Grafische Programmiersprachen im Abitur

Kerstin Strecker[1]

Abstract: In diesem Artikel wird von den Erfahrungen aus dem Unterricht in einem Informatikkurs berichtet, der im Hinblick auf das Zentralabitur in Niedersachsen mit einer grafischen Programmiersprache (BYOB) vorbereitet worden ist. Daneben werden die Unterrichtserfahrungen aus einem Abiturkurs gesetzt, der nur Java als Werkzeug verwendet hat und die Erfahrungen aus einem „Bilingual"-Abiturkurs, dem BYOB und Java gleichberechtigt nebeneinander als Sprachen zur Verfügung standen.

Keywords: graphische Programmiersprachen, Sekundarstufe 2, Abitur

1 Einleitung

Grafische Programmierumgebungen finden immer mehr Verbreitung vor allem in den unteren Jahrgängen. Dies schlägt sich in Niedersachsen auch im Kerncurriculum für die Jahrgänge der Sek 1 nieder, wo es ausdrücklich heißt (S.21): „[...] entwickeln und implementieren einen Algorithmus in einer grafischen Programmiersprache […]" [KC14].

Die Gründe für die weite Verbreitung grafischer Sprachen wie Scratch [Sc15] in den unteren Jahrgängen führe ich auf die schnelle und leichte Erlernbarkeit durch intuitive Bedienung und die anschauliche Umsetzbarkeit anwendungs- und kontextbezogener Problemlösungen, z.B. durch die Integration grafischer Elemente, wie Kostümwechsel von Objekten u.a., zurück. Insgesamt führt dies nach meinen Erfahrungen zur schnellen Umsetzbarkeit eigener (kontextbezogener) Schülerideen und damit insgesamt zu einer breiten Motivation der SchülerInnen.
Dieser Artikel berichtet von dem Versuch, diese positiven Erfahrungen in der Oberstufe bis zum Abitur fortzusetzen.

Wenn die SchülerInnen in der Sek 1 mit Hilfe grafischer Programmiersprachen wie Scratch unterrichtet und vielleicht auch stärker motiviert wurden, wie geht es dann weiter? Reicht eine grafische Programmiersprache auch für das Bestehen des Abiturs aus oder bestehen die SchülerInnen das Abitur vielleicht sogar besser, wenn sie mit einer grafischen Sprache wie BYOB darauf vorbereitet werden? Um sich einer Antwort auf diese Frage zu nähern, wurde ein Informatikkurs mit BYOB auf das Zentralabitur vorbereitet. Die Erfahrungen werden im Folgenden beschrieben. Zum Vergleich werden Unterrichtserfahrungen in einem Java-Kurs und einem „Bilingual"-Kurs herangezogen.

[1] Institut für Informatik, Goldschmidtstr. 7, 37077 Göttingen, kerstin.strecker@gmx.de

Zu BYOB und dem Umgang mit BYOB verweise ich auf die Artikel von Eckart Modrow [Mo11a], [Mo11b], [Mo12] und [By15]. Beispiele für kontextbezogene Aufgaben mit Scratch, bzw. BYOB finden sich z.B. in [St14] und [St15].

2 Randbedingungen

In diesem Artikel werden drei Kurse miteinander verglichen, die im Mittel gleich viele Schüler hatten und dabei die Größe eines durchschnittlichen Oberstufenkurses. Dem Java-Kurs stand ausschließlich Java als Werkzeug zur Verfügung, der BYOB-Kurs wurde mit BYOB auf das Abitur vorbereitet und der „Bilingual"-Kurs hat sich von Anfang an gleichberechtigt mit den beiden Sprachen BYOB und Java beschäftigt. Die SchülerInnen dieses Kurses durften die Sprache jederzeit frei wählen, z.B. auch in der Klausur von Aufgabe zu Aufgabe unterschiedlich.
Da es in Niedersachsen noch kein Kerncurriculum für die Oberstufe gibt, orientiert sich der Unterricht an sogenannten Themenschwerpunkten, die jährlich etwas variieren und die Schwerpunkte des schriftlichen Zentralabiturs festlegen [NB15]. Neben Algorithmik finden sich dabei auch Themenfelder wie Technik, Datenbanken und formale Sprachen o.ä., die hier nicht betrachtet werden. Alle Klausuren inkl. Abitur werden handschriftlich geschrieben.
Die Ergebnisse der Abiture, Materialien und Aufzeichnungen über den gesamten Unterricht liegen vor und bestätigen die im Folgenden geschilderten Erfahrungen. Aufgrund der geringen Menge der SchülerInnen (insgesamt etwas über 50) werden die Ergebnisse in Kapitel 4 jedoch nur in Ansätzen statistisch ausgewertet.

