Informatik – Biber

Lehrmittel für die informatische Bildung an der Sekundarstufe I

Kommentar für Lehrpersonen

Inhaltsverzeichnis

Programmieren – Wie können Apps körperlich behinderten Menschen helfen? **3**

Umsetzungsvorschläge (bis 4 Lektionen) **4**

Hintergrundwissen zu „Programmieren“ 5

Wie wird eine App entwickelt? *5*

Was ist ein Programm? *6*

Wie programmiere ich? *6*

Wie wiederhole ich eine Anweisung (Schleife)? *10*

Wie kann ich eine Anweisung an eine Bedingung knüpfen (Selektion)? *11*

Hintergrundwissen zur „Ereignissteuerung“ 12

Wie löse ich ein Ereignis aus? *13*

Wie können zwei Spieler miteinander kommunizieren? *14*

Wie programmiere ich eine Interaktion? *14*

Hintergrundwissen zu „Rund um die App“ 15

Was ist eine App? *15*

Was ist Barrierefreiheit? *16*

Wer macht eine App? *17*

Umsetzungshilfen **18**

Reflexion Lernfilm/Perspektiven der Lernenden 18

Experiment 1: „Programmieren“ 18

Experiment 2: „Ereignissteuerung“ 18

Experiment 3: „Rund um die App“ 19

Informatik-Biberaufgaben zum Thema Programmieren 20

Mögliche Vertiefungen **21**

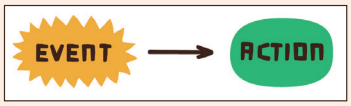
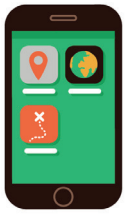
Lösungen **23**

Programmieren – Wie können Apps körperlich behinderten Menschen helfen?

Ausgangskompetenzen

- Schülerinnen und Schüler kennen die grundlegenden Programmelemente Anweisung, Schleife und Fallunterscheidung und können diese in einfachen Programmen einsetzen.
- Schülerinnen und Schüler kennen das Event-Action-Prinzip und können Beispiele dafür nennen.
- Schülerinnen und Schüler können mit der Programmierumgebung Scratch selbstständig Abläufe erstellen. Sie nutzen dazu ihre Kenntnisse zu den grundlegenden Programmelementen.
- Schülerinnen und Schüler verstehen, wie wichtig eine präzise Formulierung ihrer Programme ist, damit die Ausführung wie geplant funktioniert.

Umsetzungsvorschläge (bis 4 Lektionen)

Lernenden-Perspektiven	Anschauen des Lernfilms: Programmieren – Wie kann Informatik körperlich behinderten Menschen helfen? Thema: Programmieren Kennt ihr Apps, die körperlich behinderten Menschen helfen können? Hast du selber eine Idee für eine nützliche App?
Experiment 1	Wie kann man selber eine App entwickeln? Bezug Lernfilm: Anforderungen kennen 02:44 (4.0 Lehrerin)
Experiment 2	Wie funktioniert die Ereignissteuerung? Bezug Lernfilm: Event-Action-Prinzip 03:17 (6.0 Erklärung Ereignis/Handlung) 
Experiment 3	Was braucht es zur Entwicklung einer App? Bezug Lernfilm: Entwicklung einer App 04:51 (9.1 Beispiele) 
Bilderaufgaben	Verschiedene Aufgaben zum Thema Programmieren selbstständig lösen.
Mögliche Vertiefung	Scratch Werkstatt: http://www.swisseduc.ch/informatik/programmier-sprachen/scratch_werkstatt/

Hintergrundwissen zu „Programmieren“

Jede/r nutzt sie. Jede/r liebt sie. Die Apps, die einem das Leben erleichtern. Sei es das Festhalten einer Einkaufsliste oder die neusten Nachrichten aus dem Sport auf das Smartphone zu erhalten. Doch wie funktionieren diese Apps eigentlich? Wie wissen sie, wann sie mich informieren müssen? Apps können nicht nur das Leben von Menschen ohne Beeinträchtigung erleichtern. Sie können bereits so viel, dass auch zum Beispiel Menschen mit körperlicher Behinderung eine Unterstützung durch heutige Apps erhalten. Gerade diese Apps müssen besonders einfach bedienbar und zuverlässig sein.
Eine App zu programmieren, ist eine grosse Herausforderung, die viel Einsatz und intensive Arbeit verlangt. Jede App beinhaltet ein Programm. In diesem Modul werden die ersten Grundsteine für das Programmieren von einfachen Handlungen gelegt. Alle Programme basieren auf diesen wichtigen Grundfunktionen. Wer weiter gehen will, dem steht ein langer, aber sehr spannender Weg bevor.

Wie wird eine App entwickelt?

Eine App besteht normalerweise aus drei Komponenten:

- einer grafischen Benutzerschnittstelle (Frontend)
- Programmlogik (Steuerung)
- Datenschicht (Speicherung der Zustände)

Das Frontend entspricht der kompletten Oberfläche, die für die Nutzerin oder den Nutzer sichtbar ist, d.h. jede Schaltfläche, jede Textbox, alle Farben und so weiter. Eine zentrale Herausforderung der Benutzerschnittstelle ist meist die Usability, d.h. die Bedienbarkeit der App. Je einfacher und selbsterklärender eine App ist, desto besser ist ihre Usability. Wenn sie gut verständlich ist, wird sie von vielen verschiedenen Personen (von jung bis alt, mit oder ohne körperliche Behinderung) verwendet.

Die Programmlogik übernimmt die komplette Steuerung der App. Wenn eine Nutzerin auf der App eine Schaltfläche anklickt, so sagt die Programmlogik, was nun geschieht.

Die Datenschicht dient dazu, Informationen der App zu speichern. Zum Beispiel werden in einer Spiele-App die Punkte einer Highscore-Tabelle in der Datenschicht gespeichert.

HINWEIS: Es ist bereits jetzt offensichtlich, dass das Herstellen einer App wie im Lernfilm viel Zeit und intensive Arbeit erfordert. Seien Sie sich dessen immer bewusst, und arbeiten Sie mit Ihren Schülerinnen und Schülern an kleinen Aufgaben, oder nutzen Sie zwischendurch Onlineumgebungen, mit denen sich einfache Apps umsetzen lassen. Eine komplette App wie im Lernfilm zu entwickeln, ist NICHT Ziel dieses Moduls.

Wir konzentrieren uns in diesem Modul auf Teile der Programmlogik, um Figuren innerhalb einer Spielwiese zu steuern. Die Schülerinnen und Schüler lernen grundlegende Programmierelemente kennen. Diese werden überall genutzt, vom Erstellen von simplen Apps bis

zum Einsatz des Mars Rovers[1]. Die Benutzerschnittstelle hat eine wichtige Rolle bei der Bedienbarkeit einer App, daher soll auch das Frontend einen Platz in diesem Modul erhalten.

