Punkt, Punkt, Semikolon, Strich – Grafikorientierte Einführung in die Programmierung mit Processing

Daniel Wunderlich[1]

Abstract: Die Programmiersprache Processing ist darauf spezialisiert, einen möglichst einfachen Einstieg in die textuelle Programmierung zu bieten. Seit Beginn der Entwicklung 2001 am MIT Media Lab wird der Fokus hierbei auf die Erzeugung von Grafiken und Animationen gelegt. Processing erweitert hierzu Java um einfache Methoden zum Einbezug von Benutzereingaben durch Maus und Tastatur, wobei die Syntax und grundlegende Konzepte der Sprachen identisch sind. Gleichzeitig entfallen viele Aspekte der Java-Programmierung, die gerade beim Einstieg in die Programmierung von Lernenden nur schwer nachvollzogen werden können.

Der Workshop stellt einen Lehrgang zur Einführung in die Programmierung mit Processing in Sekundarstufe II vor, der im Rahmen des Referendariats mit ausführlicher Dokumentation entstand.

Keywords: Processing, Programmierung, Einführung, Grafik, Java, Sekundarstufe II

1 Programmierung in der Schule

Die Programmierung stellt, wie das Verfassen von Texten in sprachlichen Fächern und Naturgesetze in der Physik, ein zentrales Konzept der Informatik dar. Wichtig ist, dass sie im Kontext der schulischen Ausbildung insbesondere zu Beginn nicht als Produktschulung einzelner (meist unnötig komplexer) Entwicklungsumgebungen (IDEs – engl. *integrated development environment*) und im Hinblick auf programmiertechnische Finessen aufgefasst und gelehrt wird. Vielmehr sollten allgemeine und grundlegende Prinzipien der Informatik vermittelt werden [RNH01, S. 1 f.]. Zur Implementierung von Algorithmen ist sie außerdem Teil der *fundamentalen Ideen* der Informatik nach SCHWILL [Hu07, S. 82 ff.].

Gleichzeitig wird durch die Programmierfähigkeit die Kompetenz gefördert, die Arbeitsweise eines Computers zu verstehen – eine Grundlage zur „kritischen und verantwortungsvollen Nutzung von informationstechnischen Hilfsmitteln“ [Mi04, S. 438], wie sie z. B. im baden-württembergischen Bildungsplan 2004 gefordert wird. Diese Fähigkeit gewinnt gerade in einer zunehmend von Software geprägten Welt ständig an Relevanz.

Aufgrund ihrer Relevanz findet sich die Programmierung deshalb auch im Inhaltsbereich *Sprachen und Automaten* der Bildungsstandards der Gesellschaft für Informatik e. V. (GI) [Ge08] und der *2. Leitidee „Algorithmen und Daten“* des baden-württembergischen Bildungsplans 2004 wieder [Mi04, S. 439]. Nach letzterem sollen die SchülerInnen insbesondere „elementare Datentypen und Strukturen zur Ablaufsteuerung anwenden“ und „Algorithmen entwerfen und in Programme umsetzen“ können.

[1] Gymnasium Walldorf, Schwetzinger Str. 95, 69190 Walldorf, daniel.wunderlich@gymnasium-walldorf.de

2 Processing

Processing ist eine Programmiersprache mit zugehöriger IDE, mit deren Entwicklung 2001 von BEN FRY und CASEY REAS am *MIT Media Lab* (Cambridge, USA) begonnen wurde [FR]. Processing ist freie Open-Source-Software, sodass sich die Entwicklergemeinde stetig vergrößert. Die Programmiersprache wurde von Beginn an mit der Absicht konzipiert, in der Tradition von LOGO [Lo] und BASIC [Ho] als „erste Programmiersprache" einen einfachen Einstieg in die Programmierung zu ermöglichen. Gleichzeitig weist sie viele Parallelen zu gegenwärtig populären IDEs für den schulischen Einsatz wie *Scratch* [MI], *Kara* [Sw], *Greenfoot* [Pr] und *BlueJ* [LP] auf. Der Schwerpunkt von Processing liegt hierbei auf der Erzeugung von interaktiven, animierten Grafiken. Die erste bedeutende Version 1.0 erschien nach siebeneinhalbjähriger Entwicklung im November 2008 [Wi]. Mit Version 2.0 wurde 2012 eine umfassende Überarbeitung der Sprache veröffentlicht. Aktuell ist Version 2.2.1 (Stand: April 2015).

