Werkzeuge für den Informatikunterricht

---- MIT App Inventor ----

Empfohlen ab Klasse 10

Autor: Saskia Dübener

1. Kurzvorstellung

Der MIT App Inventor ist ein Online-Tool zur Erstellung von Apps für Android-Systeme. Er ist sehr gut dafür geeignet, mit Schülerinnen und Schülern Projekte zu erschaffen und umzusetzen und bietet somit viel Potenzial für einen starken Lebensweltbezug für die Lernenden im Unterricht. Die Programmierung findet per Drag and Drop mit einer grafischen Programmiersprache, ähnlich zu Scratch, statt und bietet viele verschiedene Möglichkeiten zur Gestaltung der Apps.

2. Einordnung in die Lehrpläne

	Klassen-stufe	Lernbe-reich	Bemerkung
OS	8	2	Einsetzbar für die Umsetzung von gemeinsamen oder eigenen Lösungen beim Problemlösen Beschränkung auf sehr einfache Probleme bzw. Anwendungen empfohlen, deshalb sind Anwendungen wie Scratch, Robot Karol o. ä. in dieser Klassenstufe geeigneter
	9	2	Durchführung eines Projektes, bei dem das Produkt eine einfache App ist.
	10	2	Selbstständige Arbeit an einem Projekt unter Nutzung des MIT App Inventors; Rückbezug zu Erfahrungen aus Klasse 9
Gym	9/10	4	Einsetzbar bei der Anwendung der Phasen des Problemlösens Beschränkung auf sehr einfache Probleme bzw. Anwendungen empfohlen, deshalb sind Anwendungen wie Scratch, Robot Karol o. ä. in dieser Klassenstufe geeigneter

	Klassen-stufe	Lernbe-reich	Bemerkung
	11/12	4, 5 Optional: 8 C	Einsetzbar zum Üben der Implementierung und Nutzung von Algorithmen, Datenstrukturen, Unterprogrammen und Parametern In Lernbereich 8C kann der MIT App Inventor für die Anwendung der Programmierprinzipien beim Bearbeiten einer komplexen Problemstellung genutzt werden, aber in diesem Lernbereich werden objektorientierte Programmiersprachen betrachtet, weshalb ein Einsatz dieser sinnvoll wäre
BS (FOS)	12	3	Die Produkte der vorgesehenen Projektarbeiten können Apps sein und die Lernenden könnten die Inhalte und Funktionen je nach eigenen Interessen und Können gestalten
BGY	-	-	Keine Eignung für das Berufliche Gymnasium, da im MIT App Inventor weder eine objektorientierte noch eine höhere Programmiersprache integriert ist, welche im Lehrplan jedoch explizit gefordert werden

Abkürzungen:

OS… Oberschule

Gym… Gymnasium

BS … Berufsschule

FOS … Fachoberschule

BGY … Berufliches Gymnasium

3. Lernziele

Kognitive Lernziele:

Die Schülerinnen und Schüler implementieren einfache Datentypen, algorithmische Grundstrukturen und ausgewählte Datenstrukturen im MIT App Inventor.

Die Schülerinnen und Schüler wenden Mittel und Methoden der Informatik bei der Durchführung eines Projektes mit dem MIT App Inventor an.

Die Schülerinnen und Schüler gestalten Projekte zur Lösung von fachrichtungsbezogenen komplexen Problemstellungen, wobei das dabei entstehende Produkt eine App ist.

Die Schülerinnen und Schüler wenden die Phasen von Projektarbeiten bei der eigenen Durchführung einer Projektarbeit an.

Psychomotorische Lernziele:

Die Schülerinnen und Schüler modellieren den Aufbau einer App zur Lösung eines gegebenen Problems und stellen diese in geeigneter Form (z. B. in einem UML Diagramm) dar.

Die Schülerinnen und Schüler erstellen und zeichnen Struktogramme zu gegebenen Problemstellungen.

Affektive Lernziele:

Die Schülerinnen und Schüler erkennen den Nutzen der einzelnen Phasen von Projektarbeiten für den Erfolg eines Projektes.

Die Schülerinnen und Schüler können ihre Arbeitsergebnisse und ihren Arbeitsprozess kritisch reflektieren und gegebenenfalls Schlussfolgerungen für folgende Projekte ziehen.

4. Kompetenzentwicklung

Fachkompetenz:

Die Schülerinnen und Schüler können Programmierstrukturen wie Sequenz, Verzweigung und Schleife verwenden sowie im MIT App Inventor implementieren.