Was ist ein Programm?

Ein Programm ist eine geordnete Ansammlung von Anweisungen oder Programmelementen, die ein Computer oder Smartphone befolgt. Die Anweisungen sind meistens simpel formuliert und beschreiben jeweils einen einzelnen Schritt, der getan werden soll. Zum Beispiel „Drehe 90° nach rechts“. Dann kommt die nächste Anweisung. Es gibt unterschiedliche Möglichkeiten, wie Anweisungen abgearbeitet werden können:

- Sequenz (lineare Abfolge von Anweisungen, z.B. 2+2+2 rechnen)
- Schleife/Iteration (Anweisungen werden mehrfach hintereinander ausgeführt, d.h. wiederholt, z.B. Addiere dreimal die Zahl 2)
- Fallunterscheidung/Selektion (Anweisungen werden nur in Abhängigkeit von bestimmten Bedingungen ausgeführt, z.B. die Summe wird nur ausgerechnet, wenn drei Zahlen vorhanden sind).

Um einen Programmfluss (oder Programmablauf) der Reihe nach verstehen zu können, wird mit den Erklärungen bei der einfachen Abfolge/Sequenz begonnen. Versteht man dieses Prinzip der Abfolge von Anweisungen, kann der eigene Programmcode mit Programmflusselementen wie Schleife oder Selektion erweitert werden.

Um die drei Programmflusselemente kennenzulernen, wird die Online-Lernumgebung code.org empfohlen. Diese Lernumgebung umfasst viele verschiedene Tutorials und Übungen, die mit der Klasse oder im Selbststudium durchgearbeitet werden können. Wichtige Programmierkenntnisse werden hier eingeführt. In Einstiegstutorien wird den Schülerinnen und Schülern in 20 kurzen Einstiegsübungen erklärt, wie sie Anweisungen miteinander koppeln können. Ein visueller Baustein entspricht dabei einer Anweisung.

Wie programmiere ich?

Wenn ich programmiere, beschreibe ich eine Abfolge von Anweisungen, die befolgt werden muss. Zum Beispiel kann ich einem anderen Menschen Anweisungen in Form eines Programms geben, welches von ihm abgearbeitet werden soll. Z.B. wie man aus einem Blatt Papier einen Hut oder ein Schiff faltet. Nur wer die Anweisungen exakt und genau in der Reihenfolge wie beschrieben befolgt, wird am Ende aus dem Stück Papier die gewünschte Form gefaltet haben.

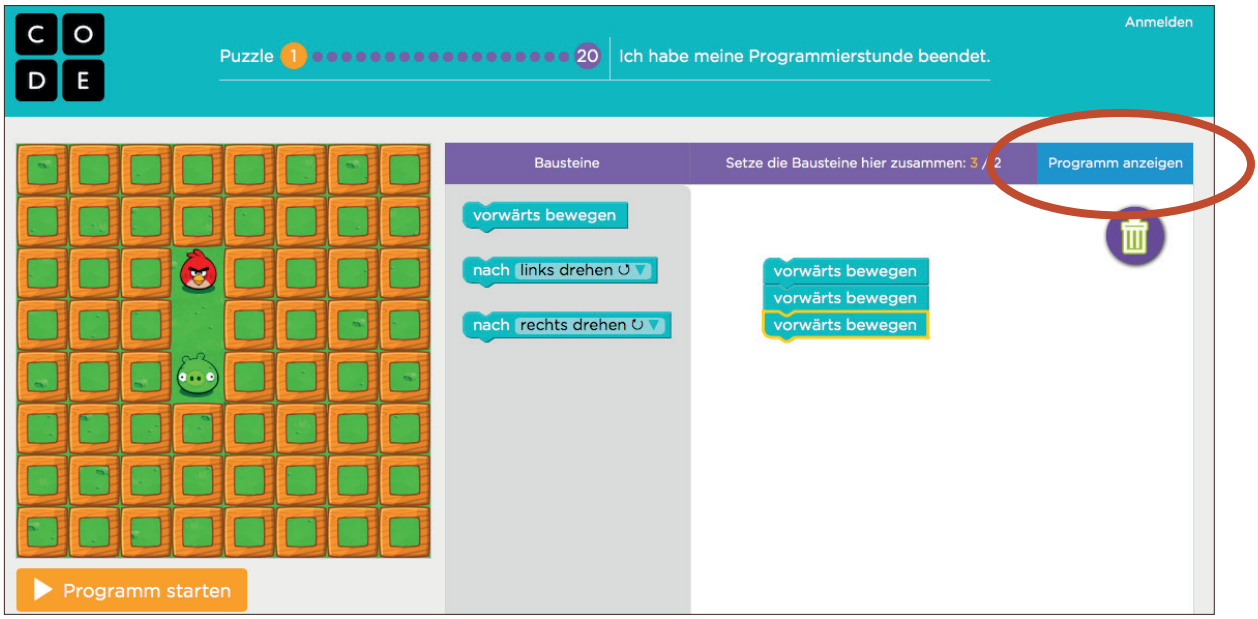
---

[1] Mit Mars Rover ist das ferngesteuerte Fahrzeug gemeint, welches 2003 für die Erforschung des Planeten Mars eingesetzt wurde. Der Mars-Rover wurde per Funk von der Erde aus gesteuert. Er hatte zum Teil auch autonome Fähigkeiten. Die Messergebnisse wurden an die Erde zurückgesendet. (Quelle Wikipedia)

HINWEIS KOPIERVORLAGEN: Für jede der folgenden Strukturen (Sequenz, Schleife und Selektion) sind Kopiervorlagen vorhanden mit Anleitungen, wie man aus einem Stück Papier einen Papierflieger, einen Hut oder ein Schiff faltet. Diese Übung kann als Einstieg genutzt werden, bevor mit den visuellen Bausteinen von code.org begonnen wird. Mit den Faltanleitungen wird klarer, was eine Anweisung zur Folge hat. Die Schülerinnen und Schüler setzen die Anweisungen von Hand um.

Beim Programmieren mit code.org entspricht jeder Baustein einer Anweisung. Im Einstiegstutorial auf code.org haben die Schülerinnen und Schüler Bausteine zur Verfügung, um Anweisungen zusammenzusetzen.

Sie können ihre Bausteine aber auch als Text-Code anzeigen lassen. Dadurch wird ersichtlich, dass man eigentlich Text und Sprache braucht, um zu programmieren. Deshalb spricht man auch von „Programmiersprachen“. Die visuellen Bausteine sind hilfreich, um beim Kennenlernen des Themas „Programmieren“ die Prinzipien kennenzulernen. Später, wenn man geübt ist, wird häufig direkt in Text programmiert.