Heute erfreut sich Processing großer Beliebtheit. Eine aktive Community von Entwicklern, Künstlern und Wissenschaftlern verwendet die Programmiersprache für diverse Zwecke. Darüber hinaus steht mittlerweile eine Vielzahl von Erweiterungen zur Verfügung, z. B. Schnittstellen zu Arduino, JavaScript, Python, Microsoft Kinect oder OpenCV [FR].

Processing erweitert Java um diverse Funktionen [Sh08], z. B. das Zeichnen von Grafikprimitiven[2], die Interaktion mit Maus und Tastatur, das Verarbeiten und Anzeigen von Texten, Bildern und Videos und 2D-/3D-Transformationen. Syntax und zentrale Sprachelemente (insb. Variablen, Kontrollstrukturen, Methoden und Objektorientierung) sind mit Java identisch und auch fortgeschrittene Programmiertechniken (z. B. Objektorientierung) können angewandt werden. Beim Starten eines Programms wird der Processing-Code intern zuerst in Java-Code übersetzt und anschließend kompiliert. Dies führt dazu, dass man theoretisch sämtliche Processing-Programme problemlos in „reines" Java übersetzen kann.

Die bereits vorhandene Funktionalität ist auf die Gestaltung von animierten und interaktiven Medien ausgelegt. Dies zeigt sich deutlich im *Processing-Prinzip* (s. Abb. 1): Die Benutzereingabe erfolgt nicht wie in „traditionellen Einstiegsprogrammen" durch Zeicheneingabe, sondern ähnlich den oben aufgeführten IDEs über Maus und/oder Tastatur. Auch die Ausgabe erfolgt nicht textuell, sondern durch eine Animation der Grafik. Klickt man im gezeigten Beispiel z. B. in den hellen Kreis, verändert sich die Hintergrundfarbe von schwarz zu weiß.

Die schlichte Processing-IDE (auch PDE – engl. *processing developement environment*) ist in Abb. 2 zu sehen. Im weißen Bereich wird der Code verfasst, hierbei farblich hervorgehoben und automatisch eingerückt. Über die Schaltfläche ▶ wird das Programm kompiliert und ausgeführt, über ■ gestoppt. Fehler- und Statusmeldungen werden in einer Statuszeile ausgegeben, eine Konsole zur Textausgabe steht zur Verfügung.

Programme in Processing werden – in Anlehnung an den künstlerischen Ursprung – *Skizzen* (engl. *sketch*) genannt. Dieser Begriff wird sowohl für die Quelltextdatei als auch für

[2] Als *grafische Primitiven* bezeichnet man in der Computergrafik elementare geometrische Formen. Meist zählen hierzu Punkte, Strecken, Polygone, Kreise und Ellipsen.

Traditionell

1. *Programmierung*: Textuell
2. *Eingabe*: Textuell

```
> java Eingabebeispiel
> Wie heißt du? Horst_
```

3. *Ausgabe*: Textuell

```
> Hallo Horst!
```

Processing

1. *Programmierung*: Textuell
2. *Eingabe*: Durch Maus/Tastatur

3. *Ausgabe*: Visuell – Animation

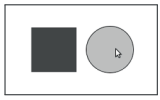
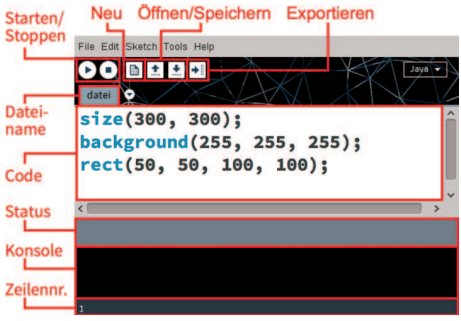
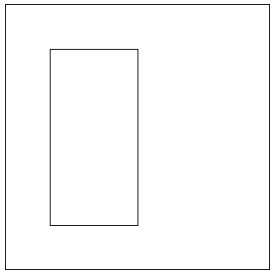


