkenntnisse zur Informatik gewinnen. Es werden Probleme der theoretischen Informatik¹ aufgegriffen.

Wird Programmieren gelehrt, um das logische, strukturierte Denken zu schulen, so kann man von einem „pädagogischen Ansatz“ sprechen. Auf die Inhalte wird dabei weniger Wert gelegt, dafür spielt der Spass am lernen eine zentrale Rolle. In beiden Ansätzen wird vor allem auf die Ebenen „Thinking differently“ und „Problem Solving“ Wert gelegt. Der „pragmatische Ansatz“ lehrt Programmieren, um Programme erstellen zu können, die uns helfen, Probleme zu lösen oder die uns bei der Arbeit unterstützen. Die Aufgaben stellen sich aus einem weiten Einzugsgebiet. Hier wird meist mit einem einzigen Programmiersystem gearbeitet. Unter diesem Ansatz spielen alle fünf Ebenen eine gleichwertige Rolle, wobei das „Problem Solving“ oft der Forschung überlassen wird.

¹Zum Beispiel: Berechenbarkeit, Turingmaschinen, endliche Automaten, konkurrierende und nebenläufige Prozesse etc.

1.4 Programmieren in der Volksschule – warum, wann und wie?

Warum?

Es ist noch unerforscht, welchen Einfluss es auf die Lernaktivität von Schülerinnen und Schülern hat, wenn sie über grundlegende Programmierkonzepte verfügen. Ich stelle folgende Vermutungen auf:

Schüler und Schülerinnen, die über grundlegende Programmierkonzepte verfügen, ...

... haben ein zusätzliches Instrument in der Hand, das sie beim konstruieren von Wissen unterstützt.

... sind vertrauter im Umgang mit IT-Mitteln.

... können strukturiert denken.

... haben einen anderen Bezug zum Konsumgut Computer.

Eine Überprüfung dieser Thesen würde den Rahmen dieser Arbeit sprengen. Sie zeigen aber einen möglichen pädagogischen Wert.

Schülerinnen und Schüler können also mit gutem Gewissen unterstützt werden, wenn sie sich dafür interessieren. Da programmieren lernen aber viel Zeit, Geduld und eine hohe intrinsische Motivation verlangt, sollte davon abgesehen werden, das Programmieren obligatorisch zu machen. Die oben genannten Bildungswerte liessen sich auch auf anderen Wegen vermitteln.

In folgenden Situation sehe ich es als sinnvoll und vertretbar an, Schülerinnen und Schülern programmieren zu lehren:

- Eine Schülerin, ein Schüler bringt bereits ein grosses Vorwissen in ICT mit. Statt sie/ihn mit Stoff zu unterfordern, den sie/er nachgewiesener weise schon beherrscht, kann sie/er sich selbständig in die Programmierung einarbeiten.
- Schülerinnen und Schüler wollen im Rahmen der Berufskunde Erfahrungen im Bereich Informatik sammeln.

Wann?

Die Antwort auf diese Frage hängt stark vom verfolgten Ansatz ab. Mit einem pädagogische Ansatz kann mit geeigneten Umgebungen schon sehr früh begonnen werden. Die Kinder müssen Kausalzusammenhänge verstehen und bereits über ein gewisses Mass an abstraktem Denkvermögen verfügen.

Sollen konkrete kleine Programme entwickelt werden, muss zusätzlich ein gewisses Grundwissen vorausgesetzt werden können (Variable, Algorithmen).

Am einfachsten hört man auf die natürliche Neugier der Schülerinnen und Schüler und bietet ihnen dann die Möglichkeit, programmieren zu lernen, wenn sie es wollen oder brauchen.

Wie?

Die Frage nach dem Wie ist im Zusammenhang dieser Arbeit die wichtigste. Es gibt viele Fragen die geklärt sein wollen. In den folgenden Abschnitten werde ich auf die verschiedenen technischen Aspekte und auf bereits bestehende Lernumgebungen eingehen, sowie eine ausführliche Analyse der zu vermittelnden Inhalte machen. Daraus lassen sich Anforderungen an eine Programmier-Lernumgebung ableiten.

1.5 Programmierparadigmen und -sprachen, Elemente einer Programmiersprache

Es gibt verschiedene Programmierparadigmen und verschiedene Programmiersprachen. In diesem Abschnitt werden die wichtigsten kurz erläutert.

1.5.1 Paradigmen

Ein Paradigma ist das Prinzip, das der Programmiersprache oder der Programmiertechnik zugrunde liegt. Eine sehr ausführliche Erklärung und Aufzählung ist bei Wikipedia² zu finden. Hier werden nur die für die Schule interessanten Paradigmen behandelt.

Imperative Programmierung Programme sind eine Folge von Befehlen, die vom Computer befolgt werden. Mit bestimmten Befehlen kann in der Folge der Befehle herumgesprungen werden.

Prozedurale Programmierung In der Prozeduralen Programmierung wird das Problem in eine Reihe von Unterproblemen aufgeteilt. Diese Unterprobleme werden in „Funktionen“ nach dem imperativen Paradigma beschrieben.

Objektorientierte Programmierung Die Objektorientierte Programmierung (OOP) versucht die Welt in so genannten Objekten abzubilden. Jedes Objekt kann Eigenschaften und Methoden besitzen. Im Ablauf des Programms können die Objekte erzeugt und verändert werden. Die Methoden eines Objekts werden nach dem Prozeduralen Paradigma beschrieben.

²<http://de.wikipedia.org/wiki/Programmierparadigma>

1.5.2 Programmiersprachen

Programmiersprachen sind künstliche oder formale Sprachen. Das heisst, sie haben eine ganz streng definierte Syntax und eine eindeutige Semantik. Die Folge von Befehlen in einer Programmiersprache nennt man Programm, Programmcode oder Quelltext.

Programmiersprachen erlauben es, den Computer mit Befehlen zu steuern, die der menschlichen Sprache ähnlich sind. Die Programmtexte (Befehlsfolgen) werden dann in die Maschinensprache übersetzt und können vom Computer ausgeführt werden.

Gewisse Programmiersprachen verlangen, dass zuerst der gesamte Quelltext von einem Compiler in die Maschinensprache übersetzt wird, bevor er ausgeführt werden kann. Andere werden während der Ausführung von einem Interpreter übersetzt. Interpretierte Programme sind langsamer als kompilierte. Eine ausführliche Aufzählung der verschiedenen Sprachen ist bei Wikipedia³ zu finden.

In der Definition einer Programmiersprache ist ganz klar geregelt, wie die Befehle lauten müssen. Oft sind die Befehle von Sprache zu Sprache sehr ähnlich. So hat man eine neue Programmiersprache schnell dazugelernt, wenn man eine andere bereits beherrscht.

1.6 Programmieren in der Volksschule – Inhaltsanalyse

1.6.1 Was soll erreicht werden?

Schülerinnen und Schüler sollen mit einer einfachen Programmiersprache verschiedene Probleme aus verschiedenen Fächern lösen können. Dabei sollen technische Schwierigkeiten möglichst ausgeschlossen werden. Die Resultate müssen einfach veröffentlicht werden können.

Es handelt sich dabei also um einen pragmatischen Ansatz mit pädagogischen Hintergedanken.

1.6.2 Welches Paradigma?

Wie oben erwähnt, bauen die verschiedenen Paradigmen aufeinander auf. Mit dem Ziel der objektorientierten Programmierung sollte man mit der prozeduralen Programmierung einsteigen und so die Imperative Programmierung gleich unterwegs mitnehmen.

1.6.3 Welche Programmiersprache in welcher Umgebung?

Die Wahl der Programmiersprache wird von vielen Faktoren beeinflusst. Der akademische Ansatz verlangt eine Sprache, die den Programmablauf graphisch visualisiert und

³<http://de.wikipedia.org/wiki/Programmiersprache>

zu jedem Zeitpunkt genaue Auskunft über den Zustand des Programms gibt. Die Datentypen und syntaktischen Strukturen sollen unmissverständlich und eineindeutig sein, was die intellektuelle Durchdringung des Problems *vor* dem Schreiben des Programms verlangt.

Der pragmatische und der pädagogische Ansatz sind da unkomplizierter, da das Produkt mehr zählt, als der Weg dazu. Eine einfachere, schnelle Sprache wird hier bevorzugt.