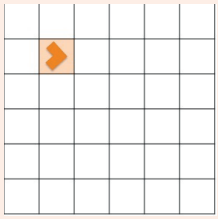
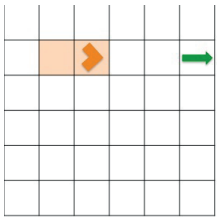
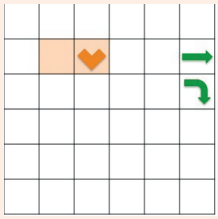
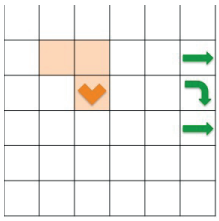
Code.org, visuelle Bausteine: Um die Bausteine als Text anzuzeigen, klickt man einfach rechts oben auf „Programm anzeigen“.

Code.org: Bausteine als Text.

Um mehr als eine Aktion auszuführen, werden so viele Bausteine wie nötig verwendet. Die Befehle werden von oben nach unten abgearbeitet, bis das Programm zu Ende ist. Z.B. „Strecke 5 Finger aus. Neige die Hand nach links. Neige die Hand nach rechts.“

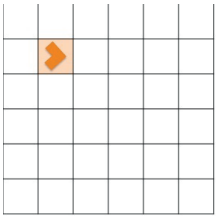
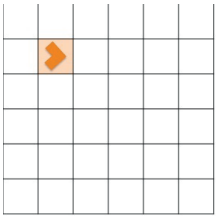
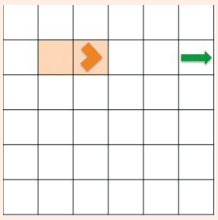
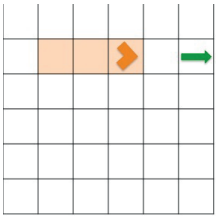
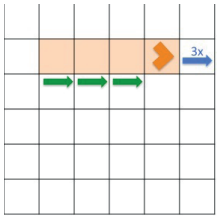
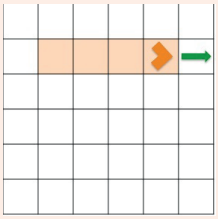
Wie gebe ich einfache Anweisungen in einer Folge (Sequenz)?

Eine Sequenz besteht aus mehreren Anweisungen, die nacheinander abgearbeitet werden. Das ist vergleichbar mit einer Wegbeschreibung: Gehe hundert Meter nach vorne, biege rechts ab, und gehe noch einmal 100 Meter weiter, dann bist zu am Ziel.

Start	
Gehe 1 Schritt vorwärts	
Drehe 90° nach rechts	
Gehe 1 Schritt vorwärts	
End	

Wie wiederhole ich eine Anweisung (Schleife)?

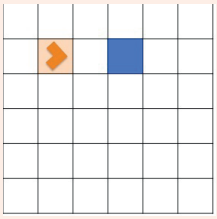
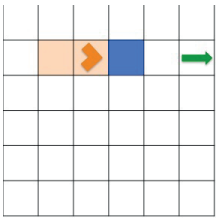
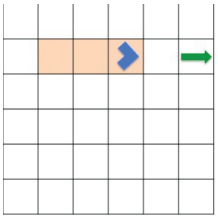
Muss eine Anweisung mehrfach wiederholt werden, so kann man diese in eine sogenannte „Schleife“ packen. Dabei wird innerhalb der Schleife festgelegt, was gemacht werden soll, und wie oft bzw. bis wann diese Sequenz wiederholt werden soll. Z.B. „Gehe geradeaus bis zur nächsten Kreuzung.“

Als Sequenz		**Als Schleife**	
Start		*Start*	
Gehe 1 Schritt vorwärts		Wiederhole 3mal	
Gehe 1 Schritt vorwärts		(Gehe 1 Schritt vorwärts)	
Gehe 1 Schritt vorwärts			
End		*End*	

Auch innerhalb einer Schleife können mehrere Anweisungen vorhanden sein. Z.B. „Gehe 1 Schritt vorwärts“ und „Drehe 90° nach rechts“. Dies würde in der oberen Schleife dreimal hintereinander ausgeführt, d.h. „Gehe 1 Schritt vorwärts, und drehe 90° nach rechts“, „Gehe 1 Schritt vorwärts, und drehe 90° nach rechts“, „Gehe 1 Schritt vorwärts, und drehe 90° nach rechts.“

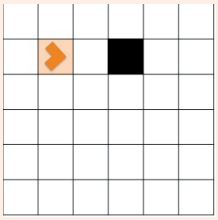
Wie kann ich eine Anweisung an eine Bedingung knüpfen (Selektion)?

Es gibt mit dem „Wenn“-Baustein eine Möglichkeit, um eine Anweisung an eine Bedingung zu knüpfen. Z.B. „Wenn die Ampel rot ist, bleib stehen“. Diese Bedingung wird mit dem englischen Wort „if“ eingeführt und nennt anschliessend die Bedingung.

Start	
Gehe 1 Schritt vorwärts	
If (Kästchen ist blau)	
Wechsle die Farbe zu blau	
End	

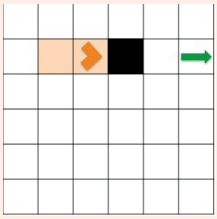
Sie kann erweitert werden zum „Wenn-Ansonsten“-Baustein. Hier wird nun zwischen zwei Optionen entschieden. Z.B. „Wenn die Ampel rot ist, bleibe stehen. Ansonsten gehe weiter.“

Start



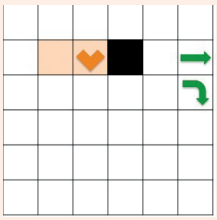
If (Weg ist frei)

Gehe gerade aus



Else

Drehe 90° nach rechts



End

Sowohl den „Wenn-“ wie auch den „Wenn-Ansonsten“-Baustein kann in Schleifen verwendet werden. Diese „Verschachtelungen“ gibt es sehr oft in Programmen.

Hintergrundwissen zur „Ereignissteuerung“

Die Ereignissteuerung, auch „Event-Action-Prinzip“ genannt, hat die Informatik, wie wir sie heute kennen, stark geprägt. Sie funktioniert wie folgt: Es wird ein Ereignis (Event) ausgelöst (z.B. jemand schreibt Ihnen eine Nachricht), die ans Smartphone gelangt. Das Smartphone interpretiert nun dieses Ereignis und entscheidet, an welche App dieses Ereignis sich richtet. Das Betriebssystem des Smartphones startet ganz kurz die App, damit diese entscheiden kann, was mit diesem Ereignis gemacht wird. Eine Nachricht wird also weitergeleitet (Push-Verfahren). Die App zeigt nun (z.B. mit einem kleinen roten Kreis) an, dass eine neue Nachricht eingetroffen ist (Action). Die App verfällt wieder in einen Tiefschlaf und spart Strom. Früher hat man diese Nachricht erst erhalten, wenn die App vom Benutzer gestartet wurde. Erst dann wurde geprüft (Pull), ob eine neue Nachricht angekommen ist.

