Scratch2Greenfoot – Eine kreative Einführung in die Programmierung mit Scratch und Greenfoot

Ralf Romeike

Institut für Mathematik und Informatik
Pädagogische Hochschule Schwäbisch Gmünd
Oberbettringer Str. 200
73525 Schwäbisch Gmünd
romeike@cs.uni-potsdam.de

Abstract: Eine Einführung in die Programmierung für Lehramtsstudierende muss auch besondere Ziele berücksichtigen: Sie sollte eine positive Einstellung zum Programmieren vermitteln und Ideen für den eigenen Unterricht liefern. Scratch2Greenfoot ermöglicht eine sanfte Progression von Scratch zu Greenfoot, während die Studierenden frühe Erfolge erzielen und Programmieren als kreatives Gestalten wahrnehmen. Eine erste Erprobung mit Studierenden für das Haupt- und Realschullehramt verlief erfolgreich.

1 Einleitung

Die Vermittlung grundlegender Progammierkenntnisse hat sich in der Informatikdidaktik als eine große Herausforderung erwiesen. Falsche Vorstellungen, Ängste und mangelnde Vorkenntnisse stellen regelmäßig Probleme dar, die nicht nur im „klassischen" Informatikstudium, sondern oft noch gravierender in der Lehrerbildung auftreten: Lehrveranstaltungen zur Programmierung zählen zu den schwierigsten und weisen hohe Durchfallquoten auf. Damit können sie dafür mitverantwortlich gemacht werden, dass mancherorts über 50% der Lehramtsstudierenden ihr Informatikstudium innerhalb der ersten zwei Studienjahre wieder abbrechen (vgl. [RS06]).

Für angehende Informatiklehrer ist dieser, oft erste, Eindruck vom Programmieren besonders problematisch, sollen sie doch später selbst mit möglichst positiver Einstellung das Thema unterrichten. Zusätzlich sind Studienabbrüche hinsichtlich der ohnehin geringen Studierendenzahlen für das Lehramt Informatik grundsätzlich problematisch. Die häufig im Rahmen des „klassischen" Informatikstudiums angebotenen und von Lehramtsstudierenden mitbelegten Lehrveranstaltungen führen überdies dazu, dass sich Studierende mit unterschiedlichen Zielen und damit auch Bedürfnissen in einer Lehrveranstaltung befinden. Insbesondere den Zielen und Bedürfnissen der Lehramtsstudierenden wird hierbei regelmäßig nicht Rechnung getragen. An der hiesigen Hochschule hatten wir im WS 2009/10 die Möglichkeit, eine eigenständige Lehrveranstaltung zur Programmierung für Studierende des Haupt- und Realschullehramts zu konzipieren und anzubieten. Die zugrunde liegenden Überlegungen, Materialien und Erfahrungen werden im Folgenden dargestellt. In Kapitel 2 werden die Hintergründe und Bedürfnisse der Studierenden analysiert und damit die inhaltsübergreifenden Ziele der Lehrveranstaltung

präzisiert. In Kapitel 3 werden bekannte Forschungsergebnisse zur Problematik diskutiert. Kapitel 4 stellt gängige Ansätze zur Einführung in die Programmierung in Schule und Studium gegenüber, aus welchen ein Ansatz entwickelt wird, wie mit Hilfe eines kreativen Zugangs und der Erstellung von Animationen und Spielen die Studierenden einen motivierten kreativen Einstieg in die Programmierung erfahren (Kapitel 5).