Die Schülerinnen und Schüler können zu einem gegebenen Problem eine geeignete Lösung modellieren und implementieren.

Die Schülerinnen und Schüler sind in der Lage einfache Datentypen, algorithmische Grundstrukturen und ausgewählte Datenstrukturen im MIT App Inventor zu implementieren.

Die Schülerinnen und Schüler sind in der Lage, Projekte zur Lösung von fachrichtungsbezogenen komplexen Problemstellungen zu gestalten.

Lern-/Methodenkompetenz:

Die Schülerinnen und Schüler können mit Unterprogrammen und Parametern beim Programmieren arbeiten.

Die Schülerinnen und Schüler können Projekte planen und umsetzen, wobei sie die kennengelernten Phasen von Projektarbeiten anwenden.

Die Schülerinnen und Schüler können Algorithmen zur Lösung von Problemen entwerfen und diese in Struktogrammen darstellen.

Sozialkompetenz:

Die Schülerinnen und Schüler sind in der Lage mit anderen gemeinsam an einem Projekt zu arbeiten und gemeinsame Absprachen zu treffen.

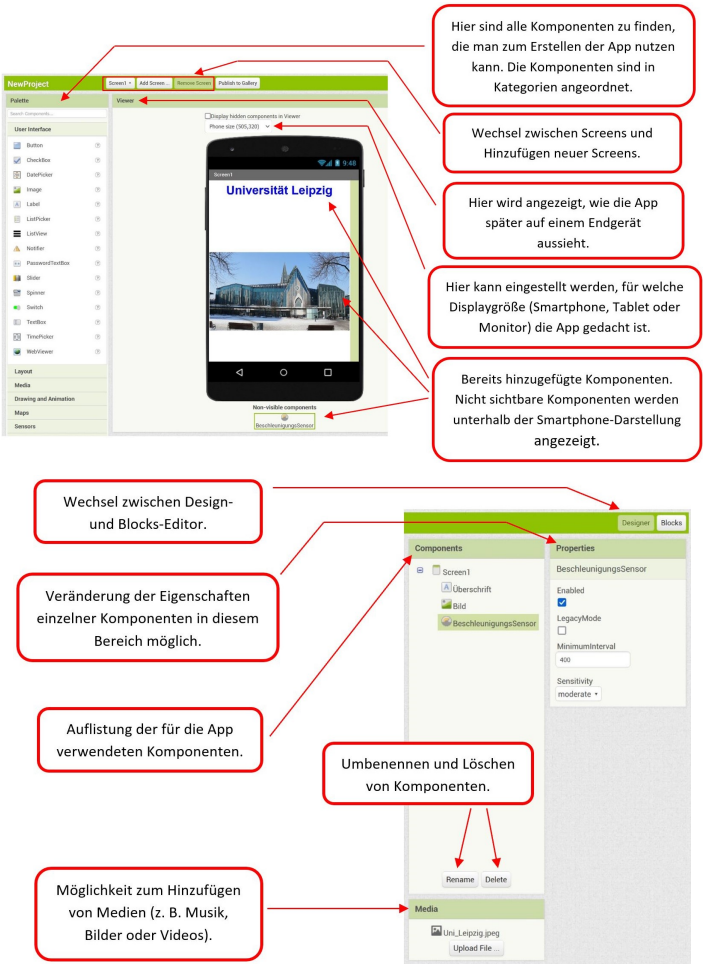
Selbstkompetenz:

Die Schülerinnen und Schüler können ihre Arbeitsergebnisse und ihren Arbeitsprozess reflektieren und Rückschlüsse für zukünftige Projektarbeiten ziehen.