Abb. 1: Benutzerinteraktion traditionell und in Processing



(a) IDE mit Programm (Skizze)



(b) Resultierende Grafik

Abb. 2: Processing IDE mit erstem Programm

die resultierende Grafik verwendet. Im Folgenden werden die Begriffe *Programm* und *Skizze* synonym verwendet. Bevor eine Skizze gestartet werden kann, muss sie gespeichert werden. Für jede Skizze legt Processing hierzu ein Verzeichnis an, in dem der Code in einer Textdatei (Endung `*.pde`) abgelegt wird.

3 Motivation

ARNULF MESTER verwendete Processing bereits vor mehreren Jahren in der Lehre der DHBW Mosbach und stellte die Programmiersprache am Informatiktag 2010 der Fachgruppe der *Informatiklehrerinnen und -lehrer in Baden-Württemberg* der GI in einem Workshop vor [Me10]. Nach einer Einarbeitungsphase im Rahmen des Referendariats waren es insbesondere die folgenden drei Aspekte, welche Processing für den schulischen Einsatz besonders attraktiv erscheinen ließen:

1. *Processing ist einfach zu erlernen*. Dies betrifft insbesondere den Einstieg in die Programmierung. Selbst SchülerInnen ohne Programmiervorkenntnisse erleben relativ schnell Erfolge und werden nicht durch unverständliche Code-Fragmente abgeschreckt. Aufgrund der minimierten Menge an Code reduzieren sich syntaktische Fehler und auch die IDE überzeugt durch ihren Minimalismus.

2. *Processing ist ansprechend*. Bereits in der dritten Doppelstunde sind SchülerInnen in der Lage, Benutzereingaben zu verarbeiten und hierdurch interaktive dynamische Skizzen zu erstellen. Nach der vierten/fünften Doppelstunde können SchülerInnen einfache Spiele erstellen. Berücksichtigt man den gegenwärtigen Boom der Videospielbranche, ist deshalb mit hoher Motivation der SchülerInnen zu rechnen, sie werden in ihrer Lebenswelt abgeholt.

3. *Processing ist anders*. Zum einen entfallen die zu Beginn typischen auf Textausgabe abzielenden Aufgabenstellungen traditioneller Programmierlehrgänge („Gib die Quadratzahlen von 1 bis 20 aus.“), welche häufig mathematische Probleme lösen und für SchülerInnen meistens unattraktiv sind. Zum anderen fördert und fordert Processing Kreativität.

4 Unterrichtsform

Selbstständiger und handlungsorientierter Unterricht stellt offensichtlich eine Notwendigkeit beim Erlernen des Programmierens dar, weshalb die Unterrichtseinheit diesen Ansatz verfolgt und hierbei auf die Definition von HILBERT MEYER zurückgreift: „Handlungsorientierter Unterricht ist ein ganzheitlicher und schüleraktiver Unterricht.“ [Me87]

Die Doppelstunden der Unterrichtseinheit wurden deshalb grundlegend in die folgenden vier Phasen gegliedert. Damit der zeitliche und inhaltliche Umfang der jeweiligen Phasen nicht zu groß wird, wurden die Doppelstunden mehrmals in diese vier Phasen geteilt.

1. *Einstieg*: Die SchülerInnen bekommen die Notwendigkeit des neuen Unterrichtsinhaltes als Motivation aufgezeigt. Wenn möglich, wird dies durch ein kurzes passendes Spiel in Kleingruppen ermöglicht. Außerdem wird als zusätzliche Motivation ein Processing-Programm demonstriert, das nach der Doppelstunde (theoretisch) von den SchülerInnen programmiert werden könnte.