Für einfache Programmierübungen eignet sich eine interpretierte Sprache besser, als eine kompilierte, da bei einer interpretierten Sprache der Zyklus „Codieren → Testlauf → Codieren“ nicht durch das Kompilieren unterbrochen wird.

Die Programmierumgebung sollte möglichst einfach zu bedienen sein und keine lange Einarbeitungszeit verlangen. Es sollten zudem einfache Mittel zur Ein- und Ausgabe während der Programmausführung zur Verfügung stehen.

Die Programmiersprache (bzw. deren Interpreter) sollte zudem auf den verschiedenen Plattformen ohne zusätzliche Installationen verfügbar sein.

1.6.4 Abfolge der Lernblöcke

Folgende Grafik zeigt die Zusammenhänge der inhaltlichen Blöcke:

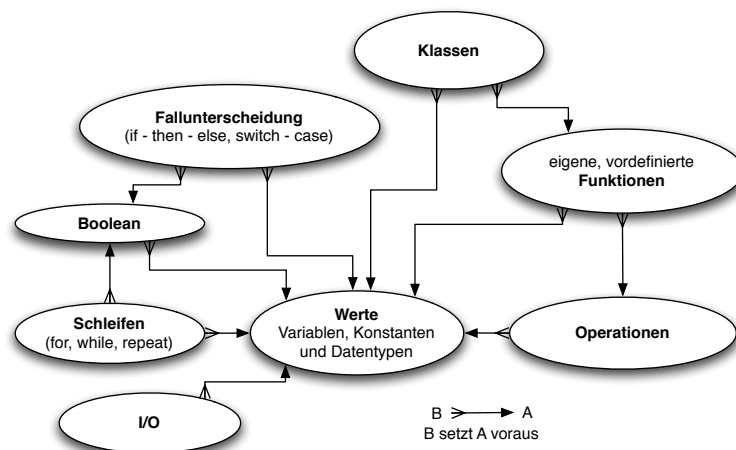


Abbildung 1.1: Abhängigkeiten der verschiedenen inhaltlichen Einheiten

Daraus ergibt sich folgende Reihenfolge:

1. Werte
Verschiedene Datentypen (Zahlen, Zeichenketten), Variablen, Wertzuweisungen

2. I/O
Ein- und Ausgabe von Daten vor und während der Programmausführung
3. Operationen
Operationen der bekannten Datentypen (Zahlen und Zeichenketten): Addition, Subtraktion, Division, Multiplikation, Modulo, Stringverkettung, Typumwandlung
4. Boolean
Vergleichsoperationen und logische Operationen
5. Fallunterscheidungen
„if else“ und „switch case“
6. Schleifen
for, while und do while (repeat)
7. Funktionen
8. Bestehende Befehlsbibliotheken
9. Arrays
10. Objekte

Das Thema Funktionen liesse sich auch bereits nach den Operationen abhandeln. Es ist aber sinnvoller, dies erst am Schluss zu tun, da die Möglichkeiten zur Kapselung grösser sind, wenn die Konzepte der Schleifen und der Fallunterscheidung bereits bestehen.

1.7 Programmier-Lernumgebungen

1.7.1 Logo und Turtle Graphic

Im Zeichen der konstruktivistischen Lerntheorie entstanden in den 1970er Jahren am MIT unter der Leitung von Seymour Papert interessante Projekte, die das Ziel hatten, Kinder bei der Erschliessung mathematischer und geometrischer Zusammenhänge zu unterstützen. Der Computer sollte den Kindern dabei als Werkzeug dienen. Papert verfolgt also den „pädagogischen Ansatz“. In diesem Kontext wurde die Programmiersprache Logo entwickelt. Logo ist eine professionelle Sprache mit entsprechender Komplexität. Für Kinder wurde eine Untermenge der Logo-Befehle definiert, die dann unter dem Namen „Turtle Graphic“ ziemlich berühmt wurde.

In Turtle Graphic wird eine kleine Schildkröte durch Programmierbefehle auf dem Bildschirm umhergesteuert. Dabei kann sie Linien zeichnen, wodurch graphische Bilder entstehen.



Abbildung 1.2: Beispiele für LOGO-Befehle mit ihrer Darstellung in Turtlegraphic. Quelle: <http://el.media.mit.edu/logo-foundation/>

LOGO-Interpreter und Turtlegraphics sind heute Bestandteil vieler Projekte, die sich auch in der Schule einsetzen lassen. Eine Übersicht ist auf <http://el.media.mit.edu/logo-foundation/> zu finden.

Das Konzept der Steuerung eines virtuellen Actors durch Programmierbefehle wurde vielfach adaptiert.

1.7.2 Karel the Robot

Die Idee des Actors wurde in von Richard Pattis 1981 an der Carnegie Mellon University mit „Karel the Robot“ weitergeführt. Auch hier wird der Actor durch Programmierbefehle gesteuert. Das Ziel ist aber nicht, Grafiken zu zeichnen, sondern in einer virtuellen Welt bestimmte Aufgaben zu erfüllen. Karel lebt im Gegensatz zu Paperts Schildkröte in einer eingeschränkten Welt mit Wänden und Hindernissen. Ausserdem hat er Sensoren, mit denen er seine unmittelbare Umwelt wahrnehmen kann, um Piepser aufzusammeln. Der spielerische Charakter ist damit grösser.

Pattis ging es um das Lösen von Problemen mittels strukturiertem Vorgehen. Er verfolgt somit eine Mischung aus pädagogischem und paradigmatischem Ansatz.

Bei der Einführung ins Programmieren mit Karel werden die typischen Strukturen von Programmiersprachen schrittweise eingeführt. Dabei verwendet er eine Pascal-ähnliche Programmiersprache.

Auch für Karel the Robot gibt es viele verschiedene Umsetzungen und viele Nachfolger, die zum Teil auch die objektorientierte Programmierung zu vermitteln suchen.

1.7.3 Kara

Einen anderen Weg schlägt ein Projekt der ETHZ ein:

Kara ist ein Marienkäfer, der in einer einfachen Welt lebt. Er kann programmiert werden und so diverse Aufgaben erledigen, zum Beispiel Kleeblätter sammeln. Kara's Programme sind endliche Automaten und werden in einer grafischen Entwicklungsumgebung erstellt.

Kara vermittelt einen Einstieg in die Grundideen der Programmierung. Zwei Eigenschaften machen den Einstieg mit Kara attraktiv: Endliche Automaten sind einfach zu verstehen, die Einarbeitungszeit ist daher minimal. Zudem arbeitet man bei Kara in einer einfachen grafischen Umgebung, ohne mit komplexen Entwicklungsumgebungen konfrontiert zu werden.

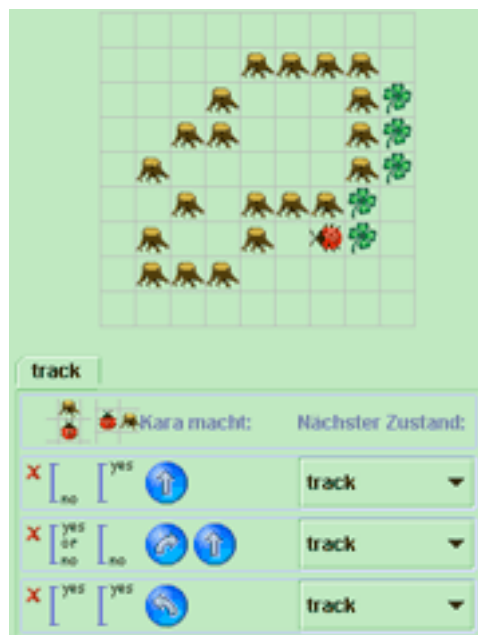


Abbildung 1.3: Kara in seiner Welt: Sammle alle Kleeblätter ein! Quelle: <http://www.swisseduc.ch/informatik/karatojava/kara/>

Kara verfolgt zwei Ziele:

- *Grundlagen der Programmierung* Ein Programm als Spezifikation eines eindeutig bestimmten zeitlichen Ablaufes von Ereignissen verstehen. Im Programm auf externe Ereignisse reagieren. Konzepte wie Boole'sche Aussagenlogik und Programmverifikation erfahren.

- *Modell der endlichen Automaten* Endliche Automaten sind eines der einfachsten Berechnungsmodelle der Informatik. Mit ihnen kann das Verhalten vieler aus dem Alltag bekannter Automaten beschrieben werden. Kara veranschaulicht das Konzept endlicher Automaten.