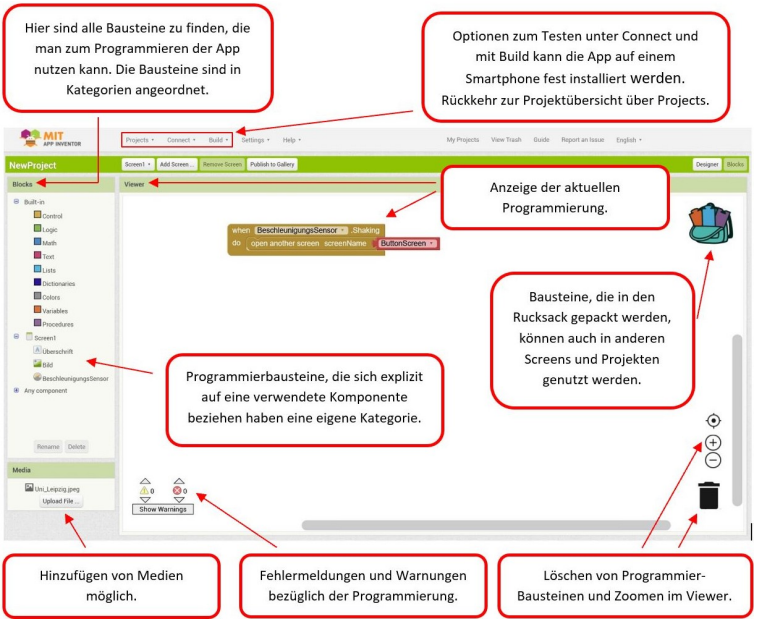
5. Prinzipieller Aufbau

Der MIT App Inventor beginnt mit einer Übersicht der eigenen Projekte. Hierbei ist es möglich, ein neues Projekt zu starten oder Projekte in den Papierkorb zu schieben. Erst wenn die Projekte aus dem Papierkorb entfernt werden sind sie wirklich gelöscht.

Durch Doppelklicken auf ein Projekt öffnet sich die Entwicklungsumgebung des Projektes, welche aus zwei Editoren besteht. Zunächst öffnet sich der **Design-Editor** der App. Im Desgin-Editor wird das Aussehen der App bearbeitet, das heißt hier wird festgelegt, welche Komponenten auf welchem Screen vorhanden sein sollen und wie diese auszusehen haben. Der Design-Editor selbst unterteilt sich in die fünf Bereiche *Palette*, *Viewer*, *Components*, *Media* und *Properties*.



Der **Blocks-Editor** besteht aus den drei Bereichen *Blocks*, *Viewer* und *Media*. Im Blocks-Editor können den einzelnen Screens Funktionen hinzugefügt werden, das heißt in diesem Editor wird die Funktionalität der App bestimmt.



6. Handhabung

Der MIT App Inventor wird in einem Browser genutzt. Dabei werden Chrome, Firefox und Safari unterstützt. Für die Nutzung des App Inventors ist eine WLAN Verbindung und ein Google Account notwendig.

Link zum Tool: https://appinventor.mit.edu

1. Ein Projekt starten

- Auf *Create App* drücken
- Mit eigenem Google Account einloggen
- Über *Start new project* ein neues Projekt starten oder durch Doppelklick ein bereits existierendes Projekt öffnen

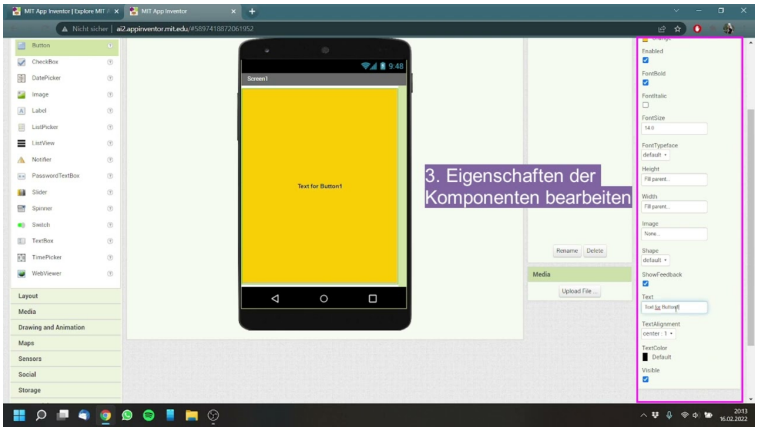
Eine App besteht aus sogenannten Screens, die alle über ein eigenes Design und eine eigene Funktionalität verfügen können. Das Aussehen der einzelnen Screens wird im Design-Editor bestimmt und im Blocks-Editor werden Funktionen und Verbindungen zwischen den einzelnen Screens hergestellt.

2. Aussehen der einzelnen Screens im Design-Editor gestalten

- Gewünschte Komponenten der App aus der *Palette* per Drag and Drop dem *Viewer* hinzufügen (ggf. Komponenten über die Suchleiste in der *Palette* suchen)
- Einzelne Komponenten im Bereich *Components* sinntragend umbenennen
- Eigenschaften und Aussehen der Komponenten im Bereich *Properties* anpassen
- Ggf. Medien wie Audio- oder Video-Dateien im Bereich *Media* hinzufügen
- Weitere Screens über *Add Screen…* erstellen und über den Button neben *Add Screen…* zwischen den Screens wechseln

Hinweis: Das sinnvolle Benennen von Komponenten und Screens ist sehr wichtig, um bei steigender Komplexität der App nicht den Überblick zu verlieren

Wie das Designen einer App beispielsweise aussehen kann, ist im nachfolgenden Video nachverfolgt zu sehen.