2. *Klassenunterricht*: Grundlegende Aspekte des Themas werden gelenkt durch den Lehrer in der Klasse erarbeitet. Die Beteiligung der SchülerInnen richtet sich hierbei nach dem Inhalt und die Dichte an neuen Informationen ist verhältnismäßig hoch.

3. *Einzelarbeit*: Die SchülerInnen üben die behandelten Inhalte und erarbeiten anschließend neue Aspekte. Dies geschieht durch Aufgaben auf einem Arbeitsblatt. Wenn möglich, werden Inhalte ohne Einsatz des Rechners gelöst. Diese Phase stellt den Schwerpunkt des Unterrichts dar.

4. *Sicherung*: Die Ergebnisse der Einzelarbeit werden gemeinsam besprochen und/oder demonstriert. Ggf. aufgekommene Fragen werden geklärt.

5 Vorhandenes Material und Literatur

Zu Beginn der Unterrichtseinheit konnten keine Materialien ausfindig gemacht werden, die nicht ohne intensives Aufbereiten einen für die Kursstufe angemessenen Programmiereinstieg ermöglicht hätten. Mittlerweile sind mit *Creative Coding* von TILL NAGEL [Na] und *Processing – Programmieren lernen* von MICHAEL KIPP [Ki] zwei vielversprechende Einführungskurse online verfügbar.

Die Auswahl an passender Literatur für Processing hält sich ebenfalls in Grenzen. In deutscher Sprache sind gegenwärtig lediglich zwei Bücher verfügbar: Das sehr kompakte *Processing – eine Einführung in die Programmierung* von ANDRES WANNER [Wa11] eignet sich eher zum schnellen Nachschlagen. Sehr viel ausführlicher, ansprechend gestaltet und empfehlenswert ist *Processing* von ERIK BARTMANN [Ba10]. Als Grundlage der Unterrichtseinheit diente dennoch ein englischsprachiges Werk: *Learning Processing* von DANIEL SHIFFMAN [Sh08]. Zum einen verfolgt SHIFFMAN konsequent das erläuterte Processing-Prinzip, indem bereits sehr früh Benutzerinteraktion durch Maus und Tastatur thematisiert wird. Diese Eingabemethoden werden in den folgenden Themen verwendet und z. B. Verzweigungen durch Farbänderungen aufgrund der Mausposition behandelt. Zum anderen beinhaltet das Buch viele Übungsaufgaben. Diese wurden zwar nur selten direkt übernommen, inspirierten jedoch sehr.

6 Folien und Arbeitsblätter

Der Klassenunterricht wird durch Folien unterstützt. Diese dienen insbesondere als Schulbuchersatz. Die Folien beinhalten zentrale Punkte des jeweiligen Themas, deren eigenständige Erarbeitung nur bedingt sinnvoll ist. Häufig werden Sachverhalte hierbei durch Grafiken veranschaulicht. Um den Klassenunterricht möglichst knapp halten zu können, werden viele Inhalte auf Arbeitsblätter ausgelagert und eigenständig erarbeitet.

Die Einzelarbeitsphasen werden durch Arbeitsblätter organisiert. Alle Teile beginnen mit einer einfachen Einstiegsaufgabe, in der z. B. ein Programm beschrieben, vorgegebene Änderungen am Code vorgenommen oder Code vom Blatt übertragen werden muss. Der Anspruch der folgenden Aufgaben steigt sukzessive: Werden zuerst noch einzelne Schritte vorgegeben, ist in späteren Aufgaben lediglich die zu implementierende Funktionalität beschrieben (ergebnisorientiert). Beendet werden die Einzelarbeitsphasen durch Zusatzaufgaben zur Differenzierung für schnelle SchülerInnen.

7 Themenverteilungsplan und Lernziele

Der vorgesehene Themenverteilungsplan ist in Tab. 1 dargestellt. Der Reihenfolge in [Sh08] folgend, wird den SchülerInnen bereits in der dritten Doppelstunde ermöglicht, Benutzereingaben in Programmen zu verarbeiten. Die Aufgaben in folgenden Doppelstunden greifen ausnahmslos hierauf zurück.