Angegliedert an Kara gibt es weitere Umgebungen mit eigens definierten Zielen:

- *JavaKara* illustriert die grundlegenden Konzepte imperativer / prozeduraler Programmiersprachen: von einfachen Methoden-Aufrufen über Verzweigungen, Schleifen bis hin zu eigenen Methoden mit Variablen, Parametern und Rückgabewerten.
- *MultiKara* illustriert die Grundlagen von nebenläufigem Programmieren und zeigt auf, dass bei nebenläufigen Prozessen spezielle, den einzelnen Prozessen übergeordnete Synchronisationsmechanismen benötigt werden.
- *TuringKara* illustriert typische Aufgaben rund um Turing-Maschinen.
- Mit *LegoKara* lässt sich ein Lego Mindstorm Roboter programmieren.

Je nach Umgebung werden verschiedene Ansätze verfolgt. Kara, TuringKara und MultiKara folgen klar dem „akademischen Ansatz“, während JavaKara das Programmieren in den Vordergrund stellt und somit dem „paradigmatischen Ansatz“ folgt.

1.7.4 ProgrammierABC

Während die drei obengenannten Umgebungen – auch wenn mit unterschiedlichen Zielen – alle das Konzept eines programmierbaren Actors umsetzen, verfolgt das ProgrammierABC einen weniger spielerischen Ansatz.

Ziele des ProgrammierABC sind einerseits das Vermitteln der wichtigen Konzepte der Programmierung und andererseits auch das Lösen von mathematischen Problemen mit einem Programm.

Dazu geht es weniger problemorientiert vor (wie Karel und Kara), wobei am Anfang ein Problem steht, das man dann lösen muss, als vielmehr ressourcenorientiert, wobei zuerst die Ressourcen vermittelt werden, die dann in entsprechenden Problemen angewendet werden können.

Dadurch ist es möglich, mit wirklichen Problemen zu arbeiten, statt kleiner Käfer zu steuern. Mögliche Probleme sind Fragen der Zahlentheorie (ggT, kgV), mathematisierte Rätselaufgaben oder Probleme aus dem Alltag wie Codierung oder Prüfsummen (Kreditkartennummern, AHV-Nummern etc.)

Solche Probleme sind mögliche Bestandteile des Mathematikunterrichts.

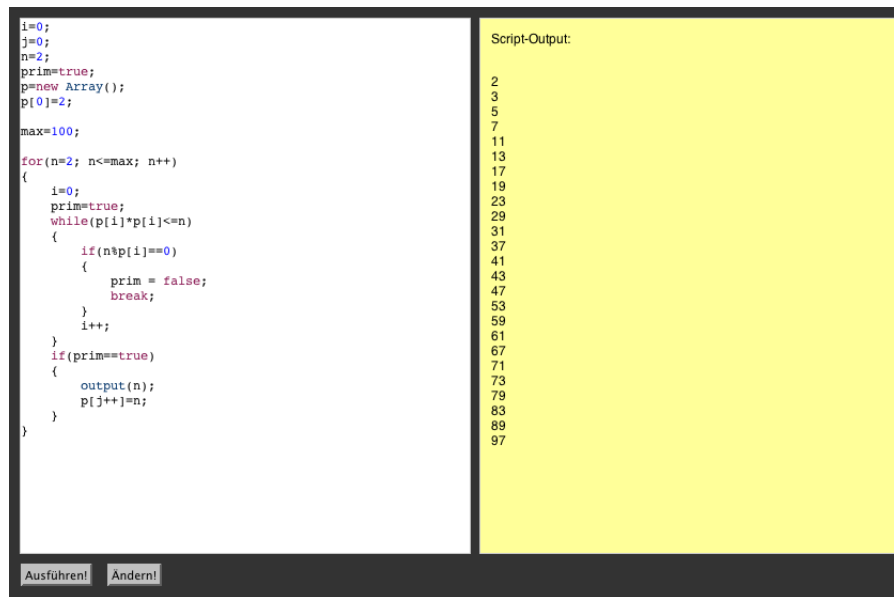


Abbildung 1.4: Das CodeLab im ProgrammierABC

Der Spieltrieb kann dann befriedigt werden, wenn es darum geht, bekannte Spiele als Computerspiele zu programmieren oder sogar eigene kleine Spiele zu entwickeln.

Die im ProgrammierABC verwendete Programmiersprache ist JavaScript, die durch einige Funktionen erweitert ist. Die von den Lernenden erstellten Programme können dank der weiten Verbreitung von JavaScript einfach im Internet veröffentlicht werden.

1.8 Zusammenfassung

Die programmierdidaktischen Aspekte sind sehr unterschiedlich und von den verschiedenen Lerntheorien mitgeprägt. Programmieren wird in den Schulen selten gelehrt und das Fach ist im Vergleich relativ jung. Es fehlen also die langzeitlichen Erfahrungen mit der praktischen Umsetzung dieser theoretischen Ansätze. Grundsätzlich werden mit dem „akademischen“, dem „pädagogischen“ und dem „paradigmatischen“ Ansatz drei verschiedene Grobziele verfolgt.

Die wenigen hier erwähnten Lernumgebungen verfolgen diese verschiedenen Ziele mit verschiedenen Mitteln. Allen gemeinsam ist aber der „Learning by doing“ oder – nach Papert – der „Learning by making“-Ansatz, der von den lernenden eine direkte Aktion verlangt, die mit Erfolg oder Misserfolg verbunden ist.

Der spielerische Charakter ist wie die didaktischen Unterstützungen und Anleitungen

unterschiedlich stark ausgeprägt.

Auch die Zugänglichkeit der verschiedenen Umsetzungen ist ganz unterschiedlich. Die meisten verlangen eine Installation auf dem lokalen System und sind z.T. grafisch wenig ansprechend. Leider haben einige Umgebungen mit der technologischen Entwicklung nicht mitgehalten.

2 Aspekte der eLearning-Didaktik

2.1 Einführung

Setzt man sich mit eLearning und der dazugehörigen Didaktik auseinander, gilt es ein weites Spektrum von Fragen aus verschiedenen Gebieten zu behandeln.

Auf der einen Seite stehen die technischen Aspekte, die durch immer modernere Technologien ständigem Wandel unterworfen sind. Zu fragen ist, welche Technologien beim lernen tatsächlich helfen und worauf man verzichten kann.

Auf der anderen Seite stellen sich menschliche Fragen. Wie wichtig ist der soziale Rahmen einer Klasse von Lernenden mit einer realen Bezugsperson beziehungsweise Lehrperson? Welche Chancen, aber auch welche Risiken birgt eLearning?

Hat man sich einmal für das eLearning – in welchem Rahmen und aus welchen Gründen auch immer – entschieden, stellen sich weitere Fragen. Je nachdem ob man eine eLearning-Umgebung erstellen will oder ein bereits vorhandenes Angebot im Unterricht einsetzen will, sind diese Fragen eher technischer und didaktischer Natur oder aber Fragen zur Methodik und zur Rollenverteilung zwischen Lehrperson, Lernenden und der eLearning-Umgebung.

Dieses Kapitel setzt sich mit diesen Fragen auseinander.

2.1.1 eLearning-Definition

eLearning ist ein weiter Begriff, was nur schon die vielen mehr oder weniger gleichbedeutenden Begriffe und Abkürzungen zeigen.

eLearning¹ steht für „Electronic Learning“, also „elektronisches Lernen“ – wahrscheinlich in Analogie zu E-Mail. Es umfasst jede Form computerunterstützten Lernens. Typische Merkmale sind:

- Lernsysteme und Lerninhalte sind digitalisiert
- Einsatz verschiedener Medien (Text, Bild, Ton, Film) im Verbund (Multimedialität)

¹Ich verwende dieselbe Schreibart, wie sie auch von Dozenten an der PHZH verwendet werden. Oft wird auch E-Learning geschrieben.

- Vernetzung der Lerninhalte untereinander (Hypermedialität)
- Interaktivität zwischen den verschiedenen Parteien (Lernende, Lehrperson, Lernumgebung)
- Verfügbarkeit unabhängig vom Stundenplan
- Abhängigkeit von der Verfügbarkeit eines Computers (oft mit Internetanschluss)

Somit gehören bildende Angebote wie zum Beispiel das Schulfernsehen nicht in diese Kategorie, obwohl auch sie elektronisch verbreitet werden. Auch digitale Lexika zähle ich nicht zu den eLearning-Systemen, weil sie kein explizites Lehrziel verfolgen.