Bei der Ereignissteuerung wird nach dem Eintreffen eines Ereignisses eine Handlung ausgeführt. Einerseits treten Events als zeitliche Folge (z.B. Erinnerung auf dem Smartphone), andererseits als örtliche Folge (z.B. Geo-Lokalisation im Navigationsgerät) auf. Häufig ist es

ein Mensch, der einen Event einläutet (z.B. Touchgeste auf dem Smartphone). Im Lernfilm tritt dieses Event-Action-Prinzip auf, wenn Wanda in die Nähe der Tramhaltestelle kommt. Dies ist das Event „Wanda ist bei der Haltestelle“. Die App ist nun so programmiert, dass dieser Event eine Aktion auslöst: Die App sucht heraus, wann das nächste Tram fährt und vibriert, sobald dieses eintrifft.

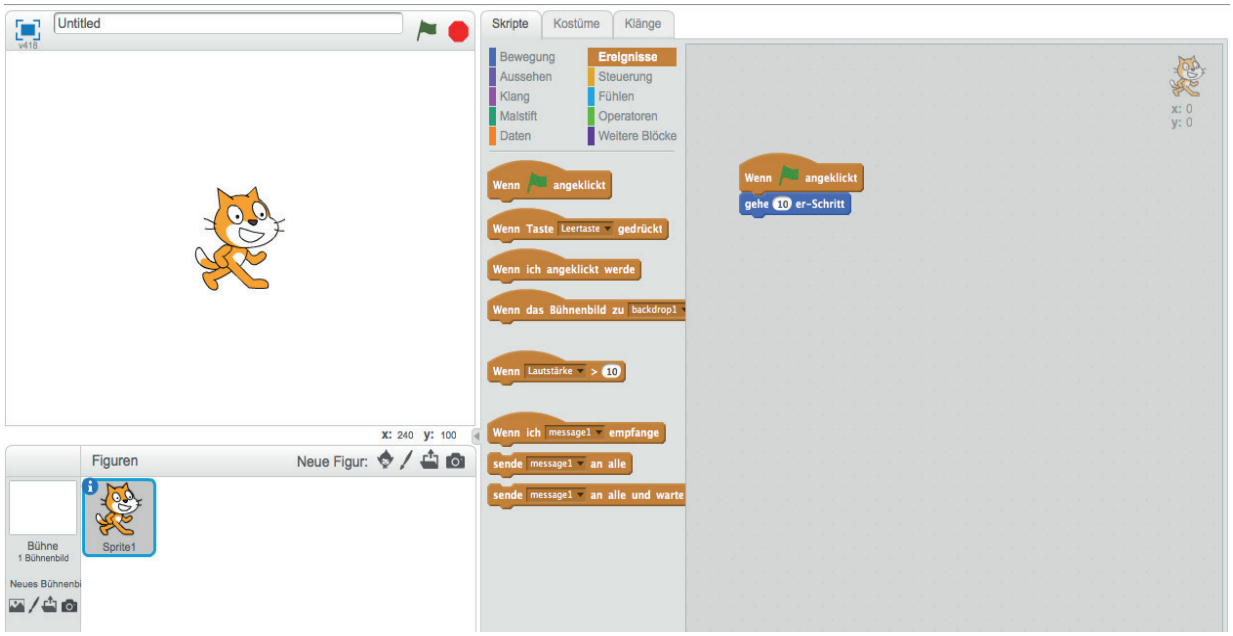
Bei der Ereignissteuerung gibt es prinzipiell zwei Möglichkeiten: Aktiv, d.h. die App erzeugt ein Ereignis. Passiv, d.h. eine App kann auf bestimmte Ereignisse reagieren.

Damit die Ereignissteuerung einwandfrei funktioniert, müssen alle Ereignisse (Events) und darauffolgende Handlungen (Actions) festgelegt sein. In diesem Modul werden folgende drei Varianten der Ereignissteuerung im Programm Scratch angeschaut:

- Ein Ereignis startet eine Handlung (z.B. Wenn die Musik spielt, rennt die Katze.)
- Zwei Programme kommunizieren miteinander: Das eine Programm erzeugt einen Event, das andere Programm reagiert auf diesen Event (z.B. Solange die Katze läuft, steht der Hund.)
- Mensch-Maschine/(Maus)-Interaktion (z.B. Tiere folgen einem Mauszeiger wie einem Magneten.)

Wie löse ich ein Ereignis aus?

Die Programmierumgebung Scratch arbeitet mit visuellen Bausteinen. Dabei gibt es unterschiedliche Bausteine im Bereich „Ereignisse“. Ein solcher Baustein besagt „Wenn XY eintrifft, dann …“ und verlangt eine Anweisung, z.B. „Bewege dich 10 Schritte vorwärts“ oder „Wechsle dein Aussehen“. Die Bausteine lassen sich verschachteln und entsprechen den Grundlagen der Programmierung, wie oben im Hintergrundwissen zum Programmieren erklärt wurde.



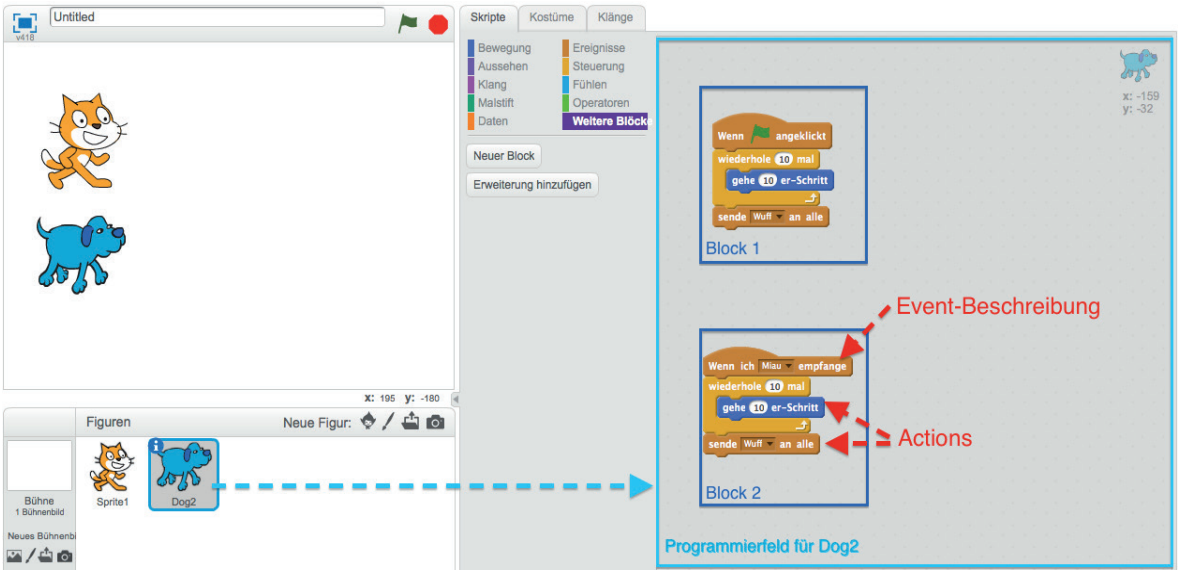
Arbeitsoberfläche Scratch 1: Ein Event führt zu einer Action.