2 Ziele und Probleme der Programmierausbildung im LA-Studium

Ob Programmieren in der Schule unterrichtet werden sollte, ist Bestandteil teilweise kontroverser Diskussionen (vgl. z. B. [GH10]). Es kann allerdings festgestellt werden, dass eine Umsetzung der GI-Empfehlungen, vor allem im Bereich Algorithmen, ohne Programmieren nicht sinnvoll möglich ist. Die Umstände und Ziele dabei haben sich in den letzten Jahren deutlich geändert. Kinder wachsen heute in einer von Computern geprägten Welt auf und werden deshalb auch als „Digital-Natives“ bezeichnet. Entsprechend sollte der Computer vermittelt werden als Medium des persönlichen Ausdrucks und als Werkzeug, das nicht nur bei der Arbeit hilft, sondern auch zum Spaß und für Hobbys genutzt wird (vgl. [Gu02]). Dennoch wird der Computer von den meisten Schülerinnen und Schülern überwiegend nur rezeptiv verwendet: zum chatten, mailen, spielen und kommunizieren in sozialen Netzwerken. Die wenigsten erstellten aber selbst Spiele und Animationen mit dem Werkzeug Computer. Es ist quasi, als ob sie lesen, aber nicht schreiben können (vgl. [Re07]). Aufgabe eines modernen Informatikunterrichts sollte es damit sein, die Schülerinnen und Schüler zu einem gestalterischen Umgang mit Informatiksystemen zu befähigen. Diese geänderte Sichtweise muss sich auch in der Lehrerbildung niederschlagen, regelmäßig greifen Lehrerinnen und Lehrer auf Mittel und Methoden zurück, wie sie auch selbst einen Inhalt erfahren haben. Diese Entwicklungen implizieren auch geänderte Ziele und Anforderungen für die Schülerinnen und Schüler: die Gestaltung von Medien und Spielen zu eigenem Nutzen statt des Entwurfs und Implementierens fremdbestimmter Probleme und Softwaresysteme.

In der Praxis stellt sich das geringe Stundendeputat des Fachstudiums als problematisch dar. So belegen Studierende des Lehramts für die Sekundarstufe I mitunter nur 6-8 SWS im fachwissenschaftlichen Bereich. Komplexe Programmiererfahrungen sind hier gar nicht möglich. Die GI-Empfehlungen selbst beschränken sich durchaus auf recht einfache und grundlegende Kompetenzen, die durch die Lehrerinnen und Lehrer zu vermitteln sind und auch tatsächlich bleiben die in der Schule vermittelten Programmierkompetenzen relativ elementar. Offensichtlich ist bei der Wahl eines Werkzeugs/einer Programmiersprache zur Einführung in die Programmierung in der Sekundarstufe I nicht nur die mögliche Komplexität, sondern auch das Potential, wie weitere Inhalts- und Prozesskompetenzen angesprochen werden können, zu berücksichtigen: Bspw. Begründen und Bewerten, Strukturieren und Vernetzen, Kommunizieren und Kooperieren. Entsprechend der genannten Probleme und Anforderungen sollte eine Lehrveranstaltung zur Programmierung neben inhaltlichen Schwerpunkten folgende inhaltsübergreifende Ziele verfolgen:

1. **Grundlegende und vielfältige Programmiererfahrungen ermöglichen**
   Die Studierenden sollten möglichst viele eigene Mini-Projekte im Semester verwirk-

lichen und ein größeres eigenständiges Programmierprojekt umsetzen.

2. **Ideen vermitteln, wie Programmierung in der Schule eingeführt werden kann**
   Studierende tendieren dazu, Inhalte in der Schule so zu vermitteln, wie sie diese selbst erfahren/erlernt haben. Entsprechend ist es wichtig, Methoden und Beispiele aufzuzeigen, die so auch in der Schule verwendet werden können.
3. **Eine positive Einstellung zum Programmieren herstellen und Berührungsängste abbauen/verhindern**
   Fast alle Studierenden äußerten zu Beginn Vorbehalte zur Lehrveranstaltung: Zu hohes Tempo, Nichtverstehen u. ä. wurden befürchtet. Solche negative Einstellung wird dann problematisch, wenn selbst Programmierung unterrichtet werden soll.

Im Kapitel 3 werden typische Probleme der Lehrveranstaltungen zur Programmierung analysiert und anschließend ein Weg vorgestellt, wie versucht wurde, o. g. Ziele zu erreichen.