2.1.2 Entwicklung des eLearning

Die Idee des elektronischen Lernens geht zusammen mit Erkenntnissen in der Lernpsychologie auf die amerikanischen Bemühungen in den 50er-Jahren zurück, neue effektive und kostengünstige Lernmodelle zu finden. Die Arbeit mit den damals entwickelten „Lernmaschinen“ wurde als programmierter Unterricht bezeichnet.

In den 80er- und 90er-Jahren wurden diese Unterrichtstechnologien, vor allem die des individuellen Lernens, von der Industrie für die Mitarbeiterfortbildung übernommen. Der Begriff des „eLearning“ stammt aus dieser Zeit. Es handelte sich meist um digitalisierte Lehrsysteme auf CD-ROM, welche grosse Firmen und auch die Armeen für ihre vielen Angestellten entwickelten, um neue Produkte und Verfahren kostengünstig, flächendeckend und einheitlich einzuführen.

Seit der schnellen Verbreitung des Internets und mit der immer grösser werdenden Bandbreite der Anschlüsse, werden eLearning-Systeme zunehmend über das Internet zugänglich gemacht. Dies bringt weitere Änderungen mit sich: Das Lernen mit eLearning-Systemen ist unabhängig von Zeit und Ort (Computer mit Internetanschluss vorausgesetzt) und es werden neue Kommunikationskanäle wie Online-Foren, Chats und E-Mail benutzt.

Seit auch die Volksschulen über Computer verfügen, ist eLearning in seiner modernen Form wieder Thema für die Schulen geworden. Modernen Lehrmitteln liegen je länger je mehr auch Lernprogramme und andere digitalisierte Medien bei oder sie werden von einem eLearning-Angebot begleitet.

Der eLearning-Boom ist zur Zeit gross. Auch an den Hochschulen stehen den Studenten immer mehr eLearning-Systeme zur Verfügung und es wird – leider aus Kostengründen – manchmal sogar auf die Vorlesungen verzichtet...

2.1.3 Chancen und Risiken

Die Unterstützer und Autoren von eLearning-Systemen werden natürlich nicht müde die Vorteile zu nennen.

- + Eine persönliche Betreuung ist möglich (im Gegensatz zu anonymen Massenvorlesungen).
- + Man lernt die Inhalte dann, wenn man sie braucht und nicht auf Vorrat (Just-in-time)
- + Berücksichtigung des individuellen Lerntempos
- + Weitgehende örtliche und zeitliche Flexibilität
- + Selbstgesteuertes und selektives Lernen
- + Jeder Lerntyp kann berücksichtigt werden
- + Interessante und motivierende Aufbereitung des Lernstoffes durch Multimedia möglich
- + Hohe Aktivität der Lernenden
- + Standortbestimmung jederzeit möglich
- + Konstante Qualität des Inhaltes
- + Verkürzung der Lerndauer durch individuelle Selektivität (Meidung von Lernredundanzen)
- + Die Multimedialität erhöht die Behaltensquote

Gerade in Zeiten des Booms, wo vor allem die Vorteile herausgestrichen werden, sollte man sich auch der Nachteile von eLearning bewusst werden.

- Hohe technische Anforderungen und Störanfälligkeit
- Technische Pannen und technisch bedingte Wartezeiten mindern die Motivation
- Hoher Aufwand bei der Erstellung des eLearning-Systems
- Hoher Aufwand bei der Systembetreuung

- Hoher Aufwand bei der Ausbildung der Lehrpersonen
- Man sitzt immer alleine vor dem Computer. Spontane Lerngruppen und Kontakte fallen aus
- Bei eLearning-Angeboten, die über das Internet angeboten werden, gilt: Die Informationen aus dem Internet sind kritisch zu betrachten
- Hohe Anforderungen an die Lernenden (Eigendisziplin, Motivation, Anwendungskompetenz, Lernkompetenz etc.)
- Abhängigkeit vom Computer, Internetanschluss und Strom

Ganz wichtig scheint mir auch festzuhalten, dass nicht alle Inhalte für den Einsatz von eLearning geeignet sind. Nur wenn eLearning die Lerninhalte und Lernziele auch besser vermittelt als die ursprünglichen Methoden, ist der Umstieg sinnvoll. eLearning ist oft kein Ersatz für herkömmliche Lernszenarien.

Persönlich bin ich der Meinung, dass eLearning in der Schule nur dort eingesetzt werden soll, wo sowieso am Computer gearbeitet werden muss. Also zum Beispiel zur Einführung in ein Anwendungsprogramm oder in die Programmierung.

Andere Einsatzmöglichkeiten sehe ich nur noch beim Auffrischen von schon vorhandenem Wissen.

2.2 Erstellen von eLearning-Angeboten

Die Erkenntnisse, die ich während meiner Vorbereitung für das Erstellen einer eigenen eLearning-Umgebung gewonnen habe, sollen in diesem Kapitel kurz zusammengefasst werden.

2.2.1 Bedürfnisabklärung, Zielgruppe, Lehrziel

Am Anfang steht die Bedürfnisabklärung. Dazu gehört auch eine Analyse bereits verfügbarer Produkte und eventuell eine Abgrenzung diesen gegenüber. Zentral ist die Frage, ob das spätere Produkt wirklich benötigt wird oder ob das selbe Resultat nicht auch auf anderen Wegen besser erreicht werden könnte.

Ausserdem ist eine Zielgruppe klar zu definieren und entsprechende Lehrziele zu formulieren.

2.2.2 Lerntheoretische Hintergründe

Bestehenden eLearning-Angeboten liegen verschiedene Lerntheorien zu Grunde. Manche (z.B. Vokabeltrainer) folgen den behavioristischen Theorien, indem sie den Lernenden positiv unterstützen. Es handelt sich dabei meist eher um Lernhilfen, als um Wissen generierende Programme.

Andere, meistens Angebote zu naturwissenschaftlichen Themen, folgen dem Kognitivismus, indem sie Inhalte didaktisch aufbereitet und in vorgegebener Reihenfolge dem Lernenden präsentieren.

Moderne eLearning-Angebote orientieren sich vermehrt an den konstruktivistischen Theorien. Nicht selten kommt es zu Mischformen, dem „Blended Learning“, dass sich aus den verschiedenen Theorien das Beste holt.

Als Verfasser sollte man sich bewusst sein, auf welchen Theorien man aufbaut und die Erkenntnisse sinnvoll und zweckmässig einsetzen.

Ein nach meiner Meinung guter Ansatz zur Verschmelzung dieser Theorien ist das PADUA-Konzept. Es stellt die Lernenden jeweils vor ein Problem (P), dass sie mit ihrem aktuellen Wissen noch nicht oder nur ungenügend lösen können. Darauf folgt ein aufbauender (A) Wissensinput durch eine Lehrperson oder durch selbständige Aufarbeitung. Das neu erworbene Wissen wird durchgearbeitet (D) und geübt (U) und schliesslich im Zusammenspiel mit schon früher erworbenem Wissen angewendet (A).

2.2.3 Ansprüche an eine eLearning-Sequenz

Da viele eLearning-Systeme auf den Markt kommen, darunter leider auch viel unbrauchbare Ware, ist die Qualitätskontrolle streng. Bei der Erstellung können aber die gängigen Qualitätsmerkmale und -prüfungspunkte mit einbezogen werden, was hilft, Qualität zu schaffen.

MÜLLER (2003) nennt für die Gestaltung eines eLearning-Angebots, das ein selbständiges Erarbeiten von Inhalten ermöglicht, folgende sechs Richtlinien:

- Das Medium soll nicht nur äusserlich attraktiv gestaltet sein, sondern auch zu Lernaktivitäten motivieren.
- Das Thema eines Lernangebots muss sofort ersichtlich sein, aber auch, wie das System zu nutzen ist.
- Das eLearning-Angebot soll Wiederholungen anbieten, d. h. es muss ein Pool von verschiedenen Darstellungen, Aufgaben etc. vorliegen. Es reicht nicht aus, wenn einzelne Informationen einfach abgespeichert und abrufbar sind.