Wie können zwei Spieler miteinander kommunizieren?

Damit zwei Objekte miteinander kommunizieren können, müssen beide etwas austauschen. Dies kann zum Beispiel ein Codewort sein oder ein Standort. Wenn zwei Spieler vorhanden sind, müssen beide einzeln wissen, was sie zu tun haben.
Scratch löst dies so, dass bei mehreren Figuren jede einzeln angeklickt werden kann und somit jede einzelne über ein eigenes Skripte-Areal verfügt. Hier können nun die Bausteine platziert werden, die nur für das jeweilige Objekt bestimmt.

Für die Kommunikation zwischen den beiden Spielfiguren kann ein Codewort bestimmt werden. Z.B. löst ein „Wuff“ des Hundes aus, dass die Bausteine bei der Katze gestartet werden. Ruft die Katze „miau“, werden die Skripte vom Hund aufgerufen.

Es können mehrere Blöcke miteinander auf dem Skripte-Areal, dem Programmierfeld von Scratch, stehen (siehe Bild Arbeitsoberfläche Scratch[2]). Je nach Ereignis wird dann eine andere Aktion aufgerufen. Wird z.B. die grüne Fahne angeklickt, startet der obere Ereignisblock des Hundes. Erst wenn er das Ereignis „Wenn ich [miau] empfange“ erhält, startet beim Hund die zweite Action.



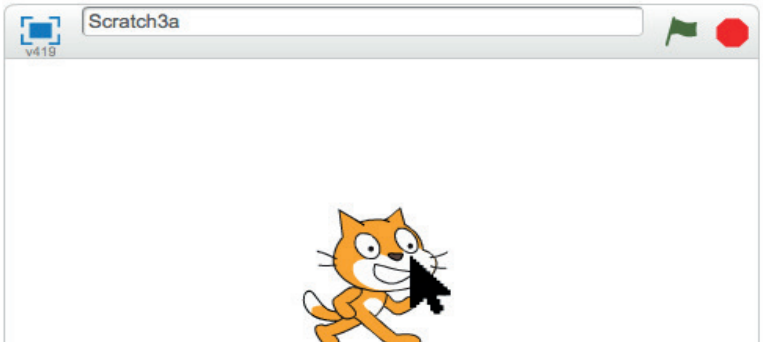
Arbeitsoberfläche Scratch 2: Je nach Event wird die eine oder andere Action aufgerufen.
Beispiel: In der Anweisung „Sende Wuff an alle“ ist „Sende“ die Kommunikation, „Wuff“ ist die Ereignisbeschreibung, und wenn „Wuff“ ausgelöst wird, ist es der Event. Dass, was mit „Wuff“ bei der Katze ausgelöst wird, ist die Action.

Wie programmiere ich eine Interaktion?

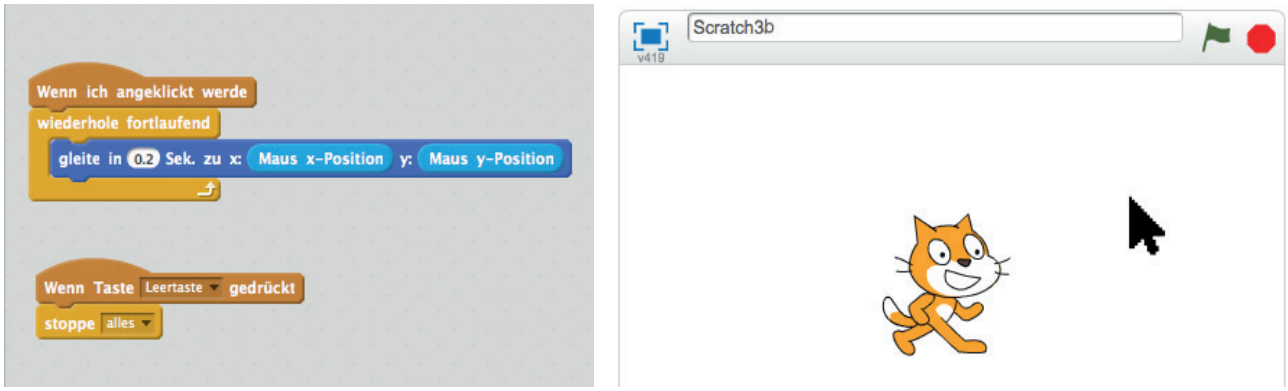
Eine Interaktion beschreibt ein wechselseitiges Aufeinanderwirken von mindestens zwei Akteuren. Es ist zum Beispiel möglich, dass eine Figur dem Mauszeiger folgt. Oft werden Interaktionen in Computerspielen (Games) verwendet, z.B. wenn mit dem Finger eine Spielfigur durch eine Landschaft bewegt wird. Grundsätzlich geht es beim Folgen eines Fin-

[2] Online-Version: http://scratch.mit.edu

gers oder eines Mauszeigers darum, dass das Objekt die Position des Mauszeigers kennen muss. Und zwar fortwährend. Nur so kann es selber die eigene Position anpassen und dem Mauszeiger oder einem Finger folgen.
Dabei kann die Spielfigur direkt (a) unterhalb des Mauszeigers „kleben“ oder aber (b) die Bewegungen des Mauszeigers versetzt nachverfolgen.



Arbeitsoberfläche Scratch 3a: Katze „klebt“ am Mauszeiger.



Arbeitsoberfläche Scratch 3b: Katze bewegt sich zum Mauszeiger hin.

Hintergrundwissen zu „Rund um die App“

Was ist eine App?

App ist die Kurzform von „Applikation“ und meint ein Computerprogramm. Im heutigen Sprachgebrauch sind Apps meist Anwendungen für mobile Endgeräte wie Smartphones und Tablets.

Eine App zu entwickeln, beinhaltet sehr viel Denkarbeit und Fleiss. Eine App-Entwicklung besteht aus vielen einzelnen Schritten:

1. Festlegen der genauen Anforderungen.
2. Programmieren der einzelnen Funktionen.
3. Damit die App bedienbar wird, benötigt sie eine ansprechende und gut bedienbare Benutzeroberfläche.
4. Damit die App auch für Menschen mit körperlicher Beeinträchtigung funktioniert, sollte

man barrierefreie Funktionen[3] beachten.
5. Die App muss mehrfach getestet werden (Testszenarien ausdenken und prüfen).
6. Die App veröffentlichen.