3 Zur Einführung in die Programmierung an Hochschulen

Probleme und Erfolge beim Lehren und Lernen der Programmierung auf Hochschulniveau gehören zu den gut untersuchten Forschungsgebieten der Informatikdidaktik. Traditionelle Lehrmethoden führen regelmäßig dazu, dass Studierende nur ein oberflächliches Programmierverständnis erreichen und Programmieraufgaben Zeile um Zeile lösen, statt Programmstrukturen im größeren Kontext zu berücksichtigen [SS89]. Zwar besitzen Studierende Wissen hinsichtlich Syntax und Semantik einzelner Befehle, allerdings fehlt Wissen und Erfahrung, sie zu Programmen zusammenzusetzen (vgl. [Wi96]). Entsprechend sollte Konzeptwissen und dessen Anwendung frühzeitig im Lernprozess berücksichtigt werden. Eine weitere Hürde stellt die Syntax der verwendeten Sprache dar, welche bei Programmieranfängern regelmäßig zu großen und frustrierenden Problemen führt (vgl. [KP05]). Durch eine komplexe Syntax verschiebt sich der Fokus: Anfänger bemühen sich mitunter derart darum, die Syntax richtig einzuhalten, dass der zugrunde liegende Algorithmus nachrangig wird. Bei z. B. [MPS06] führte eine einfachere Syntax auch zu weniger logischen Fehlern. Befürchtungen und Ängste hinsichtlich des Programmierens wirken sich unweigerlich auf die Motivation der Studierenden aus. Wird es den Studierenden ermöglicht, eigene Probleme auszuwählen und die Ideen kreativ umzusetzen, können hierdurch Motivation und Interesse gesteigert werden (z.B. [MST97]). Insbesondere für zukünftige Informatiklehrer scheint es angebracht zu sein, Programmieren als Mittel persönlichen kreativen Gestaltens einzuführen und zu motivieren. Nicht zu letzt ist bekannt, dass Frauen ihre Informatikausbildung häufig abbrechen, weil sie in Informatikkursen nicht genug Raum für individuelle Kreativität finden [Gu02]. Die dokumentierten Lernschwierigkeiten schlagen sich in Ängsten und Befürchtungen der Studierenden wieder. Studienanfänger befürchten, dass das Informatikstudium zu schwierig für sie sein könnte und sie möglicherweise falsche Vorstellungen besitzen könnten [RS06].

Vor diesem Hintergrund befragten wir die Studierenden des Kurses Objektorientierte Modellierung und Programmierung nach ihren Vorstellungen, Wünschen und Befürchtungen hinsichtlich der Lehrveranstaltung. Diese spiegeln die informell transferierten Erfahrungen anderer Studierenden und damit einhergehende Befürchtungen wieder,

äußern aber auch daraus resultierende Hoffnungen, wie der eigene Lernprozess entsprechend besser verlaufen könnte. Befürchtet wird, dass viele Sprachen nur oberflächlich angesprochen werden, dass zu viele Vorkenntnisse vorausgesetzt werden, dass es im Kurs zu schnell geht und dass die Studierenden mit dem Verständnis nicht mitkommen. Die Studierenden wünschen sich durchaus, einen hinreichenden Einblick in die Programmierung zu erlangen, ja gar „als Programmierer den Kurs zu verlassen“. Konkret wünschen sie sich mehr Praxis als Theorie und anwendungsorientiertes Lernen anhand von Projekten und dass sie anschließend Programme für den privaten Gebrauch und die Schule erstellen können. Um das Ziel zu erreichen, dass die Studierenden einen konzeptorientierten Einstieg in die Programmierung erfahren, der sie bei ihren geringen Vorerfahrungen abholt, früh Erfolgserlebnisse verschafft, Kreativität im Programmieren deutlich macht und zugleich analog in der Schule einsetzbar ist, geschieht der Einstieg in die Programmierung mit Scratch analog einem erprobten und evaluierten Unterrichtsbeispiel [Ro07]. Für die weiterführende Lehre bleiben nun verschiedene Möglichkeiten, den Übergang zu einer komplexeren Programmiersprache zu gestalten.

4 Ansätze zu Einstieg und Fortführung der Programmierung

Lehrveranstaltungen zur Programmierung lassen sich losgelöst von einer konkreten Programmiersprache (z. B. mit Hilfe von Pseudosprachen), anhand einer konkreten Programmiersprache oder mithilfe verschiedener, oft zahlreicher, Programmiersprachen durchführen – alle Varianten sind an Schulen und Hochschulen zu finden. Die Verwendung eine Pseudosprache eignet sich für die Schulpraxis nicht, verhindert sie doch, Lerninhalte zu demonstrieren, Ideen umzusetzen und schnelle Erfolge zu erzielen. Damit ist auch die Verwendung im Lehramtsstudium fraglich. Insbesondere an Schulen wird häufig versucht, innerhalb einer Sprachfamilie zu bleiben: Miniwelten eignen sich zum motivierten Einstieg in die Programmierung, der Umstieg in eine „richtige“ Programmiersprache bleibt allerdings problematisch. Ansätze zur Lösung des Problems bieten Miniwelten, die eine (ähnliche) Syntax der später intendierten Programmiersprache verwenden: z.B. Java-Hamster, Java-Kara oder Ruby-Kara. Problematisch bleibt bei diesen Beispielen der Umstand, dass die Sprachen selbst beim Erlernen aufgrund der Syntax und Komplexität Probleme bereiten und ein Verständnis der zugrundeliegenden Konzepte nur unzureichend unterstützt wird. Der Umstieg in die nächst höhere Programmiersprache erfolgt in einem großen Schritt: Die potentiellen Möglichkeiten der Programmiersprache vervielfachen sich, die tatsächlichen Möglichkeiten werden allerdings aufgrund des unzureichenden Vorwissens stark eingeschränkt: Die Möglichkeit wie bisher in Miniwelten Figuren zu animieren entfällt, neue thematisierte Projekte entstammen mitunter konträren und v. a. abstrakten Themen wie Berechnungen oder Sortieren.