- Es sollen zudem Verweise und Links zu anderen Lernangeboten vorhanden sein und so zu Vertiefungen motivieren.
- Die neuen Kommunikationstechnologien sollen zum Tragen kommen. Nebst E-Mail sollen die Möglichkeiten zu Gruppenarbeiten, Veröffentlichung von Arbeitsergebnissen, Diskussionsforen, Chat (moderiert, nicht moderiert), Pinwand etc. angeboten und genutzt werden.
- Die neuen Informations- und Kommunikationsmöglichkeiten werden als Werkzeuge verstanden und nicht als statische Wissensbehälter.

Weiter fügt er an:

Wenn immer möglich sollte ein eLearning-Angebot die direkte Zusammenarbeit zwischen den Lerngruppenmitgliedern initiieren und fördern. Zudem zeigt sich in der Praxis, dass eine eLearning-Projektarbeit erfolgreicher verläuft, wenn sich die verschiedenen Gruppenmitglieder mindestens zu Beginn und am Schluss real begegnen, um Verbindlichkeit im Umgang miteinander zu schaffen.

Knobel (KNOBEL (2003)) rückt ausserdem die Wichtigkeit der Kommunikation mit dem „tutoriellen Coach“ und mit den anderen Lernenden in den Vordergrund. Daraus lässt sich schliessen, dass ein gutes eLearning-Angebot auch entsprechende Kommunikationswege bereitstellt.²

2.2.4 Usability

Zusätzlich zu den genannten Ansprüchen sind auch die Usability-Ansprüche zu berücksichtigen. Beim Lesen am Computerbildschirm werden die Augen schneller müde, als beim Lesen auf Papier³ und Konzentration und Motivation lassen schneller nach. Dem kann einerseits mit einer ergonomischen Einrichtung des Arbeitsplatzes teilweise entgegengewirkt werden. Andererseits ist auch die bildschirmgerechte Aufbereitung der Inhalte von zentraler Wichtigkeit.

Gerade im Bereich eLearning, wo Konzentration und Motivation eine wichtige Rolle spielen, sind also die Ansprüche an die Benutzerführung, die Typographie, das Layout und an die Strukturierung der Inhalte besonders hoch. Tabelle 2.1 fasst die wichtigsten Punkte, die zu berücksichtigen sind, zusammen.

²Bei allgemein zugänglichen Angeboten sollte dies ein Online-Forum oder ein Mailkontakt mit dem Autor sein. Im schulischen oder universitären Einsatz können dies auch Treffen in der Klasse oder in der Seminargruppe sein.

³Dieser Umstand könnte sich mit der zunehmenden Auflösung der Computerbildschirme verbessern.

Layout	<input type="checkbox"/> linksbündiger Text oder Blocksatz <input type="checkbox"/> Worttrennungen vermeiden <input type="checkbox"/> die Rechtschreibung einhalten <input type="checkbox"/> konsistent arbeiten <input type="checkbox"/> stabiles, einheitliches und nicht überladenes Layout
Typographie	<input type="checkbox"/> serifenlose oder proportionale Schriften verwenden <input type="checkbox"/> Schriftgröße: 12pt bis 18pt (Richtwerte) <input type="checkbox"/> höchstens zwei verschiedene Schrifttypen <input type="checkbox"/> pro Zeile 60 Zeichen oder 8 Wörter (Richtwert)
Farbe	<input type="checkbox"/> möglichst wenig verschiedene Farben <input type="checkbox"/> nur kompatible Farbkombinationen <input type="checkbox"/> gute Sicht- und Unterscheidbarkeit <input type="checkbox"/> anregende und beruhigende Farben bewusst einsetzen
Multimedia	<input type="checkbox"/> Bilder, Animation, Simulation und Ton gezielt einsetzen <input type="checkbox"/> motivierende Wirkung
Hypertext	<input type="checkbox"/> zu viele komplexe Hypertextstrukturen führen zu Orientierungsproblemen und Navigationsschwierigkeiten <input type="checkbox"/> hierarchische Strukturen verwenden <input type="checkbox"/> transparente Verlinkung <input type="checkbox"/> „tote“ Links vermeiden
Navigation	<input type="checkbox"/> Navigation immer gewährleisten! <input type="checkbox"/> Verwendung einer Navigationshiytory <input type="checkbox"/> Bookmarks <input type="checkbox"/> Suchfunktionen <input type="checkbox"/> graphische Navigationshilfen
Lernhilfen	<input type="checkbox"/> Virtueller Leuchtstift <input type="checkbox"/> virtuelle Textnotizen <input type="checkbox"/> graphische Zusammenfassung (MindMaps)

Tabelle 2.1: Die wichtigsten Punkte der Bildschirm-Usability. Nach KNOBEL (2003), S. 83ff.

2.2.5 Technologien – Anforderungen und Möglichkeiten

Ein eLearning-System soll nach Knobel (KNOBEL (2003) S. 129 ff.) möglichst folgende fünf Teilbereiche abdecken:

- Anwenderseite

- Kommunikations- und Aktivitätsmanagement
- Erstellung des Kursinhalts (Contentmanagement)
- Assessment und Certification Management
- Administration

Es gibt verschiedene fertige Systeme, die diesen Ansprüchen genügen und lediglich mit Inhalten gefüllt werden müssen. Erzeugt man Inhalte im Rahmen eines solchen Systems sind die technischen Möglichkeiten weitgehend vorgegeben.

Anders gestaltet sich das Umfeld für kleinere Projekte. Hier hat man als Autor ein weites Feld von Möglichkeiten, das sich von spezialisierter Software (z.B. Macromedia Authorware) über allgemeine Autorensysteme (HTML) bis hin zu „umgenutzter“ Software (z.B. PowerPoint) erstreckt.

Es gilt im Vorfeld die Anforderungen klar abzuklären und dann das geeignete Werkzeug auszuwählen, wobei auch ein möglicher späterer Ausbau des Angebots zu bedenken ist. Am weitesten verbreitet sind eLearning-Angebote, die auf den gängigen Web-Technologien⁴ aufbauen. Diese haben den Vorteil, dass das Angebot leicht aktualisiert werden kann und auf allen Computern lauffähig ist, sowie mit niedrigen bis keinen Lizenzkosten verbunden ist.

2.3 Einsatz von eLearning-Angeboten im schulischen Unterricht

Der Einsatz von eLearning-Angeboten ist keinesfalls auf den Einsatz im schulischen Unterricht beschränkt. Das Interesse der Industrie wurde bereits erwähnt.

Im Folgenden wird aber nur der Aspekt des eLearnings im schulischen Unterricht betrachtet. Das heisst, welche Unterrichtsformen und welche Rollenverteilung der Einsatz von eLearning mit sich bringt.

Ich möchte betonen, dass ich eLearning keinesfalls als Ersatz für herkömmlichen Unterricht sehe, sondern lediglich als – nur manchmal – sinnvolle Ergänzung.

2.3.1 Unterrichtsformen und Methoden rund um eLearning

Für projektbezogenen, individualisierten Unterricht bietet eLearning viele interessante Möglichkeiten. Schüler und Schülerinnen können bezogen auf ihren individuellen Lernstand und ihre Interessen die Inhalte selbstgesteuert bearbeiten und das Gelernte gegebenenfalls anderen präsentieren.

Dies erfordert entsprechende Methoden, die aber nicht neu erfunden werden müssen.

⁴HTML mit CSS und JavaScript, sowie PlugIns für Video und Animation

Ein Umdenken wird dann nötig, wenn dem Anspruch des eLearnings auf Zeit- und Ortsunabhängigkeit des Lerngeschehens Folge geleistet werden soll. Dies ist aber in den heutigen Schulformen mit verbindlichem Stundenplan nicht möglich.