Die Unterrichtseinheit verfolgt folgende übergeordneten Lernziele:

Stunde(n)	Thema
1/2	Grafiken und Farben
3/4	Erste Schritte in Processing
5/6	Interaktivität in Processing
7/8	Variablen
9/10	Verzweigungen
11/12	Schleifen
13/14	Methoden

Tab. 1: Themenverteilungsplan der Unterrichtseinheit

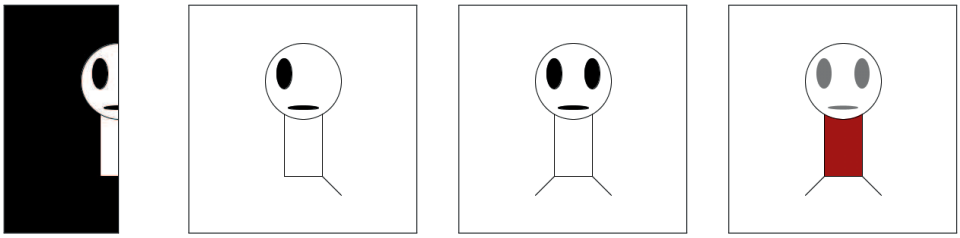
- Die SchülerInnen lernen einen zentralen Bereich der Informatik kennen.
- Die SchülerInnen erarbeiten angeleitet aber eigenständig neue Inhalte.
- Die SchülerInnen erstellen erste einfache Programme.
- Die SchülerInnen realisieren vorgegebene Sachverhalte in Programmen.
- Die SchülerInnen verbalisieren informatische und algorithmische Gegebenheiten.

8 Exemplarische Unterrichtsinhalte

8.1 Erste Schritte in Processing – Hallo Walli!

In der zweiten Doppelstunde machen die SchülerInnen zum ersten Mal Bekanntschaft mit *Walli*. Angelehnt an *Zoog* aus [Sh08], ist Walli eine einfache, achsensymmetrische Figur, welche durch die wichtigsten grafischen Primitive in Processing erzeugt werden kann. Die SchülerInnen erhalten auf dem Arbeitsblatt Quelltext, den sie in Processing übertragen sollen und der Abb. 3a erzeugt. So geben sie ihre ersten Processing-Befehle ein und haben ggf. gleichzeitig die Möglichkeit, syntaktische Fehler zu korrigieren, welche hierbei auftreten. Angeleitet sollen nun zuerst die Fenstergröße und anschließend die Hintergrundfarbe durch vorgegebene Änderung eines Parameters angepasst werden. Letzteres geschieht experimentell – die SchülerInnen sollen sich hierbei an die RGB-Spezifikation von Farben aus der vorangegangenen Doppelstunde erinnern und diese auf einen Grauwert übertragen. Abb. 3b zeigt das Ergebnis dieser Aufgabe.

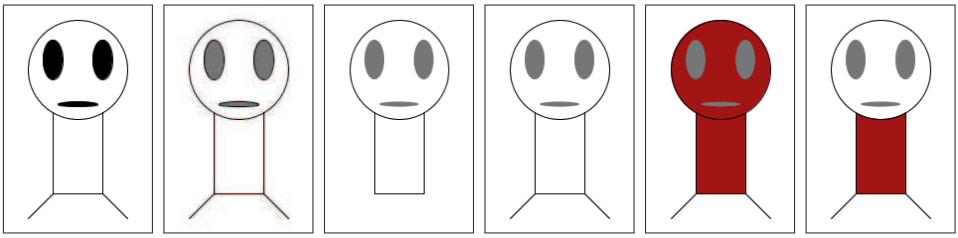
In der folgenden Aufgabe geht es darum, die ersten grafischen Primitive selbst zu programmieren. Walli soll zuerst mit einem zweiten Auge versehen werden. Die Parameter der Methode `ellipse()` werden hierzu noch auf dem Arbeitsblatt vorgegeben. Da bereits ein Auge vorhanden ist, muss nur der Mittelpunkt angepasst werden, die Symmetrie der Figur vereinfacht die Aufgabe zusätzlich. Dann wird es etwas schwieriger: Ein zweites Bein soll ergänzt werden. Zwar ist bereits ein Bein vorhanden, die Bedeutung der Parameter müssen sich die SchülerInnen jedoch selbst herleiten. Das Ergebnis dieser Aufgabe ist in Abb. 3c zu sehen.