Die Verwendung visueller Programmiersprachen ist ein gängiger Ansatz, die Einführung in die Programmierung zu erleichtern. Vorteile ergeben sich vor allem aus der didaktisch reduzierten Entwicklungsumgebung, dem Eliminieren der Möglichkeit, Syntaxfehler zu machen und motivierenden Aufgabenstellungen (vgl. [Ko09]). Ist ein Verständnis für die Programmierkonzepte erreicht worden, muss nur noch eine neue Syntax sowie der Umgang mit dieser erlernt werden. Die Gestaltung eines solchen Übergangs kann sich allerdings als schwierig gestalten. Beispielsweise sind für die weiterführende Lehre nach

Scratch noch keine Unterrichtskonzepte bzw. -erfahrungen dokumentiert. Greenfoot eignet sich für die weiterführende Lehre nach einem Einstieg mit Scratch: Die Art und Weise der Softwareentwicklung bleiben ähnlich (Animationen, Spiele und Simulationen), Greenfoot erweitert allerdings durch die Verwendung von Java die Möglichkeiten um ein Vielfaches. Auch hier bleibt aber das Problem, die hinzugekommenen Möglichkeiten so zu verpacken, dass der Schritt des Umstiegs überschaubar und der Lernende nicht „erschlagen" wird. Ein Versuch, diesem Problem gerecht zu werden, stellt Scratch2Greenfoot dar.

5. Scratch2Greenfoot

Scratch2Greenfoot[1] besteht aus einem Template für Greenfoot, in welchem viele wesentliche Programmier-Bausteine von Scratch in Greenfoot nachmodelliert sind (z.B. turn x, go to x y), verschiedenen Beispielen und einer Dokumentation. Hierdurch beschränkt sich die Hürde des Umstiegs auf das Zurechtfinden in der neuen Entwicklungsumgebung und der Verwendung von Java als Programmiersprache, welche die Einhaltung der Syntax nun konsequent erforderlich macht. Eine wesentliche Motivation zur Verwendung von Greenfoot und damit zum Umstieg auf Java wird durch die Illustration dynamischer Objekterzeugung anhand des Projekts „Springende Bälle" verdeutlicht: War es in Scratch noch mühsam, neue Bälle durch Kopieren einzeln hinzuzufügen, lassen sich in Greenfoot relativ einfach zahlreiche Instanzen der Klasse Ball erzeugen. Entsprechend erfolgt der Umstieg am Beispielprojekt Pong. Die Studierenden haben hierzu in Scratch ein einfaches Pong-Spiel implementiert und übertragen nun diese Lösung nahezu eins zu eins nach Greenfoot. Die in Scratch2Greenfoot nachmodellierten Scratch-Bausteine lassen sich analog der Programmierung in Scratch verwenden, verdeutlichen aber gleichzeitig ihre Verwendung in Greenfoot und damit ein Verständnis für die Programmiersprache Java. Vorteilhaft bei dieser Vorgehensweise ist, dass sich die Studierenden vordergründig auf die Verwendung der Konzepte beschränken können unter Beibehaltung und Erweiterung ihres bis dahin erworbenen Programmierverständnisses, Brüche werden vermieden.

