Lehramt Informatik (Gy, MS, BS, FS)
Modul „Didaktik der INF - E-Learning und Tools“

Werkzeuge für den Informatikunterricht
---- MIT App Inventor ----

Empfohlen für Klasse: 8-10

Autor: Erik Daas

1. Kurzvorstellung

Der MIT App Inventor von Google ist eine integrierte Entwicklungsumgebung mit einer grafischen Programmiersprache à la Drag and Drop und ist in dem Kontext mit Scratch oder CrypTool vergleichbar. Jedoch ist der App Inventor, wie der Name bereits impliziert, dazu da, kleine Apps für Android zu entwickeln. So ist das Handy ein fester Bestandteil des Entwicklungsprozesses, dessen zahlreichen Funktionen und Sensoren auch explizit angesteuert werden können. Zum Testen kann man auch iOS verwenden und prinzipiell ist das Werkzeug auch für Lego Mindstorms geeignet.

2. Einordnung in die Lehrpläne

Oberschule:

- Klassenstufe 8, LB 2 „Informationen verarbeiten: Modell – Algorithmus – Lösung“ zum Kennenlernen und Nutzen einer Programmierumgebung und grundlegenden Kontrollstrukturen
- Klassenstufe 9, LB 2 „Daten darstellen: Informatikprojekte“, da man hier eine eigene App im Team als Projekt gestalten kann
- Klassenstufe 10, LB 2 „Arbeit in Projekten“, da man auch hier mit diesem Werkzeug eine eigene App selbst alleine als Projekt gestalten kann

Gymnasium:

- Klassenstufe 9/10, LB 4 „Algorithmen und Programme“ zum Kennenlernen und Nutzen einer Programmierumgebung und grundlegenden Kontrollstrukturen
- Klassenstufe 11/12, LB 4 & 5 „Datenstrukturen und Modularisierung“ sowie „Algorithmen“ zum komplexeren Programmieren, auch wenn das Werkzeug dafür vielleicht schon zu stark vereinfacht ist

Berufsschule bzw. Fachoberschule:

- Klassenstufe 12, LB 3 „Projekt“, da man hier eine eigene App als Projekt gestalten kann

Für das Berufliches Gymnasium ist dieses Werkzeug laut Lehrplan ungeeignet bzw. lassen sich keine Lernziele finde, für welche der MIT App Inventor geeignet wäre.

3. Lernziele

Kognitive Lernziele:

Die SuS lernen grundlegende Programmstrukturen kennen.

Die SuS lernen den Algorithmusbegriff kennen.

Die SuS lernen die Grundlagen der Programmierung kennen.

Die SuS lernen die Phasen des Problemlöseprozesses kennen.

Die SuS lernen an Beispielen Grenzen der Algorithmierbarkeit kennen.

Die SuS verwenden die Implementierung ausgewählter Datenstrukturen in einer Entwicklungsumgebung.

Die SuS leiten die Umsetzung des Modells an einfachen Beispielen ab.

Die SuS führen die Programmier-Arbeit mit Unterprogrammen durch.

Psychomotorische Lernziele:

Die SuS wenden die Phasen des Problemlöseprozesses an.

Die SuS wenden den Problemlöseprozess unter Nutzung von Programmstrukturen an.

Affektive Lernziele:

Die SuS entwickeln Prioritäten beim selbstständigen Lösen einfacher Probleme.

Die SuS bewerten die Anwendung des Problemlöseprozesses.

4. Kompetenzentwicklung

Fachkompetenz:

Aufbauend auf dem Algorithmusbegriff beherrschen die Schüler Grundlagen der Programmierung.

Die SuS kennen Möglichkeiten der Algorithmenbeschreibung.

Die SuS kennen einfache und komplexe Algorithmen- und Datenstrukturen.

Die SuS sind in der Lage, einfache Probleme in einer Programmierumgebung zu lösen.

Die SuS können eine Implementierung auf korrekte Funktionalität, auch in Bezug auf Sonderfälle, testen.

Lern-/Methodenkompetenz:

Die SuS können Problemstellungen zerlegen und einem Lösungsprozess zuführen.

Die SuS können Problemlösungen kritisch werten und diese unter verschiedenen Aspekten beurteilen.

Sozialkompetenz:

Die SuS können im Team arbeiten.