3 Erfahrungsbericht

Im Java-Kurs wurden Aufgabenstellungen zur Implementierung von (eigenen) Algorithmen in Klausuren oft entweder (fast) korrekt bearbeitet oder erst gar nicht angefangen. Teillösungen fanden sich nur in ganz geringem Umfang.
Im BYOB-Kurs gab es bei ähnlichen Aufgabenstellungen in Klausuren genauso viele (fast) korrekte Lösungen, aber es konnten zusätzlich viel öfter Teilpunkte vergeben werden, weil mindestens Teillösungen der Implementierungsaufgabe vorhanden waren. Dabei unterschieden sich die algorithmischen Fähigkeiten der SchülerInnen nicht so auffallend, wie bei diesem Ergebnis zu vermuten wäre.
Aus der Beobachtung im Unterricht lassen sich dafür zwei Erklärungen dazu finden:

In jeder Lerngruppe gibt es SchülerInnen, die weniger aktiv am Unterricht teilnehmen und sich in Gruppen oder Partnerarbeit eher zurückziehen. Während sich solche SchülerInnen im Java-Kurs vermehrt in die Rolle des eher passiven Nachvollziehens der Lösungen ihrer Mitschülerinnen begeben haben, haben im BYOB-Kurs diese Schüler vermehrt aktiv an eigenen Lösungen gearbeitet. Durch die Möglichkeit, stets lauffähige

Produkte zu erzeugen, hat BYOB die SchülerInnen dazu motiviert, mit der Problemlösung überhaupt zu beginnen und so die Erfahrung vermittelt, dass eine Bearbeitung der Problemstellung nicht hoffnungslos ist. So konnten diese SchülerInnen im BYOB-Kurs später in Klausuren in viel stärkerem Maße auf eigene Erfahrungen zurückgreifen und diese Erfahrungen bei der Bearbeitung der neuen Aufgabe in der Klausur verwenden.

Eine weitere Erklärung für das Phänomen der Teillösungen sehe ich in dem Programmierverhalten in der Unterrichtszeit vor dem Abitur an sich. In Java programmieren die Schüler immer größere Blöcke auf einmal und testen diese dann, eventuell mit dem Debugger oder auch mit Testausgaben. Auch wenn die Blöcke nur wenige Zeilen umfassen, müssen sie zuvor vollständig durchdacht werden. In BYOB nutzen viele Schüler die Möglichkeit den Code stückweise zu entwickeln und dabei *gleichzeitig* zu testen. In dem laufenden System wird in BYOB der Zustand nach Ausführung eines Befehls oder Teilalgorithmus eingefroren, die Variablen z.B. behalten ihren Wert. Einzelne Befehle können so per Mausklick vor dem Einbauen in das Programm auf logische Korrektheit überprüft werden. Das Programm kann gleichzeitig Stück für Stück entwickelt und laufengelassen werden. Diese Arbeitsweise führt dann offenbar auch zu Teillösungen in Klausuren.

Zusammenfassend lassen sich meine Erfahrungen damit beschreiben, dass **die Verwendung von BYOB den Lernenden fast immer ermöglicht, wenigstens Teile der Lösung zu finden.**

Zwar hatten alle SchülerInnen Vorerfahrung im Programmieren, doch mussten einige Konzepte (Umgang mit Reihungen/Listen, eigene Methoden, OOP…) noch unterrichtet werden. Meine Aufzeichnungen zeigen hier eine Zeitersparnis im BYOB-Kurs im Vergleich zum Java-Kurs, weshalb neben anwendungsorientierten Aufgaben (s.u.) auch mehr Zeit für andere Themenbereiche (vgl. [NB15]) blieb, die dadurch etwas intensiver bearbeitet werden konnten und so vermutlich auch zu geringfügig besseren Ergebnissen im Abitur führten (vgl. Kapitel 4).