In der weiteren Entwicklung eigener Projekte können die Studierenden auf ihrem Wissen um Scratch aufbauen und Projekte analog implementieren. Die Dokumentation von Scratch2Greenfoot verdeutlicht dabei, wie die Java-Syntax im Vergleich zu den Scratch-Bausteinen anzuwenden und wie nicht analog implementierte Konzepte zu handhaben sind. Bei einer geschickten Steuerung des weiteren Unterrichtsverlaufs werden alle behandelten Konzepte spiralcurricular wiederholt im Greenfootkontext thematisiert und neue Konzepte bei der Erstellung komplexerer Projekte sukzessive eingeführt. Greenfoot motiviert dabei zusätzlich den Modellierungscharakter der Programmierung und die damit verbundenen Konzepte wie Vererbung, Klasse vs. Objekt, Attribute und Methoden. Die Gewissheit, in Scratch bereits Erfolge erzielt zu haben und mit Greenfoot eine leistungsfähigere Sprache zu benutzen, wirkt zusätzlich motivierend; ein „Verlorensein" wird vermieden, indem immer wieder der Bezug zu vorherigen Erfahrungen hergestellt

[1] Sprich: Scratch to Greenfoot (Von Scratch zu Greenfoot). http://greenroom.greenfoot.org/resources/8

wird. Als möglicherweise problematisch könnten sich die unterschiedlichen „Arbeitsweisen“ von Scratch und Greenfoot darstellen – während Scratch aktionsgesteuert ist, müssen in Greenfoot alle Aktionen über eine „Act-Methode“ als Controller gesteuert werden. In der Praxis hat sich dies allerdings als erstaunlich unproblematisch erwiesen.

6 Erprobung und Fazit

Eine erste Erprobung verlief erfolgreich, alle Studierenden machten frühzeitig positive Erfahrungen mit Scratch und konnten die konstruktionistische und kreative Sichtweise auf die Programmierung mit Greenfoot übertragen. Auch wenn die nun konsequent erforderliche Berücksichtigung der Syntax und die damit verbundene Notwendigkeit des Debuggens die Studierenden extra Aufmerksamkeit und Nerven kostete, war die Aussicht auf ein gutes funktionierendes Programm motivierend. Das Ergebnis des insgesamt nur 12 zweistündige Lehrveranstaltungen umfassenden Kurses war sehr zufriedenstellend: Alle Studierenden entwickelten mehrere Programme und als Abschlussprojekt ein umfangreiches, selbstentwickeltes Spiel. Damit starten sie mit einem positiven Bild von der Programmierung, guten Erfahrungen und Ideen, wie sie selbst Programmierung im Unterricht umsetzen können, in den Lehrerberuf. Eine ausführliche Evaluation und Weiterentwicklung des Konzepts sind in Vorbereitung.

7 Literatur

[GH10] Gutknecht, J.; Hromkovic, J.: Plädoyer für den Programmierunterricht. In Informatik-Spektrum 33(2), 2010; S. 230-238.

[Gu02] Guzdial, M.; Soloway, E.: Teaching the Nintendo generation to program. In Commun. ACM 45(4), 2002; S. 17-21.

[KP05] Kelleher, C.; Pausch, R.: Lowering the barriers to programming. In ACM Computing Surveys 37(2), 2005; S. 83-137.

[Ko09] Kohl, L.: Kompetenzorientierter Informatikunterricht in der Sekundarstufe I unter Verwendung der visuellen Programmiersprache Puck. Jena, 2009.

[MPS06] Mannila, L.; Peltomäki, M.; Salakoski, T.: What about a simple language? Analyzing the difficulties in learning to program. In Computer Science Educ. 16(3), 2006; S. 211-227.

[MST97] Meisalo, V.; Sutinen, E.; Tarhio, J.: CLAP: teaching data structures in a creative way. Proc. Proceedings of the 2nd conference on Integrating technology into computer science education, Uppsala, Sweden, 1997.

[Re07] Resnick, M.: Sowing the Seeds for a More Creative Society. Proc. Learning & Leading with Technology, International Society for Technology in Education (ISTE), 2007.

[Ro07] Romeike, R.: Animationen und Spiele gestalten. Ein kreativer Einstieg in die Programmierung. In LOG IN 146/147, 2007; S. 36-44.

[RS06] Romeike, R.; Schwill, A.: "Das Studium könnte zu schwierig für mich sein" - Zwischenergebnisse einer Langzeitbefragung zur Studienwahl Informatik. Proc. HDI 2006: Hochschuldidaktik der Informatik, München, Lecture Notes in Informatics, 2006.

[SS89] Soloway, E.; Spohrer, J. C.: Studying the novice programmer. L. Erlbaum Associates Inc. , Hillsdale, NJ, USA, 1989.

[Wi96] Winslow, L. E.: Programming pedagogy - a psychological overview. In SIGCSE Bull. 28(3), 1996; S. 17-22.