Worauf sollte ich bei einer gut bedienbaren Oberfläche achten?

Damit eine App eine ansprechende Gestaltung hat und zudem einfach zu bedienen ist, muss man sich auch Gedanken zum visuellen Aufbau einer App machen. Mithilfe eines Bausatzes (in der Fachsprache Mockup[4]) wird eine erste Visualisierung der App erstellt. Skizzenhaft wird versucht, die zukünftige App und ihre Funktionen „sichtbar" zu machen. Man macht sich Gedanken über die Grundfunktionen der App (was kann ich mit der App machen?), arbeitet die wichtigsten Navigationselemente heraus (z.B. wie komme ich zurück zum Startbildschirm?), und man erstellt einen ersten Design-Entwurf. Den Schülerinnen und Schülern wird in Experiment 3 eine Möglichkeit gegeben, mithilfe von Bausatz-Elementen einen eigenen Bausatz für eine App umzusetzen.

Was ist Barrierefreiheit?

In Bezug auf die App-Entwicklung für Menschen mit körperlichen Behinderungen sollte auch kurz über die Barrierefreiheit gesprochen werden. *Barrierefreiheit*[5] meint das Berücksichtigen aller Menschen mit und ohne Behinderung oder altersbedingten Einschränkungen (wie z.B. Sehschwächen). Ziel ist es, dass jeder Mensch zu Webseiten, Software, Computern, Geräten, Apps (und im ursprünglichen Sinn auch zu Gebäuden und Bauten) unabhängig einer Behinderung uneingeschränkt Zugang hat. Damit auch körperlich behinderte Menschen Zugang zu Internetseiten und Apps haben, muss beim Erstellen einer solchen darauf geachtet werden, dass Technologien genutzt werden, die keine Barrieren darstellen. Es gibt einige Dinge, die schnell und einfach angepasst werden können und andere, die etwas aufwendiger zu programmieren sind. Hier einige Beispiele (Quelle Wikipedia und www.access-for-all.ch):

- Ein gut strukturierter Text kann von blinden Menschen mit einer Software (Screenreader) gelesen werden. Bilder oder Texte, die in Bildern vorhanden ist, sind für Blinde unzugänglich. Daher sollte jedes Bild einen Alternativtext enthalten, der beschreibt, was sichtbar ist.
- Für Sehschwache sollte die Schriftgrösse angepasst werden können.
- Für gehörlose oder schwerhörige Menschen sollten Videos mit Gebärdensprache oder zumindest Untertitel oder Textfassungen des Videos vorhanden sein.
- Für motorisch behinderte Menschen sollte es möglich sein, die Inhalte nur mit einer Maus oder nur mit der Tastatur ansteuern zu können.

---

3 NZZ-Beitrag zu Barrierefreiheit: http://www.nzz.ch/aktuell/digital/barrierefreiheit-internet-apps-zugang-fuer-alle-1.18299159

4 Weiterführende Software für Mockups-Erstellung sind z.B. Balsamiq Mockups (http://balsamiq.com/products/mockups/) für Computer, iPhone-Mockup (http://iphonemockup.lkmc.ch) für online und Marvel (https://marvelapp.com/iphone/) für iPhone.

5 Apps für Menschen mit Behinderung: http://www.rehadat-hilfsmittel.de/de/infothek/informationen-produktbereiche/apps/index.html#infothek-1-4

Wer macht eine App?

Bei der App-Entwicklung sind viele verschiedene Schritte notwendig: Vom Festlegen der Funktionen und Anforderungen über das Erstellen eines Bausatzes für die gestalterische Umsetzung und das Programmieren der Funktionen bis zum Testen der App durch unterschiedliche Zielpersonen und dem Veröffentlichen der App auf Verkaufsplattformen und im Internet. Dabei sind auch unterschiedliche Berufe gefragt: Seien es Programmierer/innen, welche die App-Funktionen entwickeln oder User Action Designer/innen, welche den Bausatz erstellen und testen lassen oder Grafiker/innen, die sich um Gestaltung und Design kümmern. Zwischen den Berufen gibt es viele Schnittstellen, sodass es gut ist, wenn die eine Person auch ein bisschen etwas über einen anderen Bereich kennt (z.B. die Programmiererin, welche sich auch schon mit User Action Design befasst hat). Die Webseite von „Fit in Informatik in der Bildung“ gibt einen Einblick in verschiedene ICT-Berufe und zur Nachwuchsförderung:

- www.fit-in-it.ch/de/topics/ict-berufe
- www.fit-in-it.ch/de/topics/nachwuchsförderung

Umsetzungshilfen

Reflexion Lernfilm/Perspektiven der Lernenden

Im Plenum wird der Lernfilm „Programmieren – Wie kann Informatik Menschen mit körperlicher Behinderung helfen?“ angeschaut.
Zur Reflexion des Lernfilms können folgende Fragen in der Klasse aufgenommen werden:

- Welche Anforderungen muss die App erfüllen?
- Wie funktioniert die Ereignissteuerung?
- Kennst du Apps, die körperlich behinderten Menschen helfen können?

Diese und ähnliche Fragen werden in den Experimenten thematisiert.

Experiment 1: „Programmieren“

Code.org ist eine Non-Profit-Plattform, die zum Ziel hat, Kinder und Jugendliche in die Welt des Programmierens einzuführen. Das Einstiegstutorial eignet sich hervorragend, um die Grundfunktionen des Programmierens kennenzulernen.

Zeigen Sie zum Einstieg den Film auf der Startseite www.code.org. Unterschiedliche Personen, die mehr oder weniger bekannt sein dürften (Facebook-Gründer Marc Zuckerberg, Microsoft-Gründer Bill Gates u.a.) erklären darin ihr Interesse an der Informatik bzw. am Programmieren. Obwohl die Videos in Englisch sind, werden deutsche Untertitel oder deutsche Textprotokolle angeboten, falls Ihre Schülerinnen und Schüler zum Verstehen Unterstützung brauchen.

Das Experiment 1 enthält eine analoge Variante des Programmierens (von Thinkersmith[6]): Für Trinkbecher, die zu Türmen aufgestellt werden sollen, müssen die Programme geschrieben werden. Während eine Schülerin das Programm schreibt, führt ein anderer Schüler genau ihre Anweisungen aus. Am Ende wird verglichen, ob der Turm dem entspricht, was das Ziel war.