3. Die Funktionalität der Screens im Blocks-Editor hinzufügen

- Über die grüne Navigationsleiste in den Blocks-Editor wechseln
- Programmierbausteine per Drag and Drop aus dem *Blocks*-Bereich in den *Viewer* ziehen
- Programmierbausteine sinnvoll zusammensetzen und auf Fehlermitteilungen oder Warnungen achten
- Sich auf verschiedenen Screens wiederholende Programmierelemente in den Rucksack ziehen und bei Bedarf wieder herausholen

Hinweis: Programmier-Bausteine, die explizit eine Komponente betreffen besitzen eine eigene Kategorie im Blocks-Bereich.

4. Die App testen

Sowohl während als auch am Ende der Entwicklung ist es wichtig, die eigene App zu testen, um zu sehen wie einzelne Komponenten aussehen und wie fortgeschritten die Programmierung ist. Für das Testen der App gibt es drei verschiedene Möglichkeiten:

A. Nutzung der AI2 Companion App und WLAN

- AI2 Companion App auf ein Android-Gerät installieren
- Im App Inventor QR Code zum Verbinden über *Connect* und *AI Companion* öffnen
- Android Gerät per QR Code oder Eingabe des Zugangscodes verbinden
- App testen

Hinweis: Der Computer und das Android-Gerät müssen im selben WLAN angemeldet sein

Wie das Programmieren im Blocks-Editor und das Verbinden mit der MIT AI2 Companion App funktioniert ist im folgenden Video zu sehen.

B. Nutzung der AI2 Companion App und USB

- AI2 Companion App auf ein Android-Gerät installieren
- App Inventor-Software auf dem Computer installieren (Link: http://explore.appinventor.mit.edu/ai2/setup-device-usb) und öffnen
- Android-Gerät per USB-Kabel mit dem Computer verbinden
- Im App Inventor über *Connect* und *USB* Verbindung herstellen
- App testen

Hinweis: Falls es nicht funktioniert, könnte das Problem sein, dass USB-Debugging unter Einstellungen und Entwickleroptionen nicht eingeschaltet ist.

C. Nutzung des Emulators (Simulation)

- App Inventor-Software (aiStarters) auf dem Computer installieren (Link: http://explore.appinventor.mit.edu/ai2/setup-device-usb) und öffnen
- Im App Inventor Simulation über *Connect* und *Emulator* starten (Simulation findet im Programm aiStarters statt)

5. Die App exportieren

- Über *Build* Dateien generieren lassen, um die App fest auf einem Endgerät zu installieren.

7. Vor- und Nachteile des Tools

Vorteile	Nachteile
Projektarbeiten sind möglich, wobei ein Endprodukt entsteht, welches die Lernenden eventuell auch privat oder in der Schule nutzen können.	Das Erstellen und Programmieren der Apps kann sehr zeitaufwendig sein.
Es gibt viele Möglichkeiten, die Apps zu gestalten (Nutzung von Sensoren und Aktoren des Endgerätes bis hin zur Einbindung von Datenbanken). Dadurch ist eine Binnendifferenzierung möglich.	Die Benutzeroberfläche und vielen (Auswahl-)Möglichkeiten und englischen Bezeichnungen können für kleinere Jahrgänge erst einmal überfordernd sein.
Schülerinnen und Schüler nutzen den App Inventor eventuell auch außerhalb der Schule.	Das Designen kann vom eigentlichen Üben des Programmierens ablenken.
Lebensweltbezug im Unterricht ist möglich.	Die Nutzung ist eigentlich nur mit Android-Endgeräten bequem.

8. Erstellung einer Fitness-App mit dem MIT App Inventor (Beispielaufgaben mit Lösung)

Ziel: Entwicklung einer App, welche verschiedene Sportübungen anzeigt, wobei man über Pfeiltasten zwischen den einzelnen Übungen wechseln kann. Zusätzlich kann programmiert werden, dass wenn man auf das Bild einer Übung tippt, automatisch ein Video, auf dem die Übung noch einmal richtig ausgeführt wird, beginnt. So kann genau nachvollzogen werden, wie die Sportübung auszusehen hat.

Aufgabe 1: Startbildschirm erstellen

Hinweis: Diese Aufgabe sollte nicht länger als 10 Minuten dauern!

