ScratchTab – Eine Tablet-basierte Anwendung zum Erlernen von Programmierkonzepten

Philip Brauner, Hendrik Thüs, Martina Ziefle, Ulrik Schroeder

RWTH Aachen
{brauner,ziefle}@humtec.rwth-aachen.de
{thues,schroeder}@cs.rwth-aachen.de

Abstract: ScratchTab ist eine mobile und vielseitig einsetzbare Programmierumgebung für Tablet-Geräte. Dieser Artikel stellt die Motivation für eine derartige Umgebung dar, beschreibt die Anforderungen und den Entwicklungsprozess, sowie die Ergebnisse einer Nutzerstudie mit Programmierneulingen. Die Ergebnisse der Nutzerstudie zeigen, dass eine derartige mobile Programmierumgebung einerseits das Interesse steigert, sich mit Programmierung zu beschäftigen, und andererseits die Sensoren und Aktuatoren eines Tablet-Geräts neue und vielfältige Einsatzszenarien in der Programmierausbildung bieten können.

1 Motivation

In der heutigen Zeit hat die Informatik in nahezu jeden Bereich des alltäglichen Lebens Einzug gehalten. Elektronische Gegenstände, teilweise programmierbar, werden tagtäglich von der breiten Menge der Bevölkerung genutzt. Die Informatik ist somit ein grundlegender Bestandteil der heutigen Gesellschaft, den es zu fördern gilt. Im Bereich der MINT-Fächer und insbesondere in der Informatik sind Programmierkenntnisse elementar. Um Programmieranfänger möglichst früh für solch eine Tätigkeit zu begeistern, ist es wesentlich, die Interessierten nicht zu überfordern. Programmierparadigmen können Anfänger leicht überlasten und sollten demnach in einer einfachen Art und Weise dargebracht werden. Das hier dargestellte Projekt ScratchTab bietet eine solche Umgebung. Mit den richtigen Schnittstellen können jedoch nicht nur Programmieranfänger angesprochen werden. Auch Personen verschiedensten Alters, die nicht die Absicht haben, sich in die Programmierung einführen zu lassen, können ihren Vorteil von diesem System haben. Es ist beispielsweise denkbar, bestimmte Abläufe zu automatisieren. Die Regeln für diese Prozeduren könnten durch ScratchTab definiert werden, ohne dass die bedienende Person Vorkenntnisse der Programmierung benötigt.

2 Konzept

Bei der Konzeption von ScratchTab stehen drei Aspekte im Vordergrund. Erstens soll die Programmierumgebung eine geringe Einstiegshürde aufweisen um auch von Personen mit geringer Erfahrung im Umgang mit digitalen Medien und insbesondere ohne Programmierkenntnisse nutzbar sein. Zweitens soll die Umgebung auf Tablets basieren und sich von den sonst in der Ausbildung von Programmieranfängern eingesetzten PCs

oder Laptops absetzen. Drittens sollen die auf Tablets inhärent verfügbaren Schnittstellen und Sensoren ausgenutzt und für die Programmierung bereitgestellt werden.

Zur Erfüllung des ersten Punkts, der geringen Einstiegshürde, basiert ScratchTab auf der vom MIT Media Lab entwickelten Programmiersprache „Scratch“ [HR08], die eine Puzzlemetapher nutzt. Programme werden nicht textuell formuliert, sondern durch Befehlskacheln zusammengestellt. Diese haben je nach Funktion unterschiedliche Formen sowie unterschiedliche Aussparungen zur Verzahnung mit anderen Kacheln. Hierdurch wird sichtbar gemacht, welche syntaktische Struktur ein Programm haben kann; Syntaxfehler sind durch diese Restriktionen ausgeschlossen. Die Lernenden der Programmiersprache müssen sich so weniger auf die Syntax konzentrieren und verfügen so nach der Cognitive Load Theory [CS91] über mehr lernbezogene kognitive Kapazität um sich mit dem primären Lerngegenstand, der Semantik der Sprache, auseinanderzusetzen.