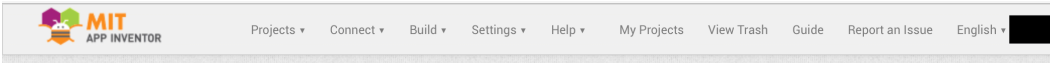
Die SuS können Strategien und Methoden der Problemlösung reflektieren und diskutieren.

Selbstkompetenz:

Die SuS sind besser in der Lage zum strukturierten Denken.

Die SuS besitzen ein größeres Abstraktionsvermögen.

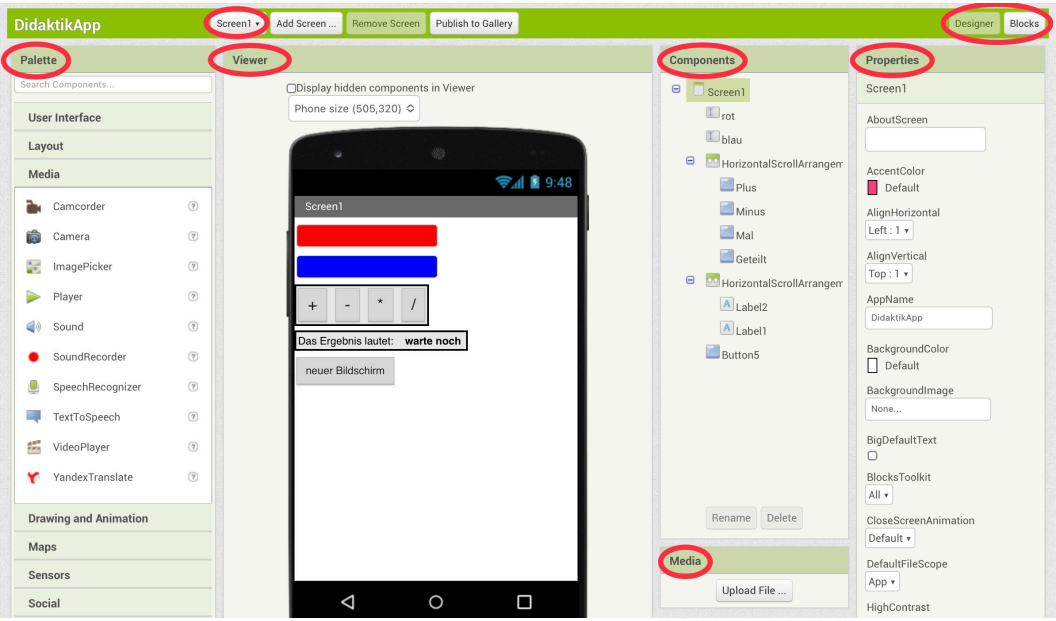
5. Prinzipieller Aufbau



Die online-Entwicklungsumgebung besitzt links in der Kopfzeile eine Auswahl an Funktionalitäten

- **Projects** für die eigenen Projekte, um diese übersichtlich anzusehen und zu verwalten,
- **Connect** für die Verbindung mit dem Testgerät, auf welchem bspw. die App AI Companion installiert ist,
- **Build** für die Erstellung von Android-Installationsdateien, um die App ohne den Companion zu nutzen, und
- **Help** für Problembehandlungen.

Auf der rechten Seite der Kopfzeile, steht seine für die Werkzeug-Nutzung notwendige Google-Konto-**E-Mail-Adresse**, eine **Sprachauswahl** sowie ein **Guide** mit weiteren Erklärungen und Beispielen zum Lernen und Verwenden des Programmierwerkzeugs.



Bei einem offenen Projekt steht der Projektname in der Kopfzeile des

nun geöffneten *Design Editor*. Direkt darunter ist die **Palette** voller funktionaler Platzhalter, die von Knöpfen über Sensoren bis hin zu Datenbanken reichen, welche man auf sein simuliertes Handy oder Tablet im **Viewer** rechts daneben ziehen kann. Es wird immer angezeigt, welcher **Screen** gerade editiert wird und diesen kann man in der Design-Editor-Kopfzeile ändern, falls man mehrere braucht.

Die Komponenten im **Viewer** lassen sich beliebig anordnen und gruppieren. Rechts daneben befindet sich eine Übersicht aller Objekte unter dem Reiter **Components**. Dort kann man diese umbenennen, löschen oder deren Eigenschaften ändern, welche wiederum rechts daneben unter **Properties** gelistet sind. Möchte man eigene Bilder oder Tonaufnahmen verwenden, muss man diese erst unter dem Reiter **Media** hochladen, welcher sich unter den Components befindet.