(a) Ausgang (b) 2. Aufgabe (c) 3. Aufgabe (d) 4. Aufgabe

Abb. 3: Erste Schritte in Processing mit der Figur *Walli*

In einer weiteren Aufgabe sollen schrittweise die wichtigsten Methoden im Zusammenhang mit Farben behandelt und in der Skizze mit Walli ergänzt werden (s. Abb. 4). Die SchülerInnen machen sich hierbei zuerst mit der Processing-Referenz vertraut und entnehmen ihr die Funktion der Methode `fill(g)`. Durch `fill()` sollen dann Wallis Augen durch den Wert 128 grau gefärbt werden. Die Änderung der Füllfarbe lässt die Strichfarbe der Augen (Schwarz) als Umrandung sichtbar werden. Das Zeichnen der Striche (insb. der Umrandung gefüllter Primitiven) soll im Anschluss durch die Methode `noStroke()` deaktiviert werden. Hierdurch werden auch Wallis Beine nicht mehr gezeichnet, was durch `stroke(0)` vor dem Zeichnen der Beine rückgängig gemacht werden muss. Danach soll Wallis Körper in einer beliebigen Farbe gefärbt werden. Der Aufruf von `fill()` vor dem Zeichnen des Körpers hat hierbei aufgrund der Reihenfolge der Methodenaufrufe (bewusst) zur Konsequenz, dass auch der Kopf gefärbt wird. Ein eingefügtes `fill(255)` vor dem Zeichnen des Kopfes vollendet Walli.



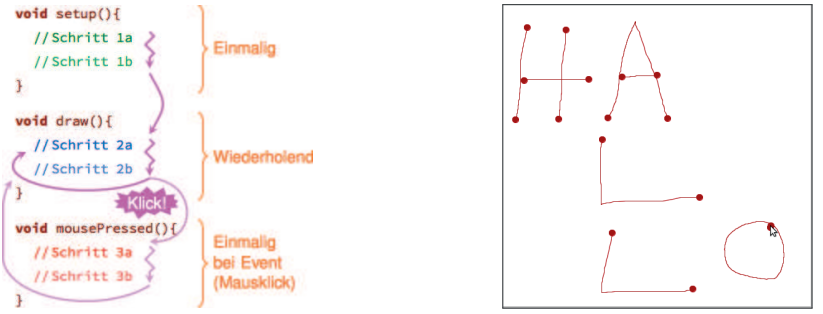
(a) Ausgang (b) `fill(128)` (c) `noStroke()` (d) `stroke(0)` (e) `fill(col)` (f) `fill(255)`

Abb. 4: Schrittweises Färben von *Walli*

8.2 Paint Light – Interaktive Skizzen durch Events

Die dritte Doppelstunde thematisiert interaktive Skizzen in Processing. Abb. 5a zeigt den schematischen Ablauf eines Processing-Programms. Nachdem einmalig die `setup()`-Methode durchlaufen wurde, springt Processing in die `draw()`-Methode, welche als Endlosschleife wiederholend abgearbeitet wird. Spezielle Methoden (z. B. `mousePressed()` oder `keyPressed()`) werden einmalig ausgeführt, wenn das entsprechende Event eintritt.

Bereits am Ende der dritten Doppelstunde sind die SchülerInnen in der Lage, ein einfaches Zeichenprogramm – *Paint Light* – zu programmieren (s. Abb. 5b). Dieses kann bei gedrückter Maustaste den Weg der Maus nachzeichnen und die Leinwand durch einen Tastendruck der Tastatur löschen.