Als zweites Ziel wurde festgelegt, dass ScratchTab auf Tablet-Geräten nutzbar sein soll. Diese mobilen Endgeräte erfreuen sich wachsender Beliebtheit. Im Jahr 2011 wurden in Deutschland mehr Tablets als Netbooks verkauft und zusammen mit Smartphones erobern diese zunehmend den Markt der Normalverbraucher [Bu11]. Ein Tablet zu besitzen ist nichts Besonderes und deutet nicht auf ein bestimmtes Interessengebiet des Besitzers hin, die Bedienbarkeit und Beherrschbarkeit ist für Personengruppen jeglichen Alters möglich, was bei herkömmlichen PCs oder Laptops nicht immer gegeben ist. Der mobile Aspekt der Tablets erleichtert es den Nutzern weiterhin, sich leichter mit anderen Personen auszutauschen und die erstellten Ergebnisse zu vergleichen. Des Weiteren ermöglichen solche Geräte den Einsatz in praxisnahen Szenarien. Beispielsweise kann ein System auf einem Tablet unabhängig vom Ort eingesetzt werden.

Ausgehend von Tablets als Medium wurde als drittes Ziel definiert, dass die integrierten Sensoren, Aktuatoren und Schnittstellen direkt in die Programmiersprache integriert werden sollen. Intention ist, dass das Programmieren-Lernen so besonders motivierend gestaltet werden kann. Die Programmierung von Gegenständen außerhalb des eigentlichen Programmiermediums steigert nicht nur die Lernmotivation, sondern gleichzeitig auch den Lernerfolg [Br10]. So werden LEGO Roboter erfolgreich für den Programmierunterricht für Schülerinnen und Schüler eingesetzt [HS05, LS09]. Diese empfinden es als reizvoll, eben nicht den PC zu programmieren, sondern den Robotern autonomes Verhalten beizubringen.

3 Realisierung

Im vorangegangenen Abschnitt wurden Anforderungen an ScratchTab definiert. In diesem Abschnitt wird nun die Umsetzung in Bezug auf die Gestaltung der Benutzeroberfläche, die verfügbaren Programmbausteine und die Programmausführung dargestellt.

3.1 Benutzeroberfläche

Die Benutzeroberfläche gliedert sich in drei Komponenten (vgl. Abbildung 1): Eine Steuerungsleiste mit den verfügbaren Programmbausteinen, sowie die Arbeitsfläche zum Zusammenstellen der Programme.

Die Steuerungsleiste dient dem Starten und Beenden von Programmen, der Konfiguration der Bluetooth-Verbindung zu anderen Geräten, sowie dem Verstellen der Zoomstufe. Zukünftig werden noch Funktionen zum Laden und Speichern von Programmen ergänzt. In der Seitenleiste befinden sich die nach Kategorien sortierten verfügbaren Befehlsblöcke. Durch diese Liste kann vertikal gescrollt werden; durch einen Tap auf den Namen einer Kategorie ist es ebenfalls möglich, diese direkt anzusteuern.

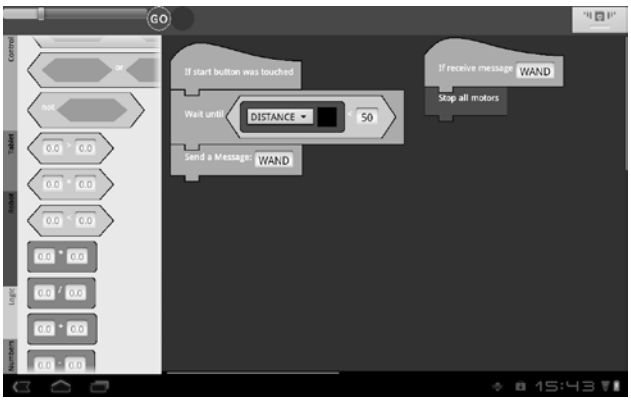

Abbildung 1: Oberfläche der Programmierumgebung ScratchTab (oben: Steuerungsleiste, links: Leiste mit verfügbaren Programmbausteinen, große Fläche rechts: Arbeitsfläche)