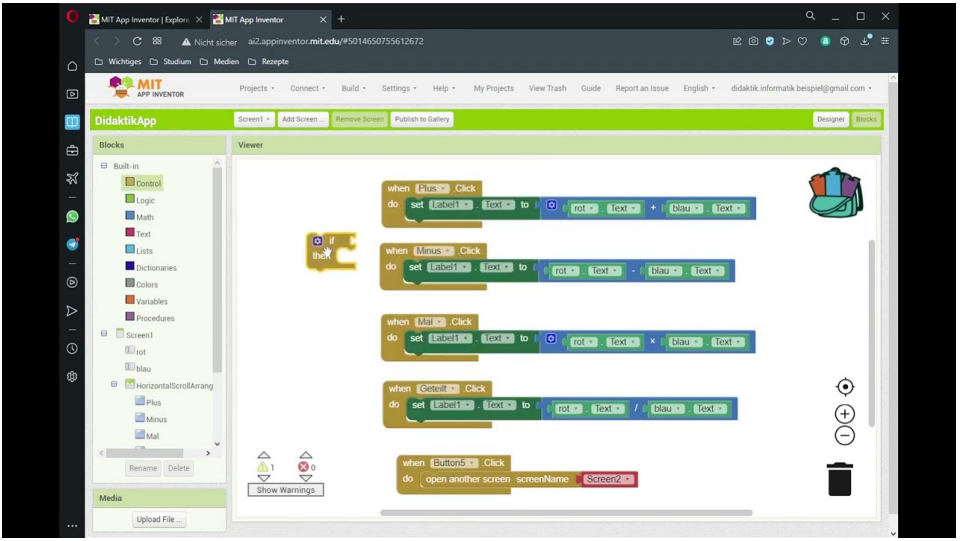
Um die erstellten Objekte zu verwenden, klickt man oben rechts auf **Blocks** und befindet sich nun im *Blocks Editor*. Für die Rückkehr zum *Design Editor* wird oben rechts **Designer** betätigt.

Im *Blocks Editor* kann man per Drag and Drop die Programmiereinheiten von links unter **Blocks** in den Viewer ziehen. Diese Programmiereinheiten sind in verschiedene, allgemeine Gruppen gegliedert, als auch in Komponenten-spezifische. Setzt man diese im Viewer zusammen, kann es trotz Steckkasten-Logik zu Fehlern in der Programmierung kommen, welche unten links bei **Show Warnings** angezeigt werden.

Möchte man einen Block löschen, zieht man diesen auf die **Mülltonne** unten rechts. Will man diesen auf dem jetzigen oder auf einen anderen **Screen** kopieren, zieht man ihn auf den **Rucksack** oben rechts, der als Zwischenablage fungiert.

6. Handhabung

- Schnellstart durch folgendes Video:



- https://appinventor.mit.edu öffnen
- „Create Apps“ betätigen und sich mit einem Google-Konto anmelden
- „Start new project“ für ein neues Projekt
- Gestaltung des Designs erfolgt per Drag and Drop über die Palette, dessen Anordnung man selbst bestimmen kann
- Änderung von Attributen der auf das simulierte Handy gezogenen Objekte erfolgt in den Properties
- zum Nutzen von eigenen Grafiken muss diese erst unter Media hochgeladen werden
- Auswahl von verschiedenen gestaltbaren/programmierbaren Bildschirmen erfolgt über dem Viewer bei „Screen 1“
- oben rechts kann man zwischen Design und Blocks Editor wechseln, um je nachdem die zu nutzenden Objekte auf dem Bildschirm anzuordnen (Designer) oder diese zu programmieren (Blocks)
- Programmierung erfolgt per Drag and Drop über die Blocks, welche in allgemeine und Objekt-spezifische Blocks unterteilt sind
- zu löschende Blöcke zieht man auf die Mülltonne
- zum Kopieren von Blöcken, explizit auch auf andere Screens, zieht man diese in oder aus dem Rucksack
- die App kann jederzeit über den Companion des Smartphones ausprobiert werden, indem man auf Connect —> AI Companion klickt und mit der dazugehörigen App „MIT AI2 Companion“ den abgebildeten QR-Code scannt
- über „Build“ kann man eine Installationsdatei (.apk) erstellen, welche man über einen mit der Handykamera scanbaren QR-Code auf seinem Smartphone herunterladen und installieren kann
- die Projekte werden automatisch gespeichert und bleiben auf Lebensdauer des Google-Kontos auch erhalten