(a) Programmfluss mit Event (b) Paint Light mit Events

Abb. 5: Events in Processing

8.3 Push the button – Erleuchtende Verweigung

Die Aufgabe *Push the button* der Doppelstunde über Vezweigungen ist als fortgeschrittene Aufgabe ergebnisorientiert und mit Zusatzteil konzipiert. Sie zeigt außerdem, wie einfach in Processing Zusatzaufgaben zur Differenzierung erstellt werden können:

Aufgabe: Push the button *In Skizze* `sketch_13_push_the_button_a.pde` *siehst du – mit etwas Phantasie – einen grünen Schalter in einem dunklen Raum. Dieser Schalter soll, sobald er mit der Maus angeklickt wird, das Licht im Raum einschalten (also den Hintergrund weiß färben). Das Licht soll brennen, bis er erneut angeklickt wird.*

1. *Implementiere diese Funktion unter Verwendung der vorhandenen Variablen.*
2. Zusatz: *Erweitere die Skizze um einen sog. Hover-Effekt des Schalters. D. h. der Schalter soll heller gefärbt werden, wenn sich der Mauszeiger darüber befindet.*

Processing stellt die Systemvariablen `mouseX` und `mouseY` zur Verfügung. Neben Variablen für die Position und Größe des Schalters (`buttonX` etc.) ist außerdem die boolesche Variable `lightOn` bereits vorgegeben. Mithilfe einer Vezweigung und logischer Operatoren kann in `mousePressed()` der Wert von `lightOn` wie in List. 1 invertiert und innerhalb von `draw()` der Hintergrund der Skizze entsprechend gefärbt werden.

```
if (mouseX > buttonX && mouseX < buttonX + buttonW &&
  mouseY > buttonY && mouseY < buttonY + buttonH) {
  lightOn = !lightOn;
}
```

List. 1: Mausklick auf „Taste“ durch Verzweigung überprüfen

9 Erfahrungen

Die Unterrichtseinheit wurde im Schuljahr 2013/2014 ursprünglich für einen zweistündigen Informatikkurs im ersten Jahr der Kursstufe an einem allgemeinbildenden Gymnasium (G8) konzipiert. Der Kurs bestand aus 14 SchülerInnen und wurde in einem Informatik-Raum mit fest installierten Rechnern unterrichtet.

Im gleichen Schuljahr wurde die Unterrichtseinheit in einem Parallelkurs unterrichtet. Im Schuljahr 2014/2015 wurde sie in einem weiteren zweistündigen Informatikkurs erneut erprobt, ergänzt und verbessert.

Insbesondere zwei negative Aspekte offenbarten sich beim Einsatz von Processing: Zum einen war – insbesondere zu Beginn der Unterrichtseinheit – die korrekte Bestimmung von Koordinaten der nötigen Grafikprimitiven sehr zeitintensiv und zeigte aufgrund des Trial-and-Error-Vorgehens der meisten SchülerInnen nur einen geringen Lerneffekt. Zum anderen zeigte sich, dass der grafische Ansatz von Processing beim Erlernen oder Überprüfen neuer Inhalte auch im Weg stehen kann. Man muss beim Erstellen neuer Aufgaben gezielt darauf achten, dass dies nicht der Fall ist. Sollen die SchülerInnen z. B. beim Erlernen von Verzweigungen innerhalb der Verzweigung ein Grafikprimitiv zeichnen, kann die Zeichnung für sie anspruchsvoller als das eigentlichen Thema sein. Somit entsteht ein falscher Fokus. Entgegenwirken kann man dieser Problematik, indem man sie bei der Aufgabengestaltung jederzeit berücksichtigt und ggf. nötige Teile des Quelltextes vorgibt. Gänzlich beheben kann man sie jedoch nicht.

Ebenfalls zeigte sich, dass die Zeit für die Themen in Abschnitt 7 sehr knapp bemessen ist. Der Kurs der ersten Durchführung arbeitete sehr gewissenhaft und auf hohem Niveau, sodass die Verteilung insgesamt eingehalten werden könnte. Bei den folgenden Durchläufen war dies (auch aufgrund der nicht optimalen Ausstattung durch einen Laptopwagen und dem resultierenden Zeitverlust) nicht möglich. Durch die Einteilung der Themen in mehrere Phasen ist eine flexible Aufteilung der Inhalte jedoch leicht möglich.