3.2 Programmbausteine

Analog zu Scratch bedient sich ScratchTab einer Puzzlemetapher und verfügt über viele Bausteine, aus denen die Lerner Programme zusammenstellen können. Form und etwaige Aussparungen der Blöcke stellen sicher, dass nur syntaktisch kombinierbare Blöcke tatsächlich kombiniert werden können. Die Kombination von Programmbausteinen erfolgt durch eine Drag-und-Drop-Geste, bei der ein Baustein entweder von der Arbeitsfläche oder aus einer Leiste mit allen verfügbaren Bausteinen herausgezogen und an andere Bausteine angedockt wird (vgl. Abbildung 2). Während des Drag-Vorgangs werden die möglichen Andockpunkte visuell hervorgehoben.

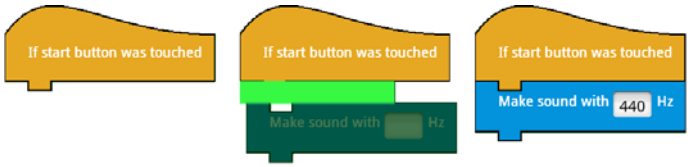

Abbildung 2: Startblock (links), Verknüpfungsvorgang von Startblock mit einem Kommandoblock (Mitte), abgeschlossene Verknüpfung (rechts)

Bausteine können untereinander gereiht werden, was für eine Programmsequenz steht. Einige Bausteine verfügen auch über „Slots“, in die andere Bausteine hierarchisch eingeordnet werden. So enthält ein Schleifenbaustein zwei Slots für die Bausteine im Schleifenrumpf und für die Bausteine im Schleifenkopf (z.B. Abbruchbedingung). In der derzeitigen Entwicklungsstufe sind Programmbausteine für grundlegende Kontrollstruk-

turen wie Bedingungen, Schleifen mit fixen Wiederholungen oder dynamischen Abbruchbedingungen, Logikoperatoren und arithmetischen Funktionen umgesetzt. Ferner können die Sensoren und Aktuatoren des Tablets genutzt werden. Alle Programmbausteine untergliedern sich in Kategorien, die im Folgenden dargestellt werden.

Zur Kategorie „Programmablauf“ gehören Bausteine für IF-THEN-ELSE-Blöcke oder numerischen oder ausdrucksorientierten Schleifen (FOR i bzw. WHILE CONDITION).

Für Berechnungen existieren Bausteine für Additionen, Multiplikationen und dergleichen. Als Operanden dienen entweder vom Programmierer vorgegebenen konstante Zahlen oder auch Sensorwerte des Tablets. Analog können mit den Bausteinen für logische Vergleiche Zahlen auf Gleichheit oder Größenunterschiede untersucht werden. Ferner existieren boolesche Operatoren für das logische UND, ODER und NICHT.

Die Sensoren und Aktuatoren des Tablets können über Programmbausteine der Kategorie „Tablet“ angesteuert werden. Derzeit stehen zur Ausgabe die Bausteine „Make sound“, sowie „Make a screenflash“ zur Verfügung. Zukünftig sollen auch komplexerer Soundsamples oder Vibrationen angeboten werden. Als Sensoren stehen die auf Tablet-Geräten häufig verfügbaren Beschleunigungssensoren, ein Kompass oder ein Lichtsensor zur Verfügung.