Das Experiment 1 befasst sich mit dem Haupttutorial „Code.org“ und erklärt darin in kleinen Schritten und mit kurzen Einstiegsvideos, wie Anweisungen mit Bausteinen gegeben werden können. Das Tutorial umfasst 20 Übungen und kann ohne Login gelöst und durchgearbeitet werden. Mit Login ist es möglich, den Stand zwischenzuspeichern und zu einem späteren Zeitpunkt weiterzuarbeiten.

Experiment 2: „Ereignissteuerung“

Mit der Online-Programmierumgebung Scratch (scratch.mit.edu) kann das Event-Action-Prinzip aufgezeigt werden. Mit Bausteinen lassen sich so ganze Animationen, Geschichten und Spiele erstellen.
Das Experiment 2 befasst sich mit Scratch und drei unterschiedlichen Event-Action-Prinzipi-

[6] Die komplette Unterrichtslektion kann hier heruntergeladen werden (englisch): http://code.org/api/hour/begin/thinkersmith

en. Ziel ist es, dass die Schülerinnen und Schüler erkennen, welche Funktionen welche Auswirkungen haben und wie die einzelnen Bausteine miteinander kombiniert werden können. Sie sind weniger angeleitet als im Experiment 1, können aber das dort erarbeitete Wissen verwenden, um die Aufgaben zu lösen.

Experiment 3: „Rund um die App“

Um eine eigene App zu planen, müssen als Erstes die Anforderungen und Funktionen festgehalten werden. Dies wird in der ersten Aufgabe des Experimentes gemacht.
Ziel ist es, dass sich die Schülerinnen und Schüler auf „analoge“ Art und Weise mit der App-Erstellung befassen. D.h., sie sollen sich überlegen, wie das Bedienerfeld aussieht und selber mithilfe der Bausatz-Elemente (siehe Kopiervorlage) eine App gestalten. Ein Austausch in den Gruppen soll dazu beitragen, die App zu „testen“. Erkennt man die wichtigen Funktionen? Ging etwas vergessen? Versteht man den Aufbau der App? Hier können wichtige Hinweise fallen, um die Gestaltung der App zu überarbeiten. Bei diesem Experiment gibt es kein Richtig oder Falsch. Hier geht es darum, Ideen zu finden, kreativ zu sein und Begründungen zu haben, weshalb eine Schaltfläche besser links statt rechts steht usw.

Informatik-Biberaufgaben zum Thema Programmieren

Die Biberaufgaben stellen konkrete Fragen zu Themen rund ums Programmieren. Sie wurden den drei Schwierigkeitsstufen einfach, mittel, schwierig zugeordnet und liegen in den entspre-chenden Verzeichnissen.

Weitere Informationen zum Informatik-Biber-Wettbewerb finden Sie auf der Webseite *http://www.informatik-biber.ch/*

Mögliche Vertiefungen

Unterrichtsmaterial zu Scratch:
http://www.swisseduc.ch/informatik/programmiersprachen/scratch_werkstatt/index.html

Es gibt viele Online-Tools mit welchen man Mockups (Bauanleitungen) erstellen kann:
- Online – iPhone Mockup: *http://iphonemockup.lkmc.ch*
- Desktop – Balsamiq: *http://balsamiq.com*
- App und Handskizze – Marvel App: *https://www.marvelapp.com/iphone/*

Als weitere Aufgabe zu Experiment 3 könnte auch eine mobile App umgesetzt werden. Dafür ist diese Online-Umgebung empfehlenswert:
Webseite: *www.adsy.me*

Spiele benötigen oft das Event-Action-Prinzip, somit kann auch die Spiele-Programmierung ein Thema sein, um die Ereignissteuerung zu vertiefen. Hierzu gibt es unterschiedliche Lernplattformen, welche die Spiele-Entwicklung unterstützen:
- Pixel Press Floor (benötigt iPad oder iPhone):
  - Website: *http://projectpixelpress.com/floors-sketch-guide/*
  - Tutorials: *http://projectpixelpress.com/floors-videos/*
- Programmieren mit Android: Jdroid
  - Webseite: *http://www.jdroid.ch*

Barrierefreiheit ist bei der Smartphone-Benutzung ganz wichtig. Selbst die bekannten Firmen entwickeln unterstützende Technologien. Wussten Sie, welche Zusatzfunktionen iPhone & Co. bereits beinhalten und wie diese genutzt werden können?
- Apple – Sonderpädagogik: *https://www.apple.com/chde/education/special-education/ios/*
- Android – Accessibility-Hilfe: *https://support.google.com/accessibility/android/?hl=de#topic=6007234*

Projektwoche in der Schule – „Hack an app in one week“:
- Webseite: *http://www.ti8m.ch/hackanapp.html*
- SRF-Fernsehbeitrag: *http://www.srf.ch/player/tv/10vor10/video/informatikbranche-sucht-nachwuchs?id=9cd4afc4-fac2-4634-baa8-8b64cb4aba26*

Weiterführende Literatur

Sande, Warren D. & Sande, Carter (2009). Hello World! Programmieren für Kids und andere Anfänger, Verlag: Carl Hanser Verlag GmbH & Co. KG , ISBN: 978-3-446-42227-8

Strom, Chris (2014). Kids programmieren 3D-Spiele mit JavaScript,
Verlag: O'Reilly, ISBN: 978-3-955-61496-6

Gressly, Philippe & Guggisberg, Martin (2010). Programmieren lernen: Aufgaben für den Informatikunterricht, Verlag: Orell Füssli Verlag, ISBN: 978-3-280-04066-9

Unterrichtsmaterial für den Informatikunterricht

http://www.swisseduc.ch/informatik/

Unterrichtsmaterial für den Informatikunterricht

http://www.educ.ethz.ch/unt/um/inf

Lösungen

Experiment 1: „Programmieren“

Aufgabe 1

a) Lösungen direkt online
b) Der „move forward“-Baustein bewirkt „Einen Schritt vorwärts gehen“. Der „turn right“-Baustein bewirkt „Drehe 90° nach rechts“. Der „turn left“-Baustein bewirkt „Drehe 90° nach links“.

Aufgabe 2

a) Lösungen direkt online
b) Bei einem „Wiederhole-Anzahl-Baustein“ gibt man an, wie oft etwas wiederholt werden soll. Z.B. wiederhole 3-mal „vorwärts gehen“. Bei einem „Wiederhole-Bis-Baustein“ wird die Anweisung so oft wiederholt, bis etwas passiert, d.h. bis eine festgelegte Bedingung (Condition) zutrifft. Z.B. wiederhole so oft, bis du an die Wand prallst „vorwärts gehen“.