Nárosy und Riedler (NÁROSY ET AL (2002)) nennen folgende erfolgreiche Anwendungsmodelle von eLearning:

- **E-Learning zur Wissensvermittlung bei neuartigen Themen:** Über das Internet lassen sich neue, komplexe Themen rascher und aktueller verbreiten als mit herkömmlichen Medien. Neuartiges Spezialwissen kann durch Onlinekurse rasch und aktuell an eine breite Zielgruppe weitergegeben werden.
- **E-Learning zur Sicherung eines homogenen Kompetenzniveaus:** Wenn in einer Klasse Lernende mit unterschiedlichen Vorkenntnissen zusammenkommen, kann mit E-Learning-Modulen eine gemeinsame Grundlage hergestellt werden. Auch zur Selbsteinschätzung kann E-Learning hier beitragen.
- **E-Learning als individuelle Vertiefung von Lerninhalten:** Oft steht zu wenig Zeit zur Verfügung, um wirklich alle individuellen Interessen der Lernenden im Unterricht abzudecken. Mit E-Learning-Angeboten können die Schüler selbst bestimmen, wie genau sie sich mit bestimmten Themen auseinander setzen möchten. Eine gemeinsame Ausgangsbasis wird in diesem Fall direkt im Unterricht erarbeitet.
- **E-Learning als Vorbereitung auf den Unterricht:** Diese Variante ist sowohl für technische als auch für inhaltliche Kompetenzen sinnvoll. In der E-Learning-Phase verschaffen sich die Schüler einen Überblick und erwerben Grundkenntnisse; Spezialfragen und weitere Anliegen werden anschließend im Unterricht diskutiert.
- **E-Learning als Auslagerung der Lehrveranstaltung:** Bei Platzmangel vor Ort, wenn die Lernenden lange Anfahrtswege zur Schule in Kauf nehmen müssten, für zeitlich unabhängiges Arbeiten (zum Beispiel im Rahmen von Abendschulen) – in all diesen Fällen kann E-Learning Sinn machen. Gerade wenn E-Learning eine Lehrveranstaltung vor Ort ersetzen sollte, ist aber die persönliche Betreuung (per E-Mail et cetera) ein zentrales Erfolgskriterium! Im Idealfall steht der Lehrende selbst als Tutor zur Verfügung – was für die Schüler wiederum den Vorteil hat, dass sie wirklich individuelle Anliegen mit dem Lehrenden erörtern

können. Doch auch in diesem Modell bietet sich eine Kombination von Präsenz- und E-Learning-Phasen an.

2.3.2 Rolle der Lehrenden

Beim Einsatz von eLearning kommt der Lehrperson die Rolle des/der Stoffvermittelnden abhanden! Stattdessen wird sie zum Lernbegleiter, zum Coach der Schülerinnen und Schüler. Sie übernimmt auch die Moderation zwischenzeitlicher Standortbestimmungen und Reflexionen über den Fortschritt der Lernenden.

Dieser Rollenwechsel behagt nicht allen Lehrpersonen gleich gut. Vielleicht ist es ein Trost zu bedenken, dass man als Lehrperson immer noch die Möglichkeit hat, den Lernerfolg der Schülerinnen und Schüler zu prüfen und gegebenenfalls Massnahmen einzuleiten.

Müller (MÜLLER (2003)) schreibt dazu:

Gefragt sind also Lehrkräfte, die ihren Unterricht vor dem Hintergrund eines konstruktivistischen Lernverständnisses gestalten. Wenn die Lehrpersonen weit gehend bereit sind, ihren Schülerinnen und Schülern die notwendige Eigenverantwortung zu überlassen und sich darüber mit weiteren Lernenden auszutauschen, werden die pädagogischen und technischen Möglichkeiten des eLearning in vollem Umfang zum Tragen kommen. Diese Lehrkräfte werden in ihren Unterricht immer wieder kooperative Lernmethoden einbauen sowie der gemeinsamen Metakognition mit den Lernenden Raum geben. Methoden, die einen individualisierten, der Heterogenität einer Lerngruppe Rechnung tragenden und trotzdem effizienten Unterricht ermöglichen.

2.3.3 Kommunikation und Lerngruppen

Im aktuellen schulischen Umfeld, wo sich alle Beteiligten immer wieder in der Klasse einfinden, ist die Gefahr weniger akut, trotzdem: Die Gefahr der Vereinsamung – auch wenn dieses Wort vielleicht zu dramatisch klingt – ist bei eLearning gross. Dem muss dringend entgegengewirkt werden.

Auch bei eLearning ist der Austausch über die gelernten Inhalte, die kritische Beurteilung des Lernens sowie natürlich die Möglichkeit, bei Unklarheiten Fragen zu stellen enorm wichtig. Es liegt an der Lehrperson entsprechende Plattformen zu schaffen und den Austausch zu fördern und zu moderieren. Dies kann über verschiedene Wege gelingen:

- **Online:** Die Lernenden erhalten die Möglichkeit individuell via E-Mail oder allgemein in einem Online-Forum Fragen zu stellen und Gedanken auszutauschen.

Dieses System wird dem Anspruch auf Zeit- und Ortsunabhängigkeit gerecht, birgt aber den Nachteil der Unpersönlichkeit. Ausserdem handelt es sich um eine asynchrone Form der Kommunikation: auch bei brennenden Fragen muss man unter Umständen lange auf Antworten warten, was dem Lernfortschritt hinderlich ist.

- **Standortbestimmung in der Klasse:** In regelmässigen Abständen finden sich alle Beteiligten zu einer Austauschrunde ein.
- **Lernjournal:** Alle führen ein öffentliches Lernjournal (in einem Heft oder Ordner oder in Form eines Blogs). In regelmässigen Abständen werden die Erfahrungen und Fragen ausgetauscht.
- **Bildung von Lerngruppen:** Es können Niveaugruppen oder Expertengruppen gebildet werden. Die Schülerinnen und Schüler unterstützen sich innerhalb der Gruppe gegenseitig. Mit Gruppen-Ralleys⁵ kann ein künstlicher Wettbewerb erzeugt werden.

2.4 Zusammenfassung

eLearning ist kein Ersatz für herkömmliche Unterrichtsformen sondern lediglich ein neues Mittel von vielen zur Inszenierung von Unterricht. Wie bei allen anderen Methoden gilt es, diese dann anzuwenden wenn sie als sinnvoll und zweckdienlich erscheint. Wegen der hohen Ansprüche an die Infrastruktur und an die beteiligten Personen wird jedoch oft eine andere Methode bevorzugt.

eLearning kommt dann zum Einsatz, wenn die Merkmale gefragt sind, die andere Methoden nicht bieten können, wie die Unabhängigkeit von Zeit und Ort oder die Individualisierung.

Bei der Erstellung von eLearning-Angeboten ist der Autor nicht von den didaktischen Überlegungen entbunden, die er auch bei der Aufbereitung von Inhalten für den herkömmlichen Unterricht anstellen muss. Zusätzlich kommen aber noch hohe Anforderungen an die technische Kompetenz und Fragen zur Gestaltung des eLearning-begleitenden Unterrichts dazu.

⁵Bei der Schlussbeurteilung zählt auch das Resultat der Gruppe, nicht nur das des Individuums

3 ProgrammierABC

3.1 Einführung

Nach den kritischen Gedanken zum Thema eLearning und seiner Rechtfertigung im Schulunterricht stellt sich natürlich als erstes die Frage, ob für das Thema „Programmieren“ ein eLearning-Kurs geeignet ist.

Ich meine entschieden ja. Um programmieren zu lernen ist man rein von der Sache her schon auf einen Computer angewiesen. Dass die Lerninhalte vom gleichen Medium überliefert werden, ist da nahe liegend.

Das ProgrammierABC ist eine eLearning-Einheit zum Thema „Programmieren“. Es umfasst neben den wissensvermittelnden Textteilen auch eine Vielzahl von Programmieraufgaben, die nach dem aktuellen Kenntnisstand des Kursteilnehmers bzw. der Kursteilnehmerin geordnet sind, sowie das sogenannte „CodeLab“, wo die aktuellen Programme eingegeben und ausgeführt werden können.

Dieses Kapitel soll aufzeigen, wie die in den zwei vorhergehenden Kapiteln beschriebenen theoretischen Grundlagen im ProgrammierABC umgesetzt werden und Ideen für den praktischen Einsatz im Unterricht liefern.

3.2 Zielgruppe(n)

Das ProgrammierABC ist nicht dafür gedacht, dass alle Schüler es durcharbeiten müssen. Vielmehr soll es interessierten Schülerinnen und Schülern, zum Beispiel im Rahmen eines Informatikprojekts oder eines Wahl-/Freifaches, die Chance bieten, einen Einstieg ins Programmieren zu finden. Ich kann mir vorstellen, dass die eLearning-Einheit in einem Informatikkurs für Fortgeschrittene oder in einem berufsvorbereitenden Kurs im Berufswahljahr (9. oder 10. Schuljahr) seine Anwendung finden könnte. Natürlich steht auch der Nutzung durch andere Personen nichts im Weg.

3.3 Überlegungen rund ums ProgrammierABC

3.3.1 Verwendete Technologien

Für meine Bedürfnisse habe ich folgende Technologien in Betracht gezogen.

- **Geschlossenes lauffähiges Programm** nach dem Vorbild von Kara. Als Programmiersprache hätte ich dann Java verwendet. Vorteil dieses Ansatzes wären die quasi unbeschränkten technischen Möglichkeiten gewesen. Als Nachteil haben aber die mangelnde Flexibilität beim aktualisieren der Inhalte dieses doch recht grossen, monolithischen Systems, sowie allfällige Probleme beim Einsatz auf verschiedenen Plattformen gedroht. Ausserdem hätte der Aufwand für die Programmierung den Rahmen dieser Arbeit gesprengt.
- **Servergebundene Online-Plattform** mit dynamisch erstellten Webseiten. Die Vorteile hätten klar auf der Seite der technischen Möglichkeiten (Aufzeichnung der Lernpfade, Datenbankankbindung etc.) gelegen. Ausserdem wäre ein solches System weitgehend unabhängig von der Plattform. Als Nachteil hat sich auch hier der hohe Aufwand bei der Programmierung, sowie die Notwendigkeit einer ständigen Internetverbindung¹ abgezeichnet.
- **Netzunabhängiges Hypertextsystem** mit statischen HTML-Seiten. Diese Technologie hat die Vorteile, dass sie auch bei niedrigem Aufwand weitgehend plattformunabhängig ist; Alles was benötigt wird, ist ein aktueller Browser. Die Inhalte können schnell und unkompliziert angepasst werden. Ein solches System kann in einer späteren Phase jederzeit in eine servergebundene Online-Plattform umgearbeitet werden. Als Nachteil ist hier vor allem die komplizierte Umsetzung des CodeLab in und für JavaScript zu nennen.

Aufgrund dieser verschiedenen Überlegungen habe ich mich schliesslich entschieden, das ProgrammierABC als netzunabhängiges Hypertextsystem zu modellieren.

Als Lern-Programmiersprache kommt JavaScript² zum Einsatz. JavaScript kann in jedem moderneren Webbrowser ausgeführt werden. Es bietet via Browser einfache Dialoge zur Ein- und Ausgabe von Daten und ist an die Syntax der weit verbreiteten Programmiersprache C angelehnt. Zudem können JavaScripts mit kleinem Zusatzaufwand in Webseiten integriert werden. So können Schülerinnen und Schüler ihre Arbeiten leicht publizieren, was zusätzlich motivieren kann.

¹Heute verfügen zwar die meisten Computer in den Schulen über Internetzugang. Trotzdem kann ein vom Internet unabhängiges System vielseitiger eingesetzt werden. Zum Beispiel an einem Nebenarbeitsplatz, der nicht mit dem Netzwerk verbunden ist.

²JavaScript wird von Microsoft auch JScript genannt, offiziell heisst es ECMA-Script. Vgl. ECMA (1999)

Der Nachteil von JavaScript ist, dass der Interpreter im Browser aus Sicherheitsgründen in einer so genannten „Sandbox“³ läuft. Das Speichern und Lesen von Daten auf dem Festspeicher ist somit nicht möglich.

Als Entwicklungsumgebung habe ich das sogenannte „CodeLab“ in JavaScript und XHTML entwickelt. Im CodeLab können die Lernenden ihre Testprogramme eingeben und ausführen. Es färbt den eingegebenen Code ein (Syntaxhighlighting) und bietet eine Konsole zur Ein- und Ausgabe von Daten. Aus dem oben genannten Grund ist es leider nicht möglich die Scripts zu speichern. Eine Möglichkeit bestünde darin, Daten in einem Cookie zu speichern. Leider funktioniert das bei manchen Browsern nicht, wenn das Script über das „file://“-Protokoll lokal aufgerufen wird. Die erstellten Scripts müssen also in einem externen Textprogramm gespeichert werden.⁴

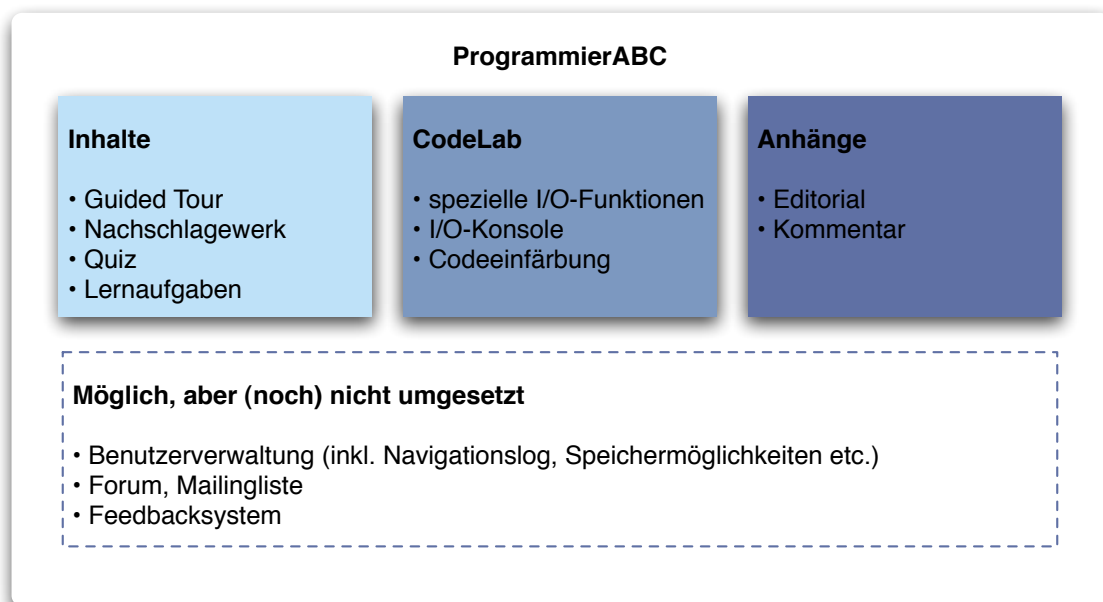


Abbildung 3.1: Die Module des ProgrammierABC

³Software, die in einer Sandbox läuft, wird vom Rest des Systems abgeschirmt, so dass sie keinen Schaden anrichten kann. Diese Technik ist ein gängiges Verfahren, um nicht vertrauenswürdigen Code sicher auszuführen.

⁴Zum Beispiel könnte das Lernjournal auch digital geführt werden und die Scripts aufnehmen.

3.3.2 Inhalte

Die Inhalte sind gemäss den Erkenntnissen aus Abschnitt 1.6.4 auf Seite 6 geordnet. Für die Lernenden stehen Navigationselemente zur Verfügung mit welchen sie einer „Guided Tour“ durch alle Lernblöcke folgen können. Alternativ ist es auch möglich – ausgehend von den Problemlöseaufgaben – die verfügbaren Inhalte als Nachschlagewerk zu verwenden. Zusätzlich zu den in Abschnitt 1.6.4 aufgeführten Lernblöcken, steht auch ein Kaptitel „Programmiertipps“ zu Verfügung. Die Lernblöcke „Arrays“ und „Objekte“ sind nicht enthalten. Diese Themen umfassen bereits fortgeschrittene Konzepte und würden den Rahmen des ProgrammierABC’s sprengen.

Jeder Lernblock umfasst sechs bis neun Seiten Einführung ins Thema, zwei bis vier Quizfragen zum erarbeiteten Text und schliesslich Aufgaben zur Vertiefung des Themas. Auf jeder Seite können über entsprechende Navigationselemente zusätzliche Informationen, kurze Aufgaben zur Vertiefung und – falls nötig – natürlich das CodeLab eingeblendet werden.

Die Inhalte sind zum Teil untereinander und mit einem Glossar verlinkt. Gegebenenfalls wird auch auf externe Quellen, wie zum Beispiel Wikipedia-Einträge, verwiesen, um das selbständige weiterforschen zu motivieren.

Layout

Bei der Entwicklung des Layouts habe ich mich an die Usability-Richtlinien aus Tabelle 2.1 gehalten. Im Gegensatz zu vielen anderen eLearning-Angeboten ist das Layout des ProgrammierABC (Siehe Abbildung 3.2) sehr schlicht und zurückhaltend. Es soll möglichst wenig von den Inhalten und dem Lernen ablenken. Das Layout gliedert sich in einen Navigationsteil (grau) und einen Inhaltsteil (weiss). Der Inhaltsteil scheint dank eines Schatteneffekts über dem Navigationsteil zu schweben. Die Form des Inhaltsteils erinnert an eine Kartei- oder Registerkarte, womit die Möglichkeit zu Blättern assoziiert werden soll. Über allem schwebt oben links das Logo. Die Schrift ist in einem sehr dunklen Blau linksbündig gesetzt.

Symbole

Es werden verschiedene Symbole verwendet. Ihre Bedeutung ist in Tabelle 3.1 beschrieben.



Abbildung 3.2: Layout des ProgrammierABC



Abbildung 3.3: Das CodeLab

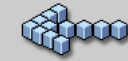
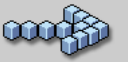

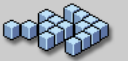
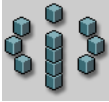

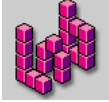



	gehe eine Seite zurück
	gehe eine Seite weiter
	gehe zur ersten Seite des vorhergehenden Kapitels
	gehe zur ersten Seite des folgenden Kapitels
	blende weitere Informationen zu dieser Seite ein
	blende Aufgaben zu dieser Seite ein
	öffne das CodeLab
	markiert einen wichtigen Hinweis
	zeigt an, dass ein Link zum Glossar führt.
	zeigt an, dass ein Link zu einer Seite im Programmier-ABC führt.

Tabelle 3.1: Symbole im ProgrammierABC

3.3.3 CodeLab

Das CodeLab ist in Programmierung und Layout unabhängig vom ProgrammierABC (obwohl es dafür entwickelt wurde). Es ist in zwei Bereiche gegliedert.

Links befindet sich der Code-Editor, der zwischen zwei Modi umgeschaltet werden kann. Im Eingabe-Modus handelt es sich um ein normales Textfeld mit schwarzem Text. Im eingefärbten Modus wird der Code nach bestimmten Regeln eingefärbt und kann jetzt nicht mehr geändert werden, bis wieder in den Eingabe-Modus gewechselt wird.

Rechts befindet sich die gelb hinterlegte Konsole. Hier werden Benutzereingaben und Programmausgaben Zeilenweise ausgegeben.

3.3.4 Umsetzung weiterer Ansprüche an eine eLearning-Einheit

Da es nicht möglich ist, aus dem ProgrammierABC heraus Daten zu speichern, ist es leider auch nicht möglich, Lernhilfen wie einen virtuellen Leuchtstift zu implementieren oder den Lernpfad zu speichern. Deshalb sind alternative Wege nötig geworden:

- Die Texte und die Aufgaben lassen sich als PDF laden und ausdrucken.
- Die Lernenden werden ermuntert ein digitales Lerntagebuch zu führen (Vorlagen für Word und OpenOffice sind verfügbar). Darin können der Lernpfad, Notizen und Resultate festgehalten werden.

Programme können per Copy&Paste in das Lerntagebuch eingetragen werden.

3.3.5 Anhänge

Rund um das ProgrammierABC gesellen sich noch weitere Dateien mit verschiedenen Zwecken. Einerseits sind da die editoriiellen Angaben zu Autor und Projekt (dazu gehört auch dieses Dokument). Andererseits steht auch eine wachsende Anzahl an Begleitmaterialien und Unterrichtsideen zur Verfügung.

Da diese Materialien sich immer wieder ändern und ständig wachsen, wird hier nicht näher darauf eingegangen.

4 Schlussgedanken

Das ProgrammierABC bietet zusammen mit dem CodeLab eine auf theoretischen Erkenntnissen basierende, praktische Umsetzung einer eLearning-Einheit zum Thema „Programmieren“. Allerdings muss es sich noch in der Praxis bewähren und gegebenenfalls mit den Erkenntnissen aus dem praktischen Einsatz verbessert und ergänzt werden.

Allenfalls ist auch eine Umarbeitung in eine dynamische Webapplikation (vgl. Kapitel 3.3.1) in Betracht zu ziehen. Die damit verbundenen Möglichkeiten würden dem ProgrammierABC zusätzliche Qualitäten verschaffen.

Literaturverzeichnis

Die Literaturangaben sind alphabetisch nach den Namen der Autoren sortiert. Bei mehreren Autoren wird nach dem ersten Autor sortiert. Es sind alle Texte aufgeführt, die ich im Rahmen dieser Arbeit beigezogen habe. Nicht alle werden auch zitiert.

RÜDEGER BAUMANN, ULRICH BACHMANN (1984): Abenteuer in BASIC – Ein Computer Comic. Stuttgart: Klett
ISBN 3-12-920412-1

MONIKA BUCHER, URS UTZINGER, URS AREGGER ET AL (2004): ICT an der Volksschule, Ergänzung zu den Lehrplänen. Luzern: BKZ
http://www.zebis.ch/inhalte/bildungsregion/lehrplaene/ict_volksschule_04.pdf

ECMA (1999): ECMAScript Language Specification. Genf: ECMA
<http://www.ecma-international.org/> (Stand: 29.05.2005)

ERICH HUI, URS BESTMANN (1986): Informatik für Gymnasien – Lern- und Arbeitsbuch mit Pascal-Programmierungskurs. Aarau: Sauerländer
ISBN 3-7941-2839-7

BRUSILOVSKY, P., CALABRESE, E., HVORECKY, ET AL (1997): Mini-languages: A Way to Learn Programming Principles. Education and Information Technologies 2 (1), S. 65-83.
<http://www2.sis.pitt.edu/~peterb/papers/minilang.html>

HANS PETER JAKOB, JÜRGEN MEIER, FREDY SCHALCHER (1988): Praktische Einführung in den Umgang mit Geräten und Programmen. Zürich: Verlag des Schweizerischen Kaufmännischen Verbandes
ISBN 3-286-31102-2

ANDREAS KNOBEL (2003): Tutorielles Coaching in virtuellen Lernszenarien. Universität Zürich: Institut für Informatik
http://www.ifi.unizh.ch/ifiadmin/staff/rofrei/DA/DA_2003.html (Stand: 03.06.2005)

- ECKART MODROW (1991): Zur Didaktik des Informatik-Unterrichts. Bonn: Ferd. Dümmlers Verlag
ISBN 3-427-46791-0
- THOMAS MÜLLER (2003): Den eigenen Lernweg gehen: Sinnvoller Einsatz von eLearning im Unterricht bei Jugendlichen mit besonderen Bedürfnissen. In Schweizerische Zeitschrift für Heilpädagogik 09/2003 Seite 11- 18
- THOMAS NÁROSY, VERENA RIEDLER (2002): E-Learning in der Schule.
<http://www.gym1.at/elearning/elearning-in-der-schule.pdf> (abgerufen am: 03.06.2005)
- SEYMOUR A. PAPERT (1996): The Connected Family: Bridging the Digital Generation Gap. Lonstreet Press
ISBN 1-563-52335-3
- ELEARNING-TEAM DER PÄDAGOGISCHEN HOCHSCHULE ZÜRICH (2005): Didaktik und Methodik für eLearning.
http://elearning.phzh.ch/ilias3/content/lm_presentation.php?ref_id=3285&obj_id=
(abgerufen am 04.12.2005)
- RAIMOND REICHEL, JÜRGEN NIEVERGELT, WERNER HARTMANN (2005): Programmieren mit Kara. Berlin: Springer
ISBN 3-540-23819-0
- STEFAN MÜNZ (2005): SELFHTML – HTML-Dateien selbst erstellen.
<http://de.selfhtml.org/> (Stand: 25.03.2005)
- WILFRIED SCHUPP (1980): Schüler programmieren in BASIC – Lehr- und Übungsbuch mit 150 Programmbeispielen und 260 Übungsaufgaben. Paderborn: Verlag Ferdinand Schöningh
ISBN 3-506-37449-4
- CLIFFORD STOLL (1999): High-Tech Heretic: Why Computers Don't Belong in the Classroom and Other Reflections by a Computer Contrarian. New York: Verlag Doubleday
- IAN STOODLEY, RUTH CHRISTIE, CHRISTINE BRUCE (2004): Master's Students' Experiences of Learning to Program: An Empirical Model. In Proceedings of QualIT2004
<http://sky.fit.qut.edu.au/bruce/pub/papers/QualIT2004-Bruce.pdf>