Wie im Kapitel Konzept dargelegt wurde, sollen ScratchTab-Programme sich nicht nur auf das Tablet auswirken, sondern auch auf Ereignisse außerhalb des Tablets reagieren können. Als Machbarkeitsbeweis wurde eine Verknüpfung zu einem Lego Mindstorms Roboter realisiert. Durch Bausteine der Kategorie „Roboter“ kann ein über eine Bluetooth-Schnittstelle verbundener Roboter gesteuert und dessen Sensorwerte abgefragt werden. Statt einer direkten Steuerung der einzelnen Motoren wurde auf das vereinfachende Konzept der Turtle-Grafik [Pa80] mit Befehlen wie „TURN LEFT“ oder „DRIVE FORWARD“ zurückgegriffen. Von den Sensoren können die für Entfernung, Geräusche oder für Berührungen verwendet werden. Mit wenigen Kacheln lässt sich so beispielsweise ein Programm schreiben, das in Abhängigkeit der Neigung des Tablets verschieden hohe Töne ausgibt. Auch ist es leicht möglich, über das Tablet einen Roboter so zu programmieren, dass dieser beim Berühren einer Wand umkehrt oder gar autonom aus einem Labyrinth navigiert. Das kreative und motivierende Potenzial von ScratchTab liegt jedoch vor allem in der Kombination der verschiedenen Befehlskacheln.

3.3 Starten von Programmen

Programme können über drei Wege gestartet werden: Erstens über den grünen Startknopf aus der Steuerungsleiste und eines dazugehörigen Startbausteins auf der Arbeitsfläche. Wird der Startknopf ausgewählt, beginnt die Programmausführung bei den Startblöcken der jeweiligen Programmsegmente. Zweitens können Programmsegmente durch das Empfangen von Nachrichten aktiviert werden, die entweder durch Sensorwerte ausgelöst werden oder aber aus dem Programm selbst gesendet werden (z.B. zum Aufrufen von Unterprozeduren). Drittens lassen sich Programmsegmente durch einen Doppel-Tap auf einem Baustein starten. Je nach Programmstruktur kann ein Programmsegment selbstständig terminieren oder durch den Stopp-Knopf in der Steuerungsleiste beendet werden. Zur Verfolgung des Programmablaufs wird zum einen die Ausführungsge-

schwindigkeit künstlich reduziert, so dass die Ausführung eines Programmbausteins etwa *250ms* benötigt. Zum anderen leuchtet der jeweils aktive Baustein für diese Zeitspanne auf.

3.4 Vergleichbare Applikationen

Die beschriebene Puzzlemetapher wird nicht nur in der schon beschriebenen Applikation Scratch eingesetzt. Andere Beispiele von Applikationen sind etwa Googles App Inventor[1] oder das von Microsoft entwickelte TouchDevelop[2]. Beide Projekte können zur Entwicklung von Applikationen für mobile Geräte genutzt werden. Das bedeutet, sie bieten eine andere Möglichkeit der Programmierung auf stationären PCs. Es können während dieser Tätigkeit keine Mehrwerte von mobilen Geräten genutzt werden, da diese erst für die Ausführung eingesetzt werden und nicht schon während der Entwicklung.

4 Evaluation

Die Gebrauchstauglichkeit der hier vorgestellten Programmierumgebung wurde regelmäßig untersucht. Während der Entwicklung wurden Nutzertests und Expertenreviews periodisch durchgeführt, um die vielfältigen Designentscheidungen zu fundieren. Abschließend wurde eine formale Nutzerstudie mit 10 Probanden durchgeführt, bei der drei Programmieraufgaben mit ScratchTab bearbeitet werden sollten.

4.1 Evaluation während des Entwicklungsprozesses

Parallel zur Entwicklung wurde die Gebrauchstauglichkeit durch Expertenreviews und Nutzertests in periodischen Abständen überprüft. Prüfungsgegenstand waren etwa Darstellungsgröße der Programmierbausteine oder die Frage, wie ineinander verschachtelte Bausteine (z.B. FOR-Schleifen) darzustellen sind. Die Ergebnisse flossen direkt in die weitere Entwicklung ein. So wurde etwa ausgehend von Nutzerwünschen eine stufenlose Zoomfunktion der Arbeitsfläche geschaffen. Zur Darstellung von verschachtelten Bausteinen präferierte in einer Onlineumfrage (n=94) mit 70% eine deutliche Mehrheit eine Darstellung, bei der einerseits alle Blöcke eine konstante Breite haben und andererseits die Verschachtelungstiefe direkt erkennbar ist.