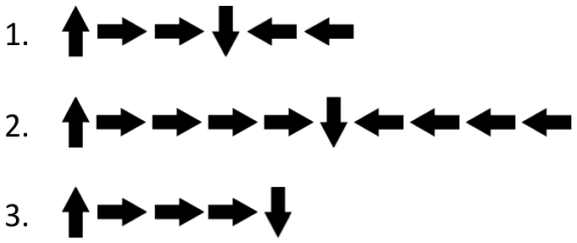
Aufgabe 3

a) Lösungen direkt online
b) Ein „Wenn-Baustein“ wird eingesetzt, um etwas zu prüfen. Die Figur befolgt solange eine Anweisung, bis die Wenn-Bedingung zutrifft. Dann führt die Spielfigur eine andere Anweisung aus. Z.B. „Gehe vorwärts“. Wenn du an eine Kreuzung kommst, gehe links.
c) Man kann mit der Erweiterung „Ansonsten“ die Figur entscheiden lassen, was geschehen soll, falls die Bedingung nicht zutrifft. Z.B. „Gehe vorwärts“. Wenn du an eine Kreuzung kommst, gehe links, ansonsten gehe rechts. Dieser Baustein heisst „Wenn-Ansonsten-Baustein“.

Aufgabe 4

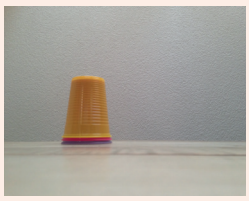
Individuelle Lösungen. Beispiel mit dem 3er-Turm:

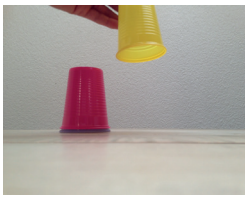
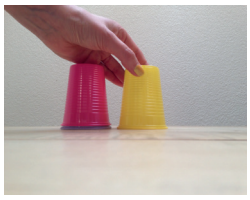
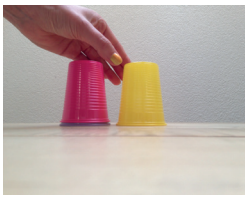
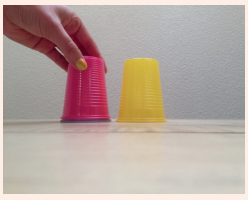
3er-Turm

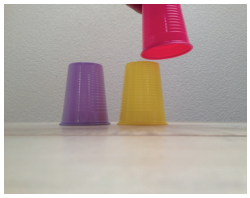
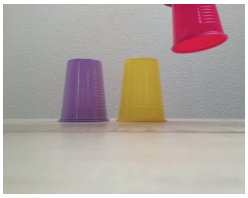


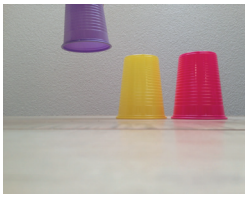
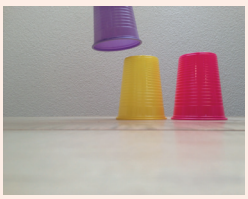
Programm zum 3er-Turm

Ablauf in Fotos:



Experiment 2: „Event und Action mit Bausteinen“

Aufgabe 1
Der richtige Baustein ist „Wenn ich Bimmbimm empfange“.

Aufgabe 2
Die Anweisungen links gehören zur Katze. Die Anweisungen rechts gehören zum Schiff.

Aufgabe 3
Mögliche Lösung:

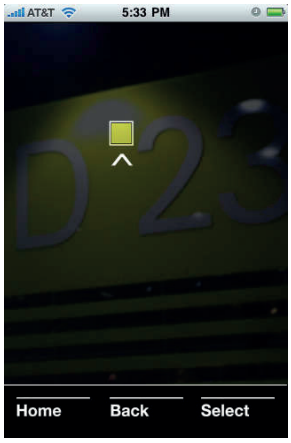
Experiment 3: „Rund um die App“
Aufgabe 1
Mögliche Einsatzzwecke können sein:
- Erkennen der Farbe eines Kleidungsstückes zu Hause oder unterwegs beim Einkaufen.
- Erkennen der Farbe einer Geldnote, z.B. erkennen gelb = 10er-Note, rot = 20er-Note
- Erkennen, wo eine Lichtquelle steht im Raum (z.B. Fenster, Lampe)
- Gegenstand fotografieren und Name des Gegenstandes mit dem Mikrofon aufnehmen. Wird der Gegenstand das nächste Mal mit der App fotografiert, wird die Nachricht vorgesprochen.
- Foto erstellen und gleichzeitig mit Audiospur besprechen. So können die eigenen Fotos wiedergefunden werden, oder mit Internetverbindung wird beim Fotografieren eines Bildes direkt gesagt, worum es sich handelt.
- Lupen-App zum Vergrössern von Texten und Objekten.

Quellen: *http://www.heise.de/ix/artikel/Smarte-Augen-2044548.html* und *http://www.heise.de/ix/artikel/Smarte-Augen-2044548.html*

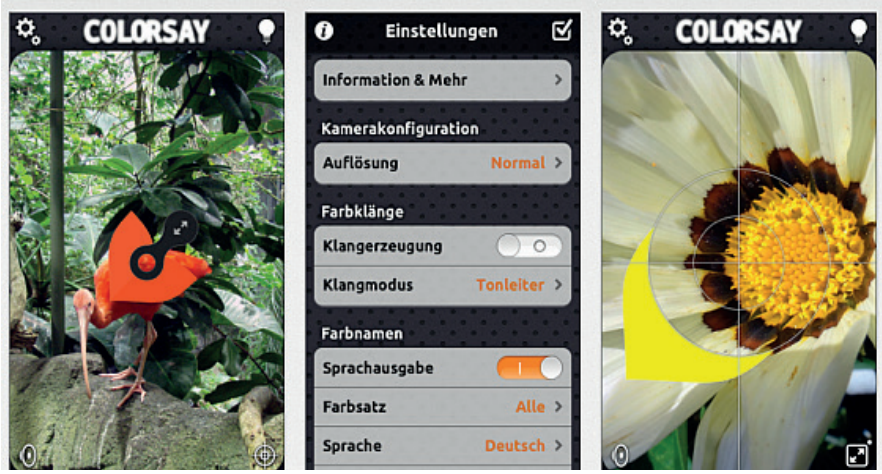
Aufgabe 2
Individuelle Lösungen

Möglicher Vergleich mit bereits vorhandenen Umsetzungen von Apps zur Farberkennung:

1) Color ID (https://itunes.apple.com/us/app/colorid/id317202736?mt=8)



2) ColorSay (https://itunes.apple.com/ch/app/colorsay-hor-die-welt-in-farbe!/id605398028?mt=8)



3) ColorVisor (https://itunes.apple.com/ch/app/colorvisor/id511093568?mt=8)

Aufgabe 3

Individuelle Lösungen

Impressum

Herausgeber	SVIA, Schweizerischer Verein für Informatik in der Ausbildung
Partner	Hasler Stiftung ICT Berufsbildung SWITCH
Konzeption/Umsetzung	Lernetz AG
Autorenschaft	Martin Guggisberg, PH FHNW Yvonne Seiler