7. Beispielaufgaben für SuS

1. Design

- Erstelle zwei **Textfelder**, ändere deren Hintergrundfarbe und benenne sie nach diesen Farben um.
- Benutze eine **Anordnung** und erstelle da drin 4 **Buttons**; jeder von diesen soll mindestens Schriftgröße 20 haben und sowohl der Text als auch der Name der Buttons wird auf die vier Grundrechenarten geändert.
- Arrangiere zwei **Label** nebeneinander, wobei der Text des rechten fett dargestellt wird und in dem des Lenken steht: „Ergebnis: “. Benenne diese Label passend zu ihren Aufgaben um.

2. Taschenrechner

- Programmiere mit der von dir erstellten Oberfläche aus Aufgabe 1 einen Taschenrechner, der die eingegebenen Zahlen in den Textfeldern auf Button-druck so verrechnet wie es der Button angibt. Das Ergebnis wird im fett-druckenden Label dargestellt.
- Teste das Programm mit der Companion-App auf deinem Handy. Gib dafür Zahlen in die Textfelder ein, drücke einen Button, dessen Rechenoperation ausgeübt werden soll, und überprüfe das Ergebnis im Label.

3. Export als .apk

- Gehe oben im Browser auf „Build“ —> „Android App (.apk)“.
- Scanne den QR-Code mit der Handy-Kamera.
- Installiere die heruntergeladene Datei, indem du sie ausführst, und ignoriere dabei jegliche Sicherheitswarnungen deines Endgeräts.
- Starte die App und lass sie deinen Sitznachbarn testen.

4. Zusatz: Bällebad

- Erstelle einen weiteren **Button** und einen weiteren **Screen** und programmiere den Button dazu, den neuen Screen zu öffnen.
- Erstelle auf dem neuen Screen einen weiteren **Button**, der den jetzigen Screen schließen kann.
- Erstelle ein **Canvas**, das die ganze Bildschirmfläche einnimmt.

- Füge einen **Ball** hinzu und ändere Farbe, Name und Größe beliebig.
- Programmiere den Ball so, dass…

  … wenn er berührt wird, entweder loswollt oder zum Stehen kommt.

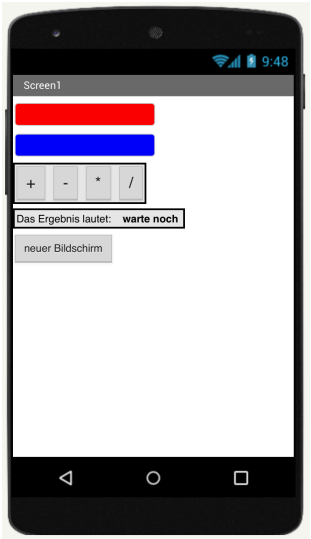
  … wenn er losrollt, zufällig in eine Richtung von 0° bis 360° schaut.

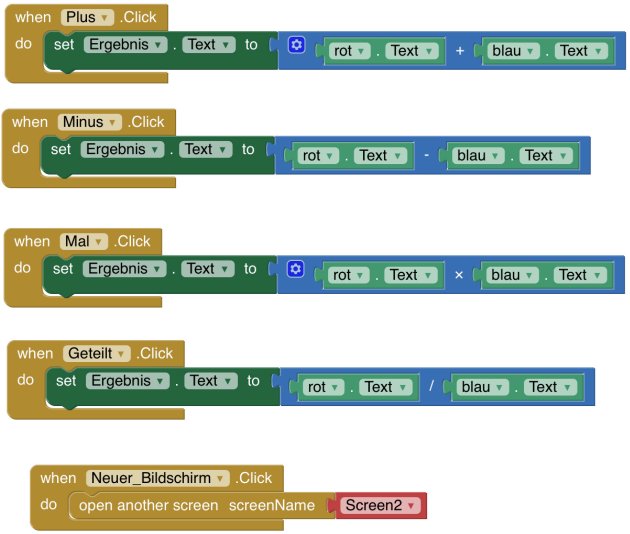
  … wenn er (hier Ball1 genannt) die Wand berührt, in diese Richtung zeigt:

- Füge noch einen **Ball** hinzu, kopiere den Code nochmal für diesen ball und ändere alle Bezüge zum alten Ball passend auf den neuen.