4.2 Summative Evaluation des derzeitigen Prototypen

ScratchTab hat hinsichtlich Programmfunktionalität und -stabilität einen Reifegrad erreicht, der es ermöglicht, die Umgebung summativ und stärker formalisiert zu evaluieren. Die Evaluation ist explorativ angelegt und soll einen Eindruck von der Gebrauchstauglichkeit liefern, Verbesserungspotential identifizieren und weitere Anwendungsszenarien aufzeigen.

---

[1] http://info.appinventor.mit.edu/
[2] http://research.microsoft.com/en-us/projects/touchdevelop/

Für die Studie wurden den 10 Probanden sequenziell drei Aufgaben steigernder Komplexität gestellt. In der ersten Aufgabe sollten sich die Probanden mit der Umgebung vertraut machen und ein Programm erstellen, das über den Lautsprecher drei aufsteigende Töne ausgibt. In Aufgabe 2 war über ein Schleifenkonstrukt und einen darin verschachtelten Noten-Block wiederholt ein Ton mit gleichbleibender Tonhöhe auszugegeben. Aufgabe 3 erfordert die Integration des Neigungssensors in das Programm; es war solange ein Ton auszugeben, bis das Tablet um einen gewissen Winkel geneigt wurde (die Musterlösungen zu den Aufgaben finden sich in Abbildung 3). Zur Lösung der ersten Aufgabe sind mindestens 4 Blöcke nötig (Start-Block, sowie 3 Noten-Blöcke). Die zweite Aufgabe erfordert 3 Blöcke (Start-Block, Schleife und ein Noten-Block). Die dritte und letzte Aufgabe erfordert 5 Blöcke (Start-Block, Wiederhole-Bis-Block mit einem Kleiner-Block für die logische Prüfung und darin der Block für die Neigungssensorabfrage, sowie ein Noten-Block).

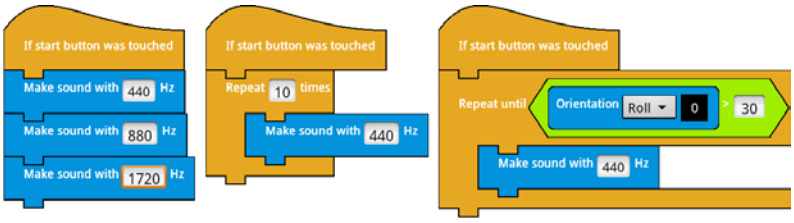

Abbildung 3: Musterlösungen für die Aufgaben 1, 2 und 3 (v.l.n.r.)

Eine Einführung in die Programmierumgebung fand nicht statt, stattdessen sollten die Probanden die Funktionsweise der Umgebung selbstständig erkunden. Erst auf Anfrage oder bei wiederkehrenden Fehlern gaben die Versuchsleiter kurze Hinweise.

An der Studie nahmen 10 Personen teil (*4* Frauen und *6* Männer). Die Probanden waren Studierende und Mitarbeiter unseres Lehrstuhls. Das Alter reichte von *23* bis *31* Jahren. Von den Probanden gab die Hälfte an, über kein oder lediglich elementares Programmierwissen zu verfügen. Vier Personen bewerteten ihre Programmiererfahrung als „mittel“ und eine Person gab an, über viel Programmierwissen zu verfügen. Letztgenannte Person hat Informatik studiert und programmiert sowohl privat wie auch beruflich.