Ein oft gehörtes Argument von Lehrerseite gegen die Verwendung von BYOB ist die Schwierigkeit des Aufschreibens von BYOB-Programmen. Zwar malten die SchülerInnen in den Klausuren keine Blöcke, sondern notierten die Befehle im Klartext und markierten am Rand (der BYOB-Syntax nachempfunden) Beginn und Ende einer Schleife oder Auswahl. Allerdings wird dies durch das ausschließliche Hineinziehen der Befehle in den Arbeitsbereich mit der Maus während des Unterrichts nicht trainiert und muss also gesondert geübt werden. In Java neigen die SchülerInnen eher dazu, die Befehle selbst einzutippen, auch wenn einige Editoren bereits die Hilfe anbieten, mit einem Klick bestimmte Programmgerüste zu erzeugen. Hier fällt das Aufschreiben leichter. Auch fällt mir die Korrektur von Java-Programmen deutlich leichter als die handschriftlicher BYOB-Programme. Hier könnte natürlich Abhilfe geschaffen werden, indem das Abitur und Klausuren an Schulen, die über eine entsprechende Infrastruktur

verfügen, auch in Teilen am Rechner bearbeitet wird.

Nicht verschwiegen werden soll, dass in einem reinen BYOB-Abiturkurs aber auch einige SchülerInnen unzufrieden mit der Auswahl der Sprache sind. Die Argumente, warum sie ungern mit BYOB arbeiten, sind eher emotionaler Natur, z.B. „Java ist näher an der Maschine“ oder „näher an der echten Informatik“ (vgl. dazu auch [Mo11b]). Interessanterweise schätzten diese SchülerInnen ein Java-Programm hoch ein, das zwar ein beeindruckendes Ergebnis lieferte, intern jedoch fast ausnahmslos aus dem Aufruf von Methoden verschiedener Java-Bibliotheken bestand und algorithmisch wenig anspruchsvoll war.
Die Akzeptanz von BYOB im Bilingual-Kurs war dagegen überhaupt kein Thema, die obigen Aussagen kamen von diesen SchülerInnen nicht.
Eine Erklärungsmöglichkeit ist folgende: Im Bilingual-Kurs wurde von Anfang an stets jedes Konzept in beiden Sprachen nebeneinander demonstriert und so immer wieder indirekt darauf hingewiesen, dass es einen Unterschied gibt zwischen dem Konzept an sich und der Umsetzung in einer beliebigen Sprache. Da BYOB und Java an vielen Stellen eine unterschiedliche Syntax besitzen, aber nebeneinander verwendet wurden, wurde deutlich, dass sprachspezifische Details oder die „Schreibweise“ eine andere Kategorie hatten als das algorithmische Problemlösen oder die in algorithmischen Grundstrukturen an sich. Dies konnte an den Nachfragen und Aussagen der Schülerinnen festgemacht werden, die anders als in anderen Kursen Fragen stellten wie: „wie mache ich Vererbung noch mal in Java?“

Meine Erfahrung lässt sich folgendermaßen zusammenfassen: **Im Unterricht ist im Großen und Ganzen ein Zusammenhang festzustellen zwischen der Akzeptanz von BYOB und der Trennung zwischen Konzept und Sprache.**

Durch die Trennung von Konzept und Umsetzung zeigte sich, dass der Großteil der SchülerInnen des Bilingual-Kurses reflektiert die Sprache nach Art der Aufgabe gewählt und bei Bedarf gewechselt hat. Während BYOB in Aufgaben, in denen es um mathematische Rechnungen ging, sehr umständlich ist und hier von mehreren SchülerInnen Java gewählt wurde, wählten die SchülerInnen z.B. zur Erzeugung der Kochkurve alle BYOB. Dies Verhalten zeigte sich übrigens auch in den Klausuren des Bilingual-Kurses, in denen teilweise aufgabenweise die Sprache gewechselt wurde.
Die Fähigkeit zur Unterscheidung zwischen Konzept und sprachlicher Umsetzung habe ich vor dem Unterricht im Bilingual-Kurs nie so intensiv wahrgenommen, und ich werte diese Kompetenz sehr hoch, auch für den weiteren Lebensweg der SchülerInnen, da sich Sprachen schnell verändern, Konzepte seltener.
Die Kompetenz, zwischen Konzept und Umsetzung zu unterscheiden und aufgabenspezifisch die geeignetere Sprache zu wählen, ist meiner Meinung nach mit zwei textbasierten Sprachen in dieser Form nicht so einfach zu erreichen, da allein das Erlernen der Sprachen zu viel Zeit in Anspruch nehmen und die SchülerInnen wahrscheinlich mit der Syntax durcheinander kommen würden.