Alle drei Durchläufe der Unterrichtseinheit offenbarten vor allem sehr großes Interesse der SchülerInnen. Die im Abschnitt 3 erhofften Vorteile Processings bestätigten sich größtenteils. Über die gesamte Unterrichtseinheit arbeiteten die SchülerInnen motiviert an den vorgelegten Aufgaben und im Klassenunterricht mit. Sicherlich ist dies auch auf die allgemeine Lernatmosphäre des jeweiligen Kurses zurückzuführen. Häufige Äußerungen einzelner SchülerInnen während des Unterrichts zeigten jedoch, dass sie sich mit Freude mit den neuen Inhalten beschäftigen. Abschließende Klausuren und über Moodle durchgeführte Umfragen über die Unterrichtseinheit fielen ebenfalls größtenteils positiv aus. Insgesamt verlief die Unterrichtseinheit erfolgreich.

Literaturverzeichnis

[Ba10] Bartmann, Erik: Processing. O'Reilly, 1. Auflage, 2010.

[FR] Fry, Ben; Reas, Casey: Processing. `http://www.processing.org`.

[Ge08] Gesellschaft für Informatik (GI) e. V.: Grundsätze und Standards für die Informatik in der Schule – Bildungsstandards Informatik für die Sekundarstufe I. 2008. http://www.sn.schule.de/~istandard/docs/bildungsstandards_2008.pdf.

[Ho] Hoffmann, Reiner: BASIC. http://www.madeasy.de/2/basic.htm.

[Hu07] Hubwieser, Peter: Didaktik der Informatik – Grundlagen, Konzepte, Beispiele. 3. Ausgabe. Springer, 2007.

[Ki] Kipp, Michael: Processing – Programmieren lernen. http://michaelkipp.de/processing.

[Lo] Logo Foundation: Webpräsenz. http://el.media.mit.edu/logo-foundation.

[LP] La Trobe University, Melbourne, Australien; Programming Education Tools Group, University of Kent, Canterbury, England: BlueJ. http://www.bluej.org.

[Me87] Meyer, Hilbert: UnterrichtsMethoden, Jgg. 2. 1987.

[Me10] Mester, Arnulf: Programmieren lernen mit Processing. Workshop des Informatiktags der ILL-BW, 2010.

[MI] MIT Media Laboratory, Cambridge, USA: Scratch. http://scratch.mit.edu.

[Mi04] Ministerium für Kultus, Jugend und Sport Baden-Württemberg: Bildungsplan 2004 – Allgemein bildendes Gymnasium. 2004. http://www.bildung-staerkt-menschen.de/service/downloads/Bildungsplaene/Gymnasium/Gymnasium_Bildungsplan_Gesamt.pdf.

[Na] Nagel, Till: Creative Coding. http://btk.tillnagel.com.

[Pr] Programming Education Tools Group, University of Kent, Canterbury, England: Greenfoot. http://www.greenfoot.org.

[RNH01] Reichert, Raimond; Nievergelt, Jürg; Hartmann, Werner: Programming in schools – why, and how? In (Hartmann, Werner; Näf, Michael; Reichert, Raimond, Hrsg.): Enseigner l'informatique, S. 143–152. Springer, 2001. http://www.swisseduc.ch/informatik/karatojava/docs/programming_why_how.pdf.

[Sh08] Shiffman, Daniel: Learninig Processing – A Beginner's Guide to Programming Images, Animation, and Interaction. 1. Ausgabe. Morgan Kaufmann, 2008.

[Sw] SwissEduc: Kara. http://www.swisseduc.ch/informatik/karatojava/kara.

[Wa11] Wanner, Andres: Processing – eine Einführung in die Programmierung. lulu.com, 1.1. Auflage, 2011.

[Wi] Wikipedia (de): Processing. https://de.wikipedia.org/wiki/Processing.