Wenig überraschend hatte der Informatiker keine Schwierigkeiten, die Aufgaben zu lösen. Selbst die in der dritten Aufgabe nötige mehrfache Verschachtelung von Bausteinen bereiteten ihm keine Schwierigkeiten. Auch die anderen Probanden mit keiner oder wenig Programmiererfahrung konnten die Aufgabe 1 problemlos lösen. Aufgabe 2 wollte etwa ein Drittel ohne den Einsatz eines Schleifenbausteins lösen. Nachdem die Funktion dieses Bausteins jedoch erneut erläutert wurde, wurde auch dieser erfolgreich eingesetzt. Die 3. Aufgabe stellte etwa die Hälfte der Probanden vor größere Schwierigkeiten, die nur mit Hilfestellungen gemeistert werden konnten. Zwar kamen alle Probanden auf die Idee, ein Schleifenkonstrukt einzusetzen, jedoch musste zum einen die Wahl des richtigen Konstrukts gelegentlich unterstützt werden, zum anderen wollten die Probanden häufig direkt den entsprechenden Sensorwert (Zahlenwert) in die Schleifenbedingung (Boolescher Wert) einsetzen. Die hier nötige Umwandlung des Sensorwerts in einen booleschen Ausdruck durch einen Vergleichsbaustein musste erst und teils wie-

derholt erläutert werden. Hier scheint der Einsatz einer visuellen Programmiermetapher der Forschung die Möglichkeit zu eröffnen, die kognitiven Prozesse beim Lernen der Programmierung besser beobachten zu können, als bei einer textuellen Programmiersprache. Festzustellen ist, dass nicht nur der Informatiker, sondern alle Probanden die drei Aufgaben erfolgreich lösen konnten.

Bei allen Versuchsteilnehmern traten verschiedene Interaktionsfehler auf. So wurde gelegentlich ein einzuschachtelnder Baustein nicht in den vorgesehenen Slot, sondern daneben abgelegt. Offenbar ist die Fläche eines Slots mit $5x5mm^2$ zu klein, um Präzise Drag-und-Drop Operation ausführen zu können, da der Finger den Zielbereich verdeckt.

Gefragt nach möglichen Einsatzbereichen äußerte mit *6* von *10* Probanden die Mehrheit den Einsatz in der Ausbildung von Programmieranfänger/-innen. Hier wurde von einigen Probanden der Einsatz in der Grundschule und im Kinderkarten genannt, um dort bereits frühzeitig die Grundzüge der Programmierung zu vermitteln. Dies würde sich mit den Ansätzen und guten Erfahrungen der ElectronicBlocks [WP03] decken. Weniger offensichtlich waren die mehrfach genannten Vorschläge ScratchTab für Gebäudeautomation einzusetzen. Ebenfalls wurde empfohlen ScratchTab in Richtung eines Baukastensystems für kognitionspsychologische Experimente auszubauen. Es soll ein einfacher Weg gefunden werden, damit Psychologen ohne Programmiererfahrung Experimente frei gestalten können, bei denen verschiedene Stimuli dargeboten werden, Reaktionszeiten bestimmt und in Abhängigkeiten von der Nutzerantwort unterschiedliche Feedbacks gegeben werden können.

Abschließend sollten sich die Probanden auf einer 6-stufigen Likert-Skala dazu äußern, ob ihnen die Arbeit mit ScratchTab Spaß gemacht habe, ob sie die Umgebung gut lernbar fanden und ob sie eine visuelle Programmiersprache für geeigneter halten als textuelle. Alle Probanden äußerten, dass ihnen die Arbeit mit ScratchTab Spaß gemacht habe (keine negativen Äußerung, 2 stimmten eher zu, 2 stimmten zu und 6 stimmten voll zu) und dass die Umgebung schnell zu erlernen sei (keine negativen Äußerungen, 1 Person stimmte eher zu, 3 stimmten zu und 6 stimmten voll zu). Die visuelle Programmierung wird von einer Mehrheit, jedoch nicht von allen Probanden bevorzugt (3 Personen bevorzugen eher eine textuelle Sprache, 7 Personen bevorzugen eine visuelle Sprache).