1.1 Überschrift hinzufügen

Füge dem *Screen 1* die Komponente **Label** aus dem Bereich *User Interface* der *Palette* per Drag and Drop hinzu. Benenne die Komponente in *Components* in „Überschrift“ um.

Bearbeite die Eigenschaften deiner Überschrift im Bereich *Properties*, sodass die Überschrift nun fett und in einer größeren Schriftgröße geschrieben ist und trage bei „Text“ den Inhalt deiner Überschrift ein. Optional kann auch noch die Schriftfarbe geändert werden.

1.2 Einleitungstext erstellen

Erstelle nun einen Einleitungstext, der dem Nutzer erklärt, was in der App passieren soll. Nutze hierfür wieder die Komponente **Label**.

*Hinweis: Denk daran, das **Label** wieder sinnvoll umzubenennen. Zeilenumbrüche im Text können durch
 erzeugt werden.*

1.3 Ausrichten der Texte

Die Überschrift sollte über dem Einleitungstext stehen. Ist das noch nicht der Fall, ziehe die Überschrift im *Viewer* per Drag and Drop über den Einleitungstext.

Drücke im Bereich *Components* auf „Screen 1“, um jetzt in den *Properties* des Screens die *Alignments (AlignHorizontal / AlignVertical)* auf „Center“ zu stellen. Damit müssten nun die Texte in der Mitte des Bildschirms angezeigt werden.

Aufgabe 2: Die Übungen anzeigen lassen

Als nächstes geht es an die Erstellung des Bildschirms, auf dem die Sportübungen angezeigt werden sollen.

2.1 Screen erstellen und Bildmaterial hochladen

Erstelle einen neuen Screen mit **Add Screen…** und bezeichne ihn mit „Übung“.

Um die einzelnen Übungen anzeigen, müssen die passenden Bildmaterialien im Bereich *Media* hochgeladen werden.

Suche dir vier Bilder mit Sportübungen aus dem Materialienordner aus, welchen du unter https://transferxl.com/download/08v2nHSgh-L7HmG herunterladen kannst, und lade sie im App Inventor hoch. Lade zusätzlich auch die beiden Pfeilbilder aus dem Ordner in den Bereich *Media* hoch.

2.2 Übung anzeigen

Erstelle einen **Button**, indem du die Komponente aus dem Bereich *User Interface* der *Palette* in den *Viewer* ziehst.

Bearbeite die Eigenschaften des **Buttons** in den *Properties* so, dass eine der ausgewählten Übungen in der Größe 300x300 Pixel auf dem **Button** angezeigt wird.

Füge abschließend eine Überschrift mit dem Namen der Übung hinzu. Gehe dabei wie in Aufgabe 1.1 vor.

Aufgabe 3: Vom Startbildschirm zur Übung wechseln

Es soll nun programmiert werden, dass wenn der Startbildschirm angezeigt wird, durch Schütteln des Endgerätes der Bildschirm gewechselt wird.

Füge dazu die Komponente **Accelerometer-Sensor** aus der Kategorie *Sensors* dem Startbildschirm hinzu.

Wechsle in den **Blocks-Editor** und nutze die dir zur Verfügung stehenden Elemente von **Accelerometer-Sensor** und *Control*, um beim Schütteln des Handys zum zweiten Screen zu wechseln.

Aufgabe 4: Zwischen den Übungen wechseln

4.1 Buttons zum Übungswechsel erstellen

Gehe zurück in den **Design-Editor**.

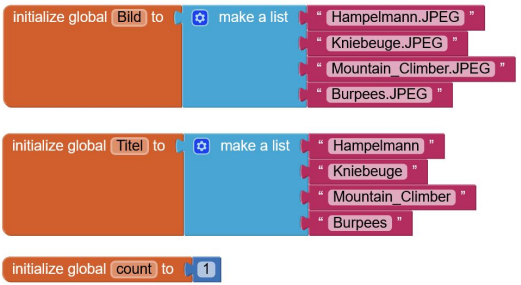
Füge deinem *Übungs-Screen* die Komponente **HorizontalArrangement** aus dem Bereich *Layout* in der *Palette* hinzu. Stelle in den *Properties* die Höhe und Breite auf „Fill Parent“.

Erstelle nun zwei neue **Buttons**, welche du per Drag and Drop in das **HorrizontalArrangement** ziehst. Benutze den Bereich *Properties*, um auf dem linken Button den Pfeil nach links und den rechten Button den Pfeil nach rechts aus den zuvor hochgeladenen Bildern anzeigen zu lassen *(Breite: 30%, Höhe: Fill Parent)*. Benenne den linken Button in „Rückwärts“ und den rechten Button in „Vorwärts“ um.