Die rückblickende Durchsicht der Kurshefte zeigte für mich überraschend, dass meine Aufgabenstellungen im BYOB-Kurs und Java-Kurs unterschiedlich waren. Beispielsweise sehen die thematischen Schwerpunkte [NB15] die Implementation eines Stapels vor. Während ich im Java-Kurs den Anwendungsbezug eher vernachlässigt habe, weil die Implementation eines dynamischen Stapels sehr viel Zeit in Anspruch genommen hat, habe ich im BYOB-Kurs mit Hilfe der dynamischen Liste den Stapel nur nebenbei als Bestandteil des Lindenmayersystems eingeführt. Durch die grafischen Elemente (hier die Zeichenbefehle) konnte dies schnell umgesetzt werden und die Zeit, die vorher für die Java-Implementation erforderlich war, wurde im BYOB-Kurs für die Erzeugung künstlicher Pflanzen genutzt.

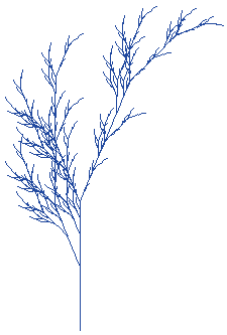

Abb. 1: Erzeugung künstlicher Pflanzen

Während im Java-Kurs viele klassische Algorithmikaufgaben behandelt worden sind, habe ich in BYOB durch die integrierten grafischen Elemente (Kostüme, Kostümwechsel, und das Hinterlassen von Abdrücken in diesem Kostüm…) eher Anwendungs- und kontextbezogene Aufgaben gewählt, beispielsweise die Generierung eines EAN-Strichcodes mit Prüfsumme im Themengebiet „Codierung und fehlererkennende Codes“ in BYOB statt dem Anhängen eines Paritäts“bits“ an eine Zeichenkette aus 0en und 1en in Java.
Dieselbe Erfahrung zeigte sich auch im Wahlverhalten der SchülerInnen des Bilingual-Kurses. Trotz ausgewogener Anteile bei der Wahl von Java oder BYOB im Unterricht zuvor, haben in einer Phase, wo eigene Projekte umgesetzt werden konnten, die meisten SchülerInnen ihre Ideen in BYOB umgesetzt, sei es die Programmierung von Geschicklichkeitsspielen oder eine physikalische Simulation, weil insgesamt die Visualisierung hoch gewertet wurde.

Meine Erfarungen zeigen also, dass sich **anwendungsorientierte Aufgaben mit BYOB oft leichter umsetzen lassen.**

Als Konsequenz müssten sich dann auch Klausuraufgaben verändern, da sich „alte“ Aufgaben schwer wiederverwenden lassen: Ich hatte für alle Kurse dieselbe Aufgabe ausgewählt, in der eine leicht schräge Kugelrinne beschrieben wurde, in die zufällig Kugeln verschiedener Farben geworfen werden. Liegen drei Kugeln derselben Farbe nebeneinander, darf der Werfer sie entfernen und bekommt einen Punkt. Die Kugelrinne wurde mit einer Reihung von Zeichenketten (als Farben) beschrieben. Es galt, alle vorhandenen Dreiergruppen zu entfernen.
Während sich alle SchülerInnen im Java-Kurs darauf konzentrierten, dass eine Reihung durchlaufen werden musste und Zeichenketten auf eine bestimmte Eigenschaft hin untersucht werden mussten, gab es im BYOB-Kurs auch SchülerInnen, die damit starteten, die Kugelrinne und den Wurf einer Kugel zu simulieren bzw. zu implementieren, um diese zunächst zu füllen.

Das oben beschriebene Beispiel der Lindenmayersysteme zeigt auch, wie im BYOB-Kurs mit Objekten umgegangen worden ist.
Zunächst bestand das Programm zur Erzeugung künstlicher Pflanzen aus mehreren nacheinander ausgeführten eigenen Methoden zur iterativen Berechnung der Ableitungen und dem Zeichnen der Pflanzen. Die Funktionalität des Stapels wurde mit einer Liste umgesetzt, auf der jeweils wieder als Liste x-Koordinate, y-Koordinate und Richtung abgelegt wurden. Im zweiten Schritt wurde ein Stapelobjekt erzeugt u.a. mit den klassischen Methoden pop und push und einer internen Liste. Dieser Stapel wurde nun in das „Hauptprogramm“ eingebunden und sogar in späteren Projekten wiederverwendet. Anschließend wurde ein weiteres Objekt „Koordinate“ erstellt, das eigene Variablen für x-Koordinate, y-Koordinate und Richtung besaß. Dieses Objekt, der Prototyp für alle vorkommenden Koordinaten, konnte unabhängig von den anderen Objekten getestet werden. Da je nach Schachtelungstiefe vor Ausführung des Programms unbekannt ist, wie viele Koordinaten man zur Erzeugung künstlicher Pflanzen braucht, wurde dem Koordinatenobjekt eine weitere Methode hinzugefügt, die einen Klon dieses Objekts zurückgegeben hat. Auch dies wurde ins Hauptprogramm eingebunden, es wurden nur noch Koordinaten auf den Stapel gelegt und man konnte bei der Ausführung mitverfolgen, wie die Koordinaten dynamisch erzeugt wurden und ihre Individualität im Wortsinn „sichtbar“ wurde.
Hier konnte also der Weg vom konkreten Objekt zur Verallgemeinerung gegangen werden. In Java ist nur eine Generierung eigener Klassen möglich. Egal ob später nur ein Objekt erzeugt wird oder eine unbekannte Zahl von Objekten erzeugt werden, man muss den Weg über eine Klasse gehen, bevor Objekte erzeugt werden können. Didaktisch gesehen gefällt mir die Umsetzung des OOP-Konzepts in BYOB auf diese Weise etwas besser. Es kann in den Unterrichtsgang gut integriert werden, zumal man alle (dynamisch erzeugten) Objekte nach dem Programmablauf einzeln untersuchen kann.
Im Vergleich zum Klassenkonzept von Java können so auch wieder die Konzepte der OOP und die sprachspezifische Umsetzung getrennt und auch hier eine Vermischung vermieden werden.

Bei mir selbst habe ich außerdem beobachtet, dass ich durch die Möglichkeit des schnellen Zusammenschiebens von Teilalgorithmen Konzepte oft mit einer kurzen BYOB-Sequenz veranschaulicht, ausgeführt und damit geeigneter visualisiert, in einem reinen Java-Kurs hingegen in solchen Fällen oft nur Teilcode an die Tafel geschrieben habe.

4 Zusammenfassung und Auswertung meiner Erfahrungen

Grafische Programmiersprachen sind (meiner Meinung nach) eine echte Bereicherung für die Schule, auch in der Oberstufe. SchülerInnen, die mit BYOB auf das niedersächsische Zentralabitur vorbereitet wurden, haben dies im Mittel eher besser bestanden als reine Java-Kurse. Auch wenn kaum statistische Aussagen möglich sind, lässt sich sagen, dass sie den Aufgaben gewachsen waren und ihnen keine Konzepte oder Kompetenzen fehlten. Für weniger leistungsstarke SchülerInnen ist BYOB ein besonderer Gewinn, weil Teillösungen einfacher zu realisieren scheinen.
Im Detail konnten im Mittel im Java-Kurs 58% der Punkte im Teilbereich Algorithmik und 82% in allen anderen Teilbereichen des Abiturs erreicht werden.
Im BYOB-Kurs konnten im Mittel 77% der Punkte im Teilbereich Algorithmik des Abiturs erreicht werden und 85% der Punkte in allen anderen Teilbereichen. Für die Vorbereitung der anderen Teilbereiche stand im BYOB-Kurs (vgl. Kapitel 3) etwas mehr Zeit zur Verfügung. Es ergibt sich ein Vergleich wie folgt:

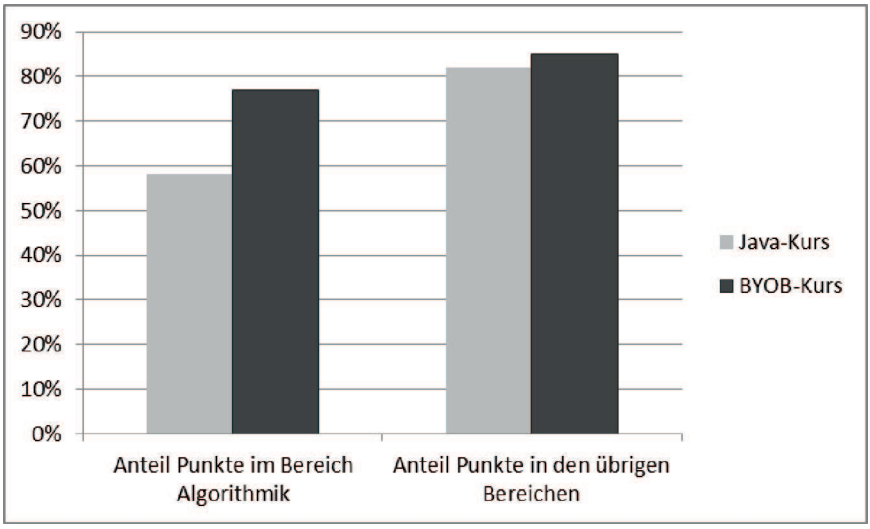

Abb. 2: Abiturergebnisse Java- und BYOB-Kurs

Die grafischen Elemente von BYOB verlieren bei der Umsetzung anwendungsorientierter Aufgaben auch in der Oberstufe nicht ihren Reiz. Weiterhin scheinen anwendungsbezogene Aufgaben auch für OberstufenschülerInnen motivierend zu sein, in eigenen Projekten wählen sie selbst anwendungsbezogene Beispiele und die scheinen sich besser mit BYOB umsetzen zu lassen.

Auch wenn Unterrichtende an textbasierten Sprachen festhalten möchten, ist der zusätzliche Einsatz von BYOB ein Gewinn für die SchülerInnen. Aus meinen Aufzeichnungen geht hervor, dass es kaum mehr Zeit erfordert, neben z.B. Java die Konzepte auch noch in BYOB darzustellen. Man gewinnt aber die Möglichkeit, den SchülerInnen den Unterschied zwischen Konzept und sprachspezifischer Umsetzung zu verdeutlichen. Die Kompetenz, sich aufgabenbezogen reflektiert für die in diesem Fall geeignetere Sprache zu entscheiden, kann kaum anders vermittelt werden.

Literaturverzeichnis

[By15] BYOB, URL: http://snap.berkeley.edu/old-byob.html, Zugriff: 26.1.2015.

[KC14] Niedersächsisches Kultusministerium: „Kerncurriculum für die Schulformen des Sekundarbereichs I. Schuljahrgänge 5-10. Informatik“, http://db2.nibis.de/1db/cuvo/datei/kc_informatik_sek_i.pdf, Zugriff: 26.1.2015.

[Mo11a] Modrow, Eckart; Mönig, Jens; Strecker, Kerstin: „Wozu Java? Plädoyer für grafisches Programmieren“, Zeitschrift LOG IN, Heft 168, LOG IN-Verlag, 2011.

[Mo11b] Modrow, Eckart: „Visuelle Programmierung – oder: Was lernt man aus Syntaxfehlern?“, in Marco Thomas (Hrsg.): „Informatik in Bildung und Beruf“, INFOS 2011, Lecture Notes in Informatics - Proceedings, Gesellschaft für Informatik, 2011.

[Mo12] Modrow, Eckart: „Objektorientiertes Programmieren mit BYOB“, Zeitschrift LOG IN, Heft Nr. 171, LOG IN-Verlag 2011/2012.

[NB15] Zentralabitur Niedersachsen, URL: http://www.nibis.de/nibis.php?menid=1395, Zugriff: 30.4.2015.

[Sc15] Scratch, URL: http://scratch.mit.edu/, Zugriff: 26.1.2015.

[St14] Strecker, Kerstin: „kontextbezogene Aufgaben“, Zeitschrift LOG IN, Heft Nr. 176/177, LOG IN-Verlag 2014.

[St15] Strecker, Kerstin: „Informatik konkret: 28 Anwendungsbeispiele. Lehrplanthemen erarbeiten – in Scratch programmieren – auf medizinische Kontexte anwenden“, AOL-Verlag, 2015.