5 Fazit und Ausblick

Obwohl die hier vorgestellte Programmierumgebung in einem frühen Prototypenstatus ist, zeigt der erste Nutzertest gute Ergebnisse und lieferte Hinweise für Verbesserungen an der Benutzerschnittstelle und Anregungen für weitere Einsatzmöglichkeiten, die weit über den Einsatz in der Programmierausbildung hinausgehen.

Es ist hervorzuheben, dass ScratchTab eine der wenigen Programmierumgebungen ist, die für den Gebrauch auf mobilen Endgeräten mit kleinen Bildschirmen entwickelt wurde. Durch die Möglichkeit, ortsunabhängig zu programmieren, eröffnet die Applikation eine Reihe von Anwendungsgebieten, die mehr Interaktivität zulassen oder Mobilität der Nutzer voraussetzen. Durch die Schnittstellen der mobilen Endgeräte – wie Bluetooth und WLan – wird das Kommunizieren mit anderen Geräten, wie Robotern, vereinfacht.

Nutzer/-innen haben so die Möglichkeit, ihre mit ScratchTab erstellten Programme direkt auf dem ausführenden Gerät zu testen, Erfolge werden so direkt sichtbar.

Während der Nutzertests wurden weitere Schnittstellen vorgeschlagen, die in Zukunft eingearbeitet werden sollen. Denkbar ist etwa die Steuerung von Arduino-Mikrocontrollern, mit denen sich eine Fülle weiterer Möglichkeiten eröffnet.

Danksagungen

Die Autoren bedanken sich bei Alexander Friesen, der große Teile von ScratchTab realisiert hat. Auch danken wir den Teilnehmerinnen und Teilnehmern der Nutzertests.

Literaturverzeichnis

[Br10] Brauner, P. et al.: The effect of tangible artifacts, gender and subjective technical competence on teaching programming to seventh graders. In: Hromkovic, J. et al. (eds.) Proceedings of the 4th International Conference on Informatics in Secondary Schools: Evolution and Perspective 2010 (ISSEP), Zurich. pp. 61-71 Springer, Heidelberg (2010).

[Bu11] Bundesverband Informationswirtschaft, T. und neue M. e. V.: Tablet Computer erobern den Massenmarkt, http://www.bitkom.org/61374_70631.aspx, erstellt am 14.10.2011, zuletzt besucht am 14.08.2012.

[CS91] Chandler, P., Sweller, J.: Cognitive Load Theory and the Format of Instruction. Cognition and Instruction. 8, 4, 293-332 (1991).

[HS05] Hartmann, S., Schecker, H.: Bietet Robotik Mädchen einen Zugang zu Informatik, Technik und Naturwissenschaft? - Evaluationsergebnisse zu dem Projekt Roberta. Zeitschrift für Didaktik der Naturwissenschaften. 11, 7-19 (2005).

[HR08] Hernández, A.M., Resnick, M.: FEATURE: Empowering kids to create and share programmable media. Interactions. 15, 2, 50-53 (2008).

[LS09] Leonhardt, T., Schroeder, U.: go4IT!: Initiierung und nachhaltige Förderung von Interesse an MINT-Fächern bei Mädchen. Informatische Bildung in Theorie und Praxis, Beiträge zur INFOS 2009, 13. GI-Fachtagung - Informatik und Schule. , Berlin (2009).

[Pa80] Papert, S.: Mindstorms: Children, Computers, and powerful Ideas. Basic Books, Inc. New York, NY, USA. (1980).

[WP03] Wyeth, P., Purchase, H.C.: Using developmental theories to inform the design of technology for children. IDC '03: Proceedings of the 2003 conference on Interaction design and children. pp. 93-100 ACM, New York, NY, USA (2003).