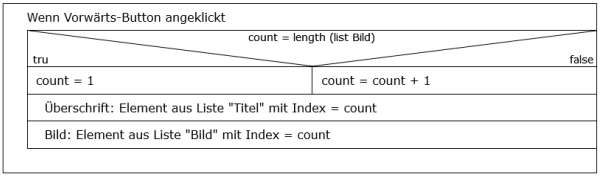
4.2 Funktionalität der Buttons programmieren

Wechsle in den **Blocks-Editor** und übernimm in deinen *Viewer* den folgenden Programmteil. Anstelle der hier verwendeten Übungen und Dateinamen müssen die von dir ausgewählten Übungen und Dateinamen eingetragen werden

Hinweis: Die passenden Programmier-Bausteine findest du in Variables, Lists, Math und Text. Um die Größe der Liste zu verändern muss man auf das blaue Rädchen im blauen Baustein drücken.



Damit stehen nun zwei Listen zur Verfügung, in denen die Titel der Übungen bzw. die Namen der Bilddateien gespeichert sind, und eine globale Zähl-Variable.



Setze nun das Struktogramm von der vorhergehenden Seite in deinem Blocks-Editor um.

Erstelle nun auch einen Programmblock für den Rückwärts-Button.

Teste deine App

Alle weiteren Aufgaben sind Zusatzaufgaben für schnelle und interessierte Schülerinnen und Schüler!

Aufgabe 5: Hinzufügen der Videos

5.1 Einen neuen Screen erstellen und Videos hochladen

Erstelle einen neuen Screen namens „Video“ und füge im *Media*-Bereich die passenden Videos zu deinen ausgewählten Übungen aus dem Materialienordner hinzu.

Füge im **Design-Editor** einen *Video-Player* zum Screen hinzu und übergib ihm in den *Properties* als „Source“ das passende Video zu deiner ersten Übung.

5.2 Liste der Videos erstellen

Erstelle im **Blocks-Editor** des Screens, auf dem die Übungen angezeigt werden, eine Kopie der Liste mit den Namen der Bilddateien. Ändere die Bezeichnung der Variable und die einzelnen Listenelemente so, dass nun anstelle der Namen der Bilddateien die Namen der jeweiligen Videodateien stehen, das heißt die Endungen „.jpeg“ müssen mit „.mp4“ ersetzt werden.

Kopiere nun diese neue Liste in den *Rucksack*, indem du den Programmteil in den *Rucksack* ziehst. Wechsle nun zum Screen „Video“ und ziehe den Programmteil wieder aus dem *Rucksack* heraus.

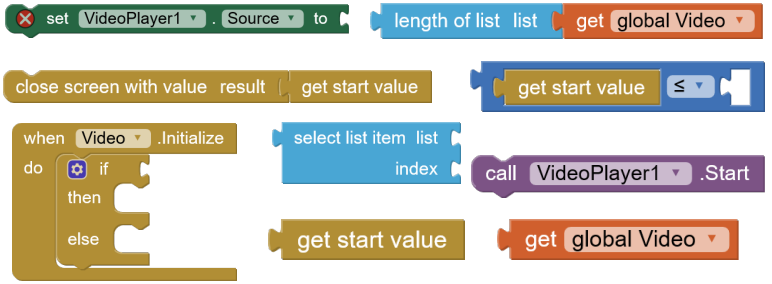
5.3 Funktionalität hinzufügen

Gehe zurück in den **Blocks-Editor** des Übungs-Screens. Erstelle selbstständig einen Programmteil, der beim Klicken auf das Bild den Screen „Video“ öffnet und als Startwert die Variable „count“ übergibt.

Wechsle zur Programmierung des Video-Screens.

Setze die folgenden Programmierblöcke so zusammen, dass eine Prozedur entsteht, welche beim Initialisieren des Video-Screens das richtige Video aus der Liste mit den Namen der Videos heraussucht und das jeweilige Video im Player startet.

Hinweis: Die Blöcke sind in den Kategorien Control, Math, Lists, Variables, Video und VideoPlayer zu finden. Jeder Programmblock wird nur genau einmal verwendet!



Erstelle eigenständig einen Programmteil, der nach dem Beenden des Videos den Screen wieder schließt

Hinweis: Die benötigten Blöcke sind in den Kategorien Video und Control zu finden.