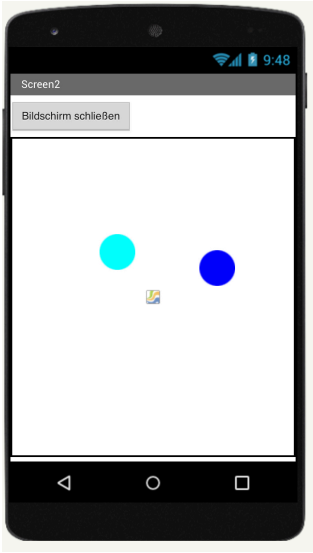
8. Musterlösung

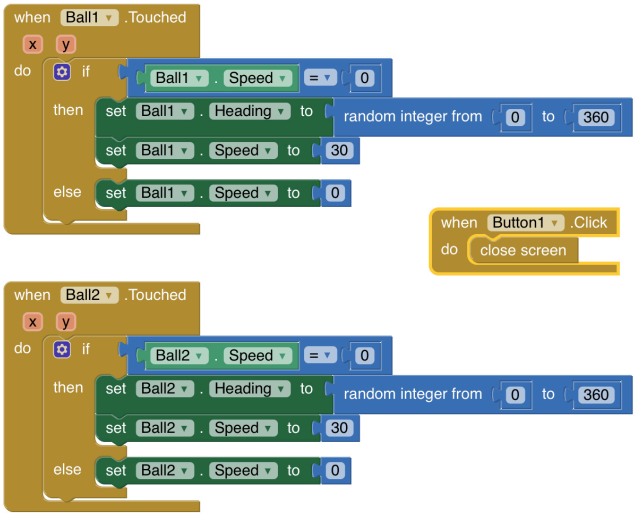
Aufgabe 1 + 2:

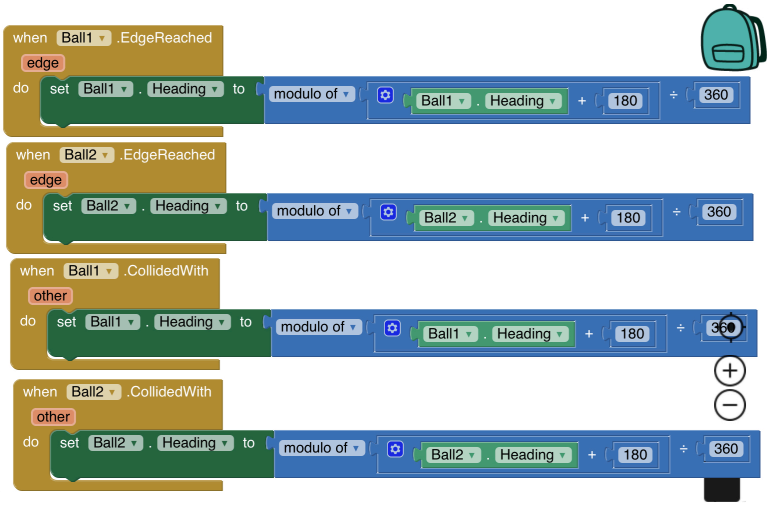




Aufgabe 4:


9:48
Screen2
Bildschirm schließen


when Ball1 .Touched
x y
do if Ball1 . Speed = 0
then set Ball1 . Heading to random integer from 0 to 360
set Ball1 . Speed to 30
else set Ball1 . Speed to 0
when Button1 .Click
do close screen
when Ball2 .Touched
x y
do if Ball2 . Speed = 0
then set Ball2 . Heading to random integer from 0 to 360
set Ball2 . Speed to 30
else set Ball2 . Speed to 0


when Ball1 .EdgeReached
edge
do set Ball1 . Heading to modulo of Ball1 . Heading + 180 ÷ 360
when Ball2 .EdgeReached
edge
do set Ball2 . Heading to modulo of Ball2 . Heading + 180 ÷ 360
when Ball1 .CollidedWith
other
do set Ball1 . Heading to modulo of Ball1 . Heading + 180 ÷ 360
when Ball2 .CollidedWith
other
do set Ball2 . Heading to modulo of Ball2 . Heading + 180 ÷ 360