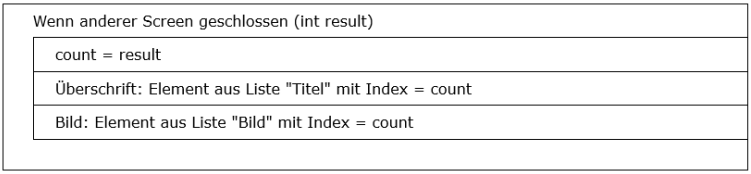
Teste deine App

Aufgabe 6: Der letzte Schliff

Beim Testen deiner App ist dir vielleicht aufgefallen, dass man nach dem Schließen des Video-Screens nicht wieder zu der ursprünglichen Übung zurückkommt, sondern wieder von vorne startet. Um bei der Rückkehr wieder bei der ursprünglichen Übung weiter machen zu können, muss beim Schließen des Video-Screens der Startwert übergeben werden.

Wandle deine Prozedur nach dem Beenden des Videos im Video-Screen wie folgt ab:

Wechsle zur Programmierung des Übungs-Screens und setze dort folgendes Struktogramm um:



Hinweis: Bewege deine Maus über die rot hinterlegten Teile des Programm-Bausteins aus der Kategorie Control, um nötige Funktionen zu erhalten.

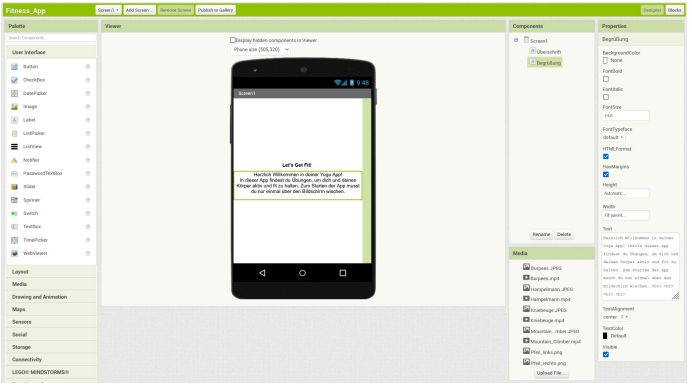
Aufgabe 7: Erweitere deine App

Jetzt hast du die Möglichkeit, deine App individuell zu erweitern. Falls du keine eigene Idee hast, was du noch implementieren möchtest, sind hier noch ein paar Ideen aufgelistet:

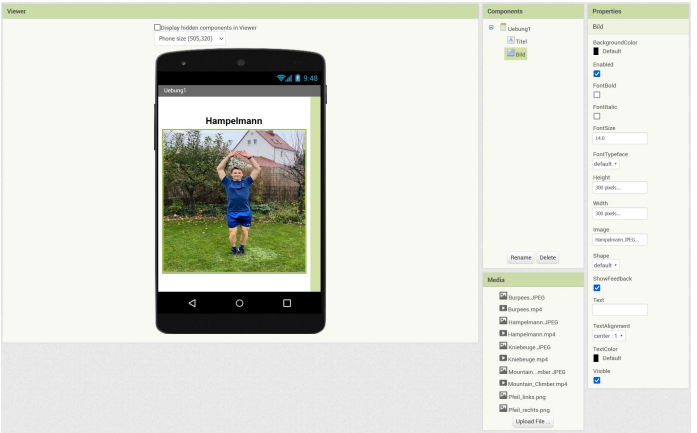
- Sich wiederholende Elemente können zu einer Prozedur zusammengefasst werden
- Füge für die einzelnen Bilder die passenden Bildgrößen hinzu, sodass beim Bildwechsel, die Größe des Bild-Buttons angepasst wird, um keine verzerrten Übungsbilder zu erhalten
- Audios aufnehmen, die Erklärungen zu den einzelnen Übungen und der richtigen Ausführung beinhalten, und diese in die App integrieren
- Automatisches Wechseln der Übungen nach einer bestimmten Zeit
- Zufällige Auswahl der nächsten Übung
- Einen Timer-Button einfügen, der sobald er ausgelöst wird eine bestimmte Anzahl an Sekunden herunterzählt und am Ende einen Ton abspielt, damit man weiß, dass man die Übung nun wechseln kann

Lösungen:

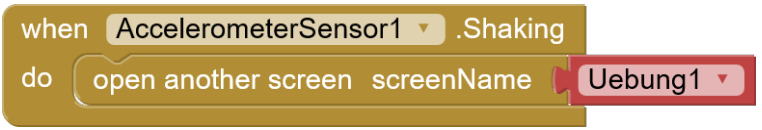
Aufgabe 1:

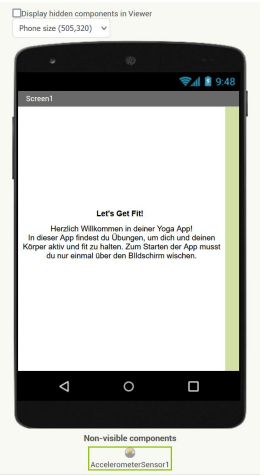


Aufgabe 2:

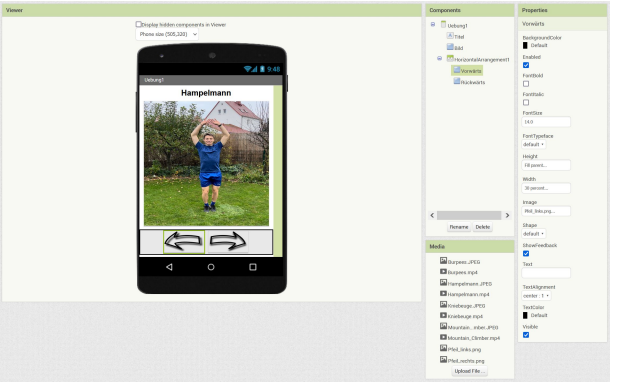


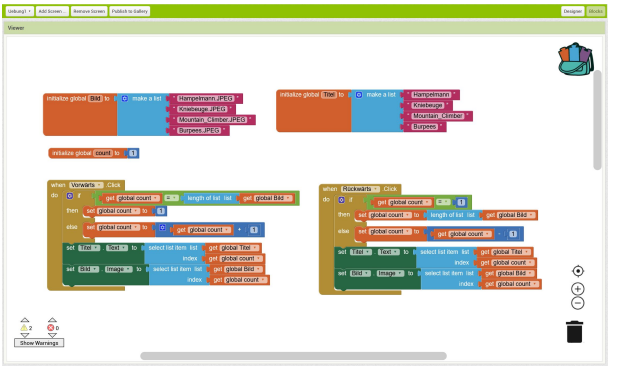
Aufgabe 3:



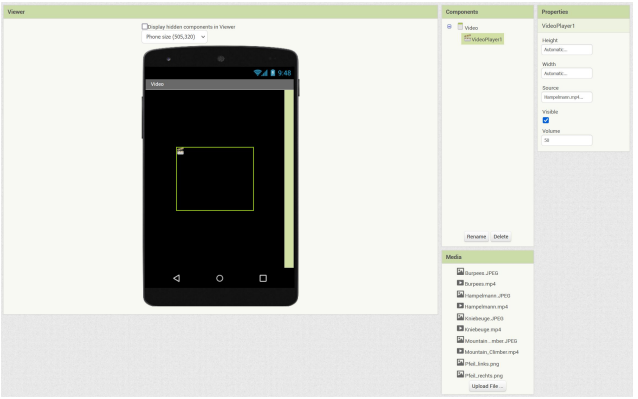

Display hidden components in Viewer
Phone size (505,320)
9:48
Screen1
Let's Get Fit!
Herzlich Willkommen in deiner Yoga App!
In dieser App findest du Übungen, um dich und deinen Körper aktiv und fit zu halten. Zum Starten der App musst du nur einmal über den Bildschirm wischen.
Non-visible components
AccelerometerSensor1

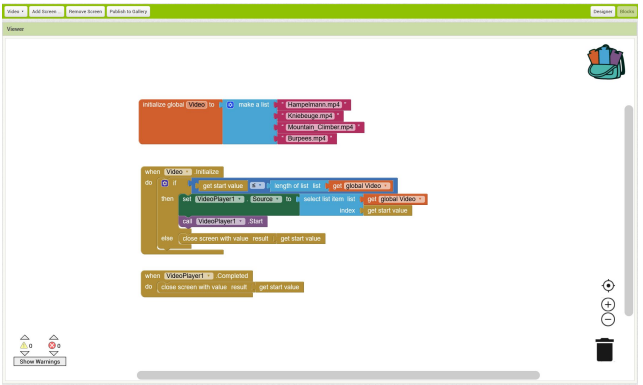
Aufgabe 4:


Viewer
Components
Properties
Hampelmann
Media


Viewer
Show Warnings

Aufgabe 5:





Aufgabe 6: