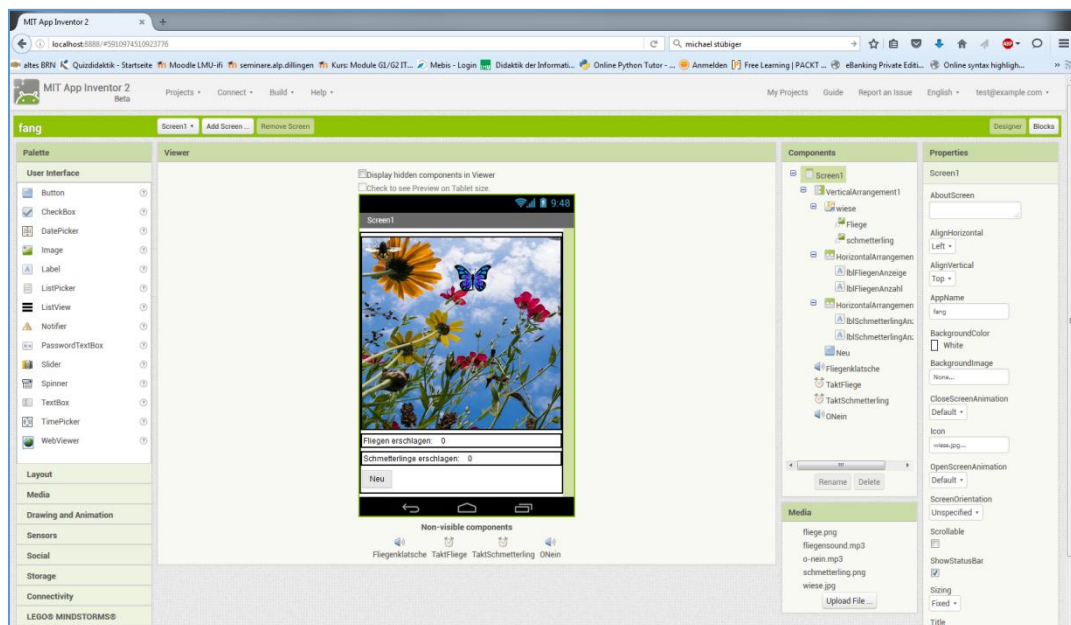




# App-Programmierung mit dem MIT App-Inventor 2



Erstellt von Anja Rosenbaum



## Einführung

Android Apps programmieren die Profis mit Java oder einer anderen Programmiersprache. Es ist jedoch auch möglich, einfache Apps mit dem MIT App-Inventor zu erstellen. Die Oberfläche ist vergleichbar mit z. B. der Software „Scratch“. Dabei werden die einzelnen benötigten Blöcke ineinander geschoben und angepasst.

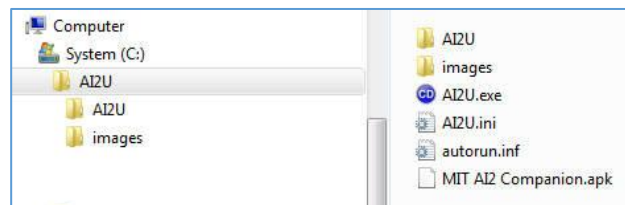
Der App-Inventor kann online unter <http://appinventor.mit.edu/explore/> verwendet werden, dazu wird jedoch ein Google-Account benötigt. (Bei oben genannten Link klickt man auf „Create Apps“, gibt anschließend seinen Account ein und gelangt somit auf die Online-Oberfläche. Möchte man den Emulator (AIStarter) benutzen, so muss dieser extra installiert werden.

Ist kein Google-Account vorhanden bzw. möchte diesen nicht nutzen, so gibt es mit z. B. AI2U die Möglichkeit, den App-Inventor 2 offline und sogar auch als portable-Version zu nutzen. Damit spart man sich dann auch die Installation des Emulators. AI2U, ausgesprochen „App Inventor 2 Ultimate“, wurde von Krupong entwickelt. Erhältlich ist er unter <https://sourceforge.net/projects/ai2u/>. Im Bereich „Files“ findet man die einzelnen Versionen. Oben stehen die Neuesten, zur Zeit ist das ai2u 3.6, Stand Juni 2016 (Hinweis: Dieses Skript wurde noch unter der Version 2.8 erstellt.) Zum Download der Portable-Version wählt man nach einem Klick auf die Versionsnummer den "Portable-Ordner". Je nach Betriebssystem lädt man sich die 32bit bzw. die 64bit Version herunter. Das Paket „AI2Starter.zip“ benötigen wir nicht.

### Wichtig:

Die entpackten Dateien müssen sich direkt im Windowsverzeichnis (i. A. C:\) befinden, der übergeordnete Ordner muss „AI2U“ heißen.

**C:\AI2U**


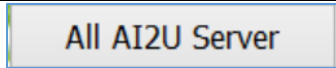

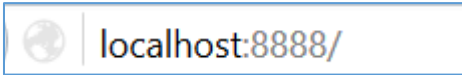
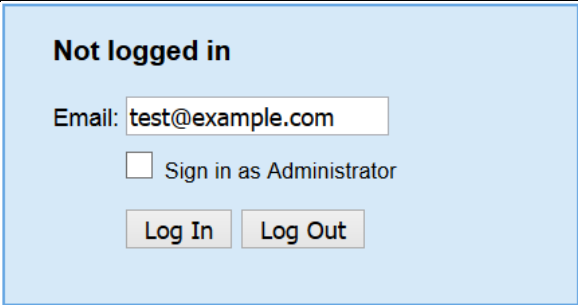
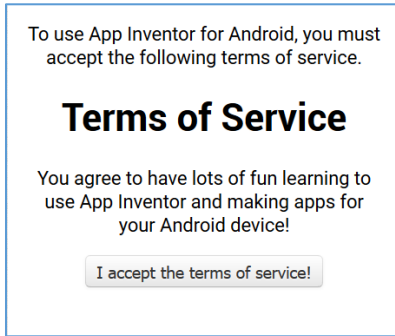


## Start des App-Inventors:

Start des App-Inventor mit Doppelklick auf AI2U.

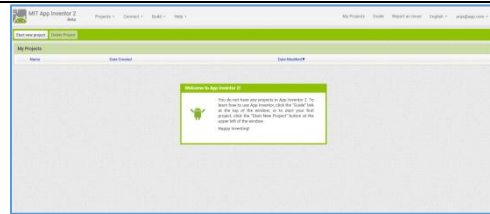




Es erscheint nebenstehendes Popup.	
Klick auf „All AI2U Server“	
Im unteren Desktopbereich erscheinen diverse Programme, <b>nicht schließen</b> - einfach ignorieren.	
Öffnen eines Browsers, z. B. „Mozilla firefox“. Navigieren nach <b>localhost:8888</b> (Achtung: Unter dem Internet Explorer läuft der AI2U nicht)	
Akzeptieren aller eventuellen Meldungen der Firewall bzw. Lizenzvereinbarungen.	
Im folgenden Bild kann eine beliebige (nicht dringend existente) E-Mail-Adresse eingegeben werden. Diese dient als Account, alle erstellten Projekte werden darunter gespeichert.	
Klick auf „Log In“ und anschließend auf „I accept the terms of service“.	



Starten eines neuen Projekts unter der Oberfläche des App-Inventors 2 mit Hilfe von „Start new project“.



## Die Oberfläche des Designers

Im Designer wird das Aussehen der App gestaltet.

### Der Viewer:

Hier siehst du, wie in etwa deine App später aussehen wird.

### Die Eigenschaften:

Hier kannst du die Attributwerte zu den zugehörigen Attributen deiner App ablesen.

### Blocks:

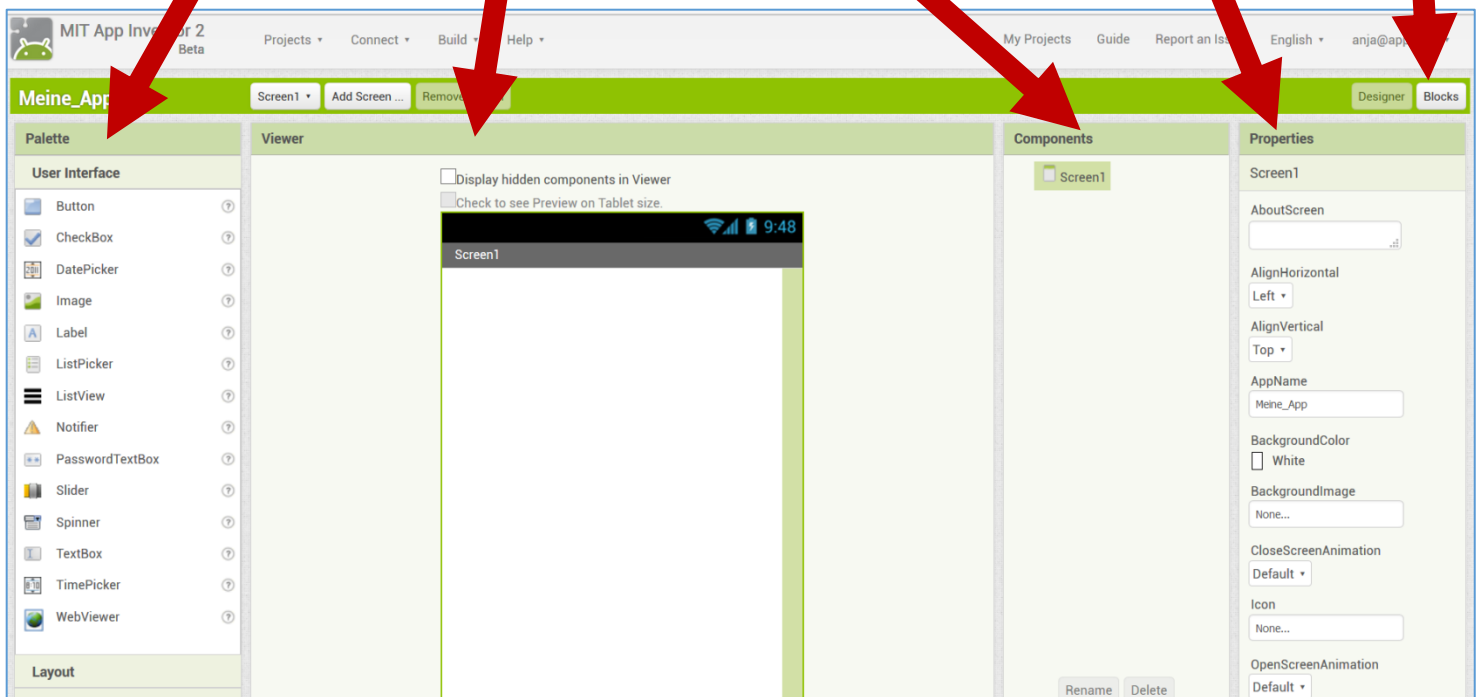
Hier wechselst du in den Programmierbereich deiner App.

### Die Palette:

Wähle hier die gewünschte Komponente und ziehe sie in den „Viewer“.

### Die Komponenten:

Hier findest du deine gewählten Komponenten wieder. Mit Rename bzw. Delete kannst du sie auch umbenennen bzw. löschen.





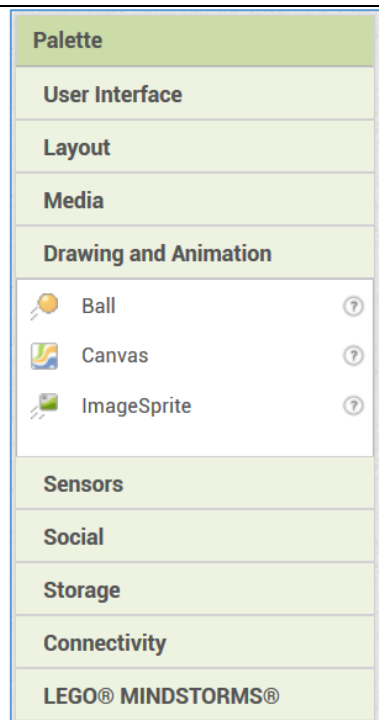
## Die App „Fliegenfang“

In der App "Fliegenfang" sollen vor dem Hintergrund einer Blumenwiese zufällig Fliegen und Schmetterlinge erscheinen. Aufgabe ist es, die Fliegen zu fangen, Schmetterlinge jedoch nicht zu berühren.

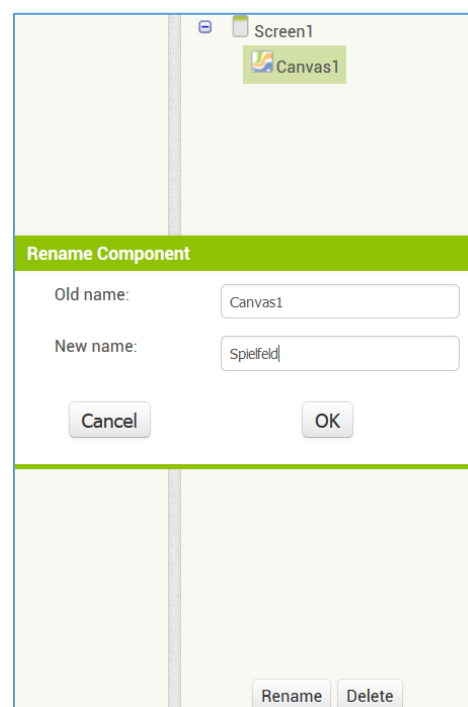
### Erstes Design

In der Palette befindet sich der Bereich „Drawing und Animation“. Da sich die Fliege bzw. der Schmetterling bewegt, handelt es sich um eine Animation. Somit werden Komponenten aus diesem Bereich benötigt.

Den Hintergrund einer Animation bildet immer eine „Canvas“ (deutsch: Leinwand). Die „Canvas“ wird in den Viewer gezogen, es erscheint dort ein Objekt namens "Canvas1".



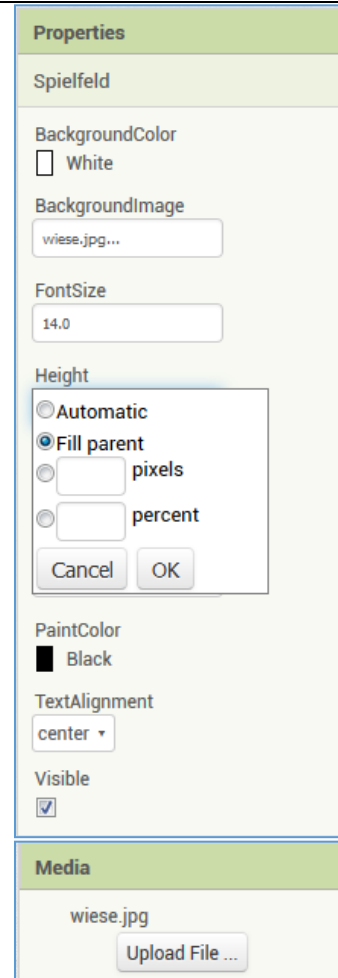
Unter „Components“ kann die "Canvas1" mit Hilfe von „Rename“ umbenannt werden.



Attributwerte der Leinwand, wie z. B. Höhe, Breite oder auch das Hintergrundbild können unter "Properties" verändert werden.

"Fill parent" bei Höhe bzw. Breite bedeutet, dass die Größe der übergeordneten Komponente - hier wäre das das eigentliche Smartphone - angepasst wird.

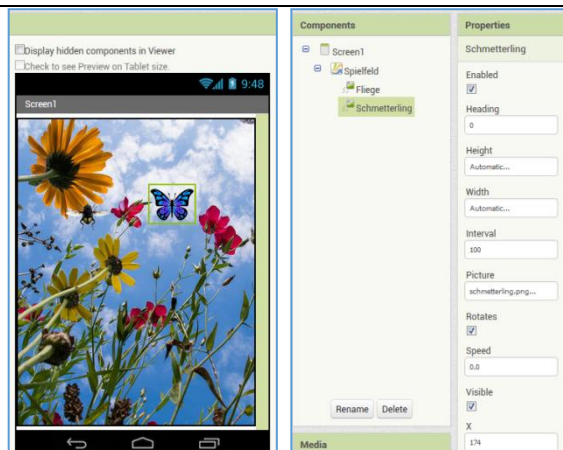
Für den Hintergrund kann eine Bilddatei hochgeladen werden. Diese befindet sich dann im Bereich "Media"



Die Fliege bzw. der Schmetterling bilden "ImageSprites".

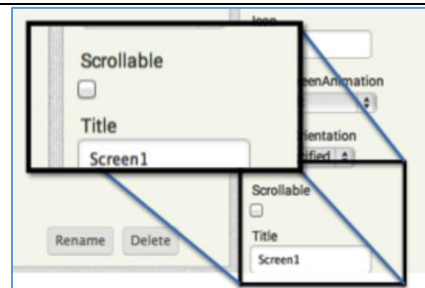
Es werden also zwei solche in den Viewer gezogen und deren Attributwerte ggf. angepasst.

Die beiden Sprites erscheinen im Objektbaum der "Components"



**Achtung:** Es ist darauf zu achten, dass der Bildschirm (hier Screen1) nicht Scrollable ist.

Anstatt "Screen1" kann der Titel natürlich geändert werden, z. B. in "Fliegen fangen".

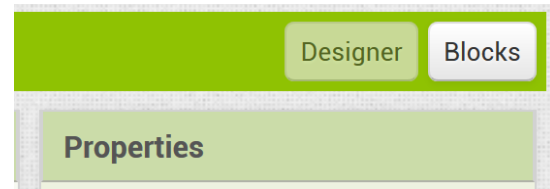




## Erstes Programmieren

Im ersten Schritt sollen der Schmetterling bzw. die Fliege nach Berührung an einem zufälligen anderen Ort der Leinwand erscheinen.

Die Programmierung findet im Bereich "Blocks" statt.



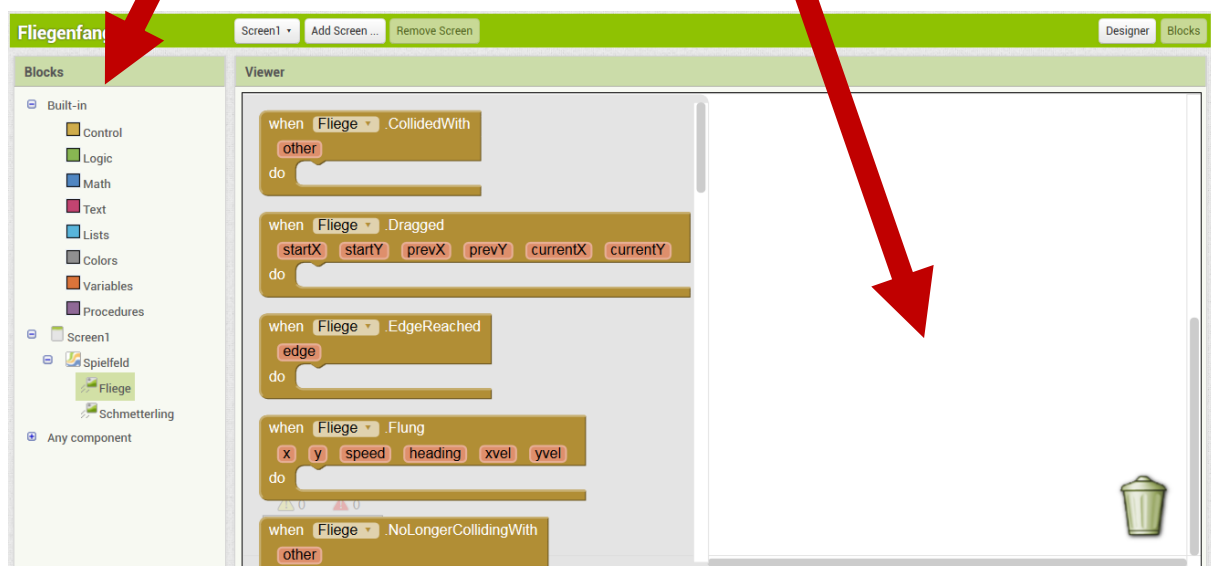
### Die Oberfläche von „Blocks“

#### Die Blöcke:

Nach Klick auf eine Komponente erscheinen im Viewer zugehörige Programmierblöcke, die verwendet werden können.

#### Der Viewer

Hier befindet sich das selbst zusammengestellte Programm. Mit CTRL C bzw. CTRL V können schon programmierte Teile kopiert bzw. eingefügt werden. Gelöscht werden sie, indem sie in den Papierkorb gezogen werden.

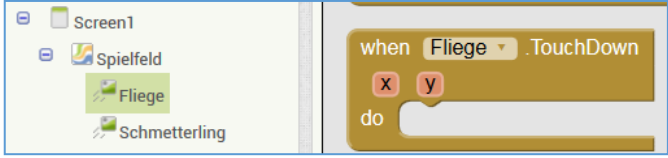
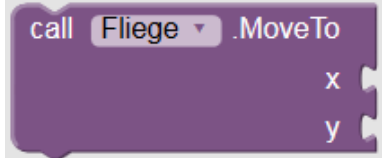
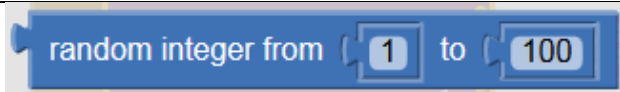
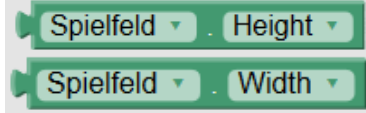
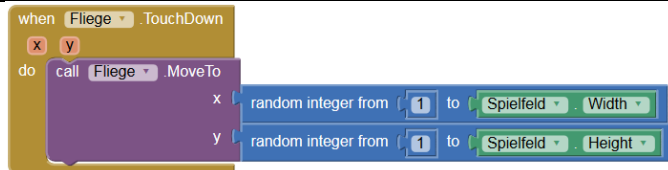




## 01 - Ortwechsel der Sprites nach Berührung

Der App-Inventor führt Anweisung nach Ereignissen (events) aus. Also z. B. beim Drücken eines Buttons, beim „Wischen“ über den Bildschirm, bei Berührung eines Objekts usw.

*Aufgabe:* Nach Berührung sollen die beiden Sprites jeweils an einem zufälligen anderen Ort der Leinwand erscheinen.

Das für die Fliege zugehörige Ereignis heißt "TouchDown" und befindet sich im Blockbereich der Fliege. Das Ereignis wird in den Viewer gezogen. (Analog beim Schmetterling)	
Bewegung wird mit der Methode "MoveTo" erreicht.	
Die Zufallsfunktion befindet sich im Bereich "Math"	
Der Zufallsbereich ist von 1 bis zur Höhe bzw. Breite des Spielfelds. Diese Werte befinden sich im Blockbereich von "Spielfeld".	
Die erforderlichen Blöcke werden zusammengesetzt. (Analog beim Schmetterling)	

(Vgl. TdIFliege01.aia)

## Erstes Starten der App

Es gibt drei verschiedenen Möglichkeiten zum Testen der App, man findet sie alle im Menübereich „connect“ des App-Inventors.

Die einfachste Variante ergibt sich, falls sich Handy und PC im gleichen WLAN-Netzwerk befinden und über dieses das Internet erreichbar ist. Auf dem Handy muss die App „MIT AI2 Companion“ installiert sein. (Achtung: Zur Installation auf dem Smartphone „fremde Quellen zulassen“ – Dies geschieht unter Einstellungen – Sicherheit). Die zur Version passende „apk-Installationsdatei“ befindet sich im Ordner AIU2.





Die App wird gestartet und im App Inventor 2 U "AI Companion" unter "connect" gewählt. Die Möglichkeit des QR-Codes ist für AI2U nicht gegeben, d. h. es ist ein Buchstabencode aus 6 Zeichen einzugeben. Anschließend ist die Durchführung der App zu sehen.

Eine weitere Variante ist die Verbindung mit Hilfe eines USB-Kabels. Dazu muss jedoch ein Treiber des Smartphones auf dem Computer installiert sein.

Als Letztes gibt es noch die Möglichkeit, einen Emulator zu starten. Dazu wählt man diesen im Bereich „connect“ aus. Es dauert dann ein bis zwei Minuten und die Simulation der App beginnt.

## 02-Ortswechsel der Sprites in bestimmten Zeitintervallen - die Komponente "Clock"

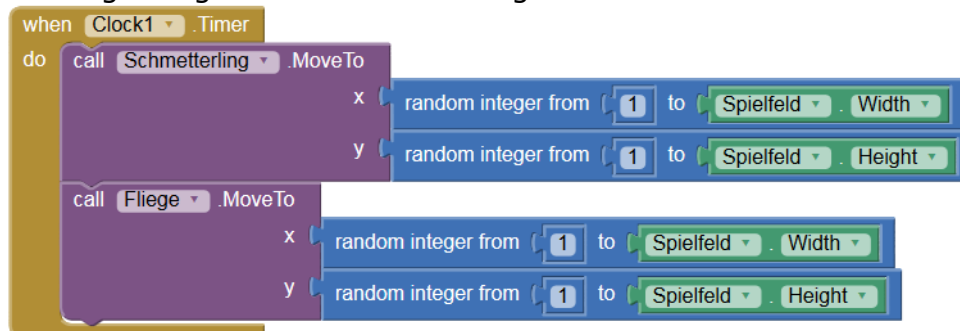
*Aufgabe:* Die beiden Sprites sollen den Ortswechsel nicht nur bei Berührung, sondern auch immer nach einem bestimmten Zeitintervall durchführen.

Dazu wird im Designer im Bereich "Sensors" eine "Clock" benötigt.

Bei Sensoren (wie auch bei weiteren Komponenten, wie z. B. Sound) handelt es sich um sogenannte "Non-visible components". Nach Ziehen in den Viewer erscheinen diese unterhalb des Screens. Auch bei ihnen können Namen und Attributwerte geändert werden.

Eine "Clock" hört auf das Ereignis "Timer".

Der zugehörige Block sieht nun folgendermaßen aus:



(vgl. TdIFliege02.aia)

## 03-Erzeugen eines "Sounds" nach Berührung

*Aufgabe:* Nach Berührung von Fliege oder Schmetterling sollen unterschiedliche akustische Signale erfolgen.

Ein Sound ist im Designermodus im Bereich "Media" zu finden. Da für Fliege und Schmetterling unterschiedliche Geräusche zu hören sein sollen, ist diese Komponente zweimal hinzuzufügen und deren Attributwerte und Namen sind anzupassen.



Im Blockmodus wird die Soundmethode "Play" jeweils im Ereignis "TouchDown" der beiden Sprites aufgerufen.

(vgl. TdiFliege03.aia)

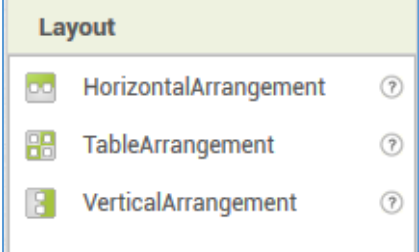
## 04-Zählen der Berührungen und Reset-Button im Designmodus

**Aufgabe:** Die Anzahl der Berührungen der Fliege bzw. des Schmetterlings soll gezählt werden. Zusätzlich soll die Möglichkeit gegeben sein, die Zahlen auf 0 zurückzusetzen.

Dazu werden im Designer sogenannte "Label" bzw. ein „Button“ benötigt. Diese sind in der Palette unter "User Interface" zu finden.

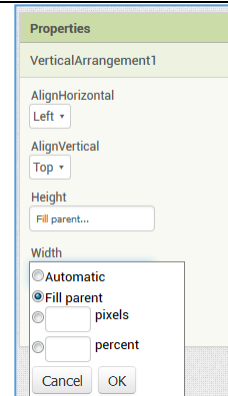
Um Text anzuzeigen, verwendet man im Allgemeinen "Label", d. h. der Text kann nur programmseitig geändert werden. Wird zusätzlich eine Benutzereingabe gefordert, so kann dies mit Hilfe einer "TextBox" realisiert werden.

Um die Labels bzw. den Button auf dem Screen1 zu platzieren, wird eine passende Stelle außerhalb der Leinwand benötigt. In der Fliegenfang-App sollen die Komponente unterhalb des Spielfelds erscheinen.

<p>Im Bereich Layout befinden sich mehrere mögliche Arrangements</p> <ul style="list-style-type: none"> <li>- horizontal (nebeneinander)</li> <li>- vertikal (übereinander)</li> <li>- Tabelle (in Tabellenform)</li> </ul>	
<p>Für die Fliegenfang-App wird ein vertikales Arrangement benötigt. Dieses wird z. B. oberhalb der Leinwand eingefügt.</p>	
<p>Das Spielfeld gehört jedoch zum Arrangement, es wird dorthin verschoben.</p>	



Auch die Attributwerte des Arrangements können verändert werden.

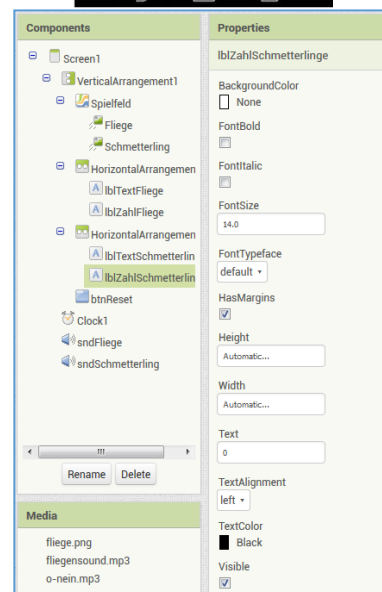
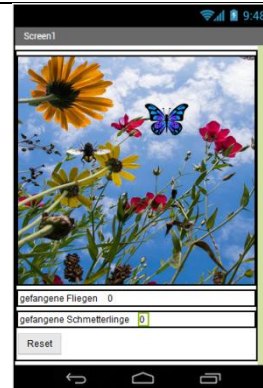


Die Labels und der Button werden hinzugefügt.

Dafür empfiehlt es sich, die Höhe des Spielfelds vorübergehend zu verkleinern, die Komponenten entsprechend einzufügen und die Größe des Spielfelds anschließend wieder zurückzusetzen.

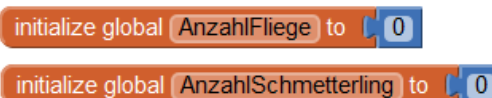
Um Labels nebeneinander erscheinen zu lassen, werden horizontale Arrangements benötigt.

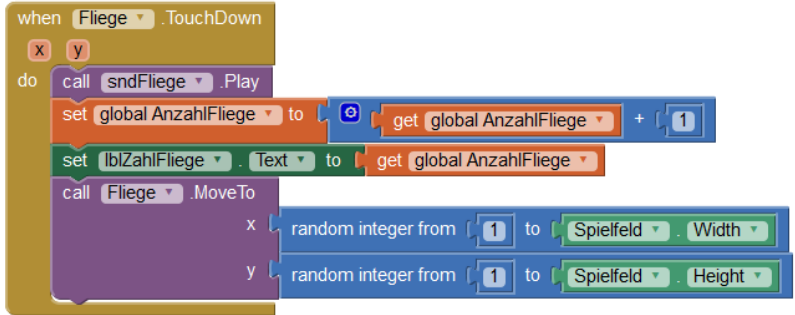
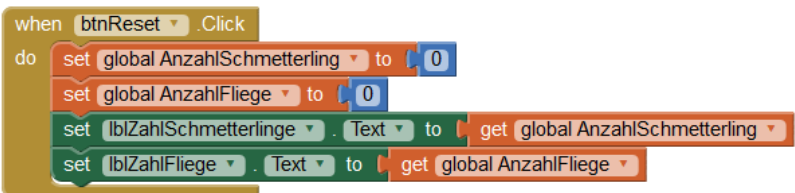
Die Attributwerte der Arrangements, Labels und Buttons werden ebenfalls angepasst.



## 04-Zählen der Berührungen und Reset-Button im Blockmodus - Variablen

Um die Anzahl der jeweils gefangenen Sprites zu zählen, werden Variablen benötigt. Diese befinden sich unter "Built-in".

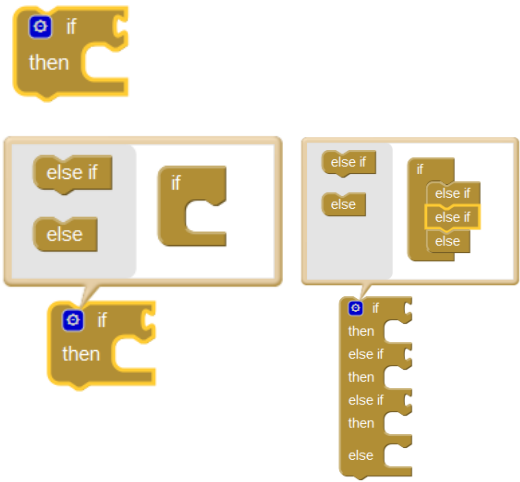


<p>Nach Berührung des jeweiligen Sprites wird die entsprechende Variable erhöht und dem Label der Wert als Text zugewiesen. (nebenstehend exemplarisch für die Fliege)</p>	 <pre> when Fliege TouchDown do   call sndFliege .Play   set global AnzahlFliege to (get global AnzahlFliege + 1)   set lblZahlFliege .Text to (get global AnzahlFliege)   call Fliege .MoveTo     x random integer from 1 to Spielfeld .Width     y random integer from 1 to Spielfeld .Height         </pre>
<p>Beim Drücken des Reset-Buttons werden die beiden Variablen auf 0 zurückgesetzt.</p>	 <pre> when btnReset Click do   set global AnzahlSchmetterling to 0   set global AnzahlFliege to 0   set lblZahlSchmetterlinge .Text to (get global AnzahlSchmetterling)   set lblZahlFliege .Text to (get global AnzahlFliege)         </pre>

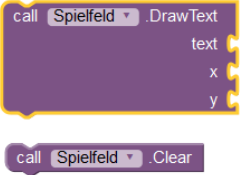
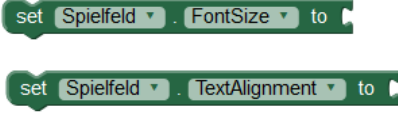
(vgl. TdIFliege04.aia)

## 05-Spiel verloren bei zu viel gefangenen Schmetterlingen - Kontrollstrukturen

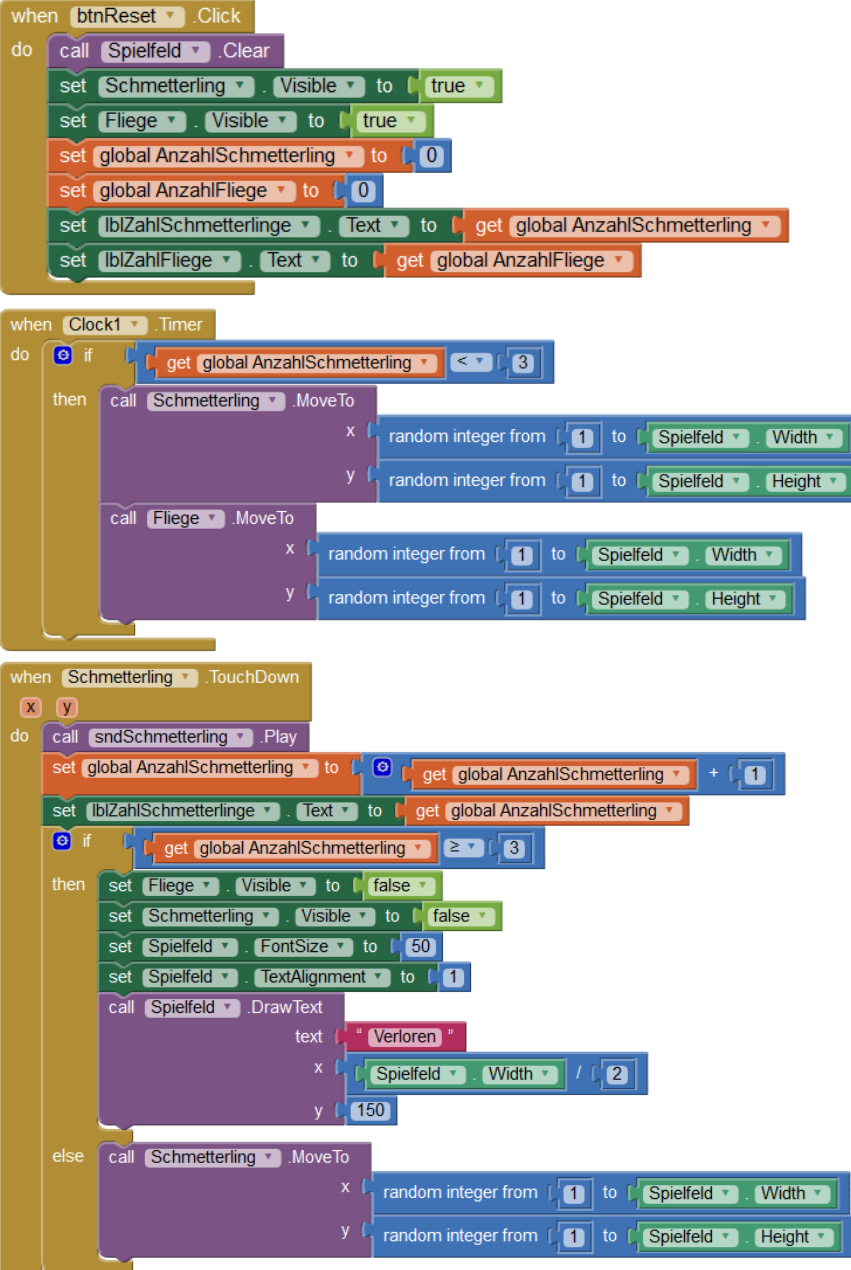
*Aufgabe:* Nach dem dritten gefangenen Schmetterling soll "Verloren" auf dem Bildschirm erscheinen, nach „Reset“ ist der Text zu entfernen, ebenso sollen sich Fliege und Schmetterling nur bewegen, falls noch keine drei Schmetterlinge erschlagen wurden.

<p>Es werden Kontrollstrukturen benötigt. Diese findet man unter "Control".</p> <p>Eine einseitige Auswahl erhält man z. B. durch nebenstehenden Block</p> <p>Um eine mehrseitige Auswahl zu erhalten, wird das Zahnrad angeklickt und so viele "else if" bzw. "else" in die Struktur gezogen, wie benötigt.</p>	
--	--



Auf eine Canvas lässt sich mit Hilfe der Methode "DrawText" schreiben. Die Methode "Clear" bereinigt alles Geschriebene.	
Schriftattribute lassen sich ändern. Siehe nebenstehende Beispiele	

Übersicht über die veränderten Blöcke:

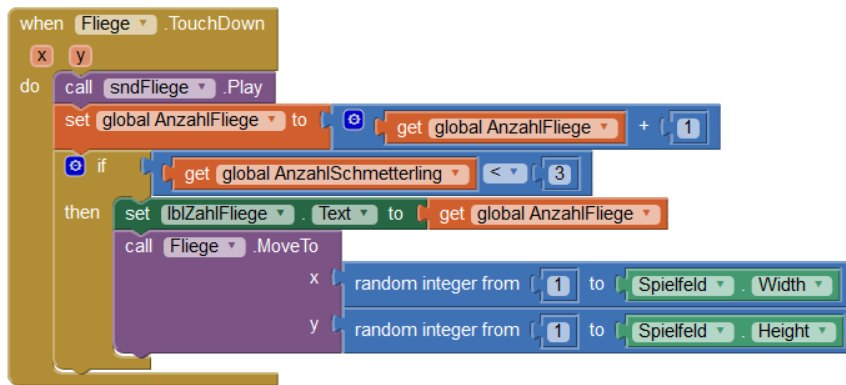


```

when btnReset.Click
do
  call Spielfeld.Clear
  set Schmetterling.Visible to true
  set Fliege.Visible to true
  set global AnzahlSchmetterling to 0
  set global AnzahlFliege to 0
  set lblZahlSchmetterlinge.Text to get global AnzahlSchmetterling
  set lblZahlFliege.Text to get global AnzahlFliege

when Clock1.Timer
do
  if (get global AnzahlSchmetterling < 3)
  then
    call Schmetterling.MoveTo
      x random integer from 1 to Spielfeld.Width
      y random integer from 1 to Spielfeld.Height
    call Fliege.MoveTo
      x random integer from 1 to Spielfeld.Width
      y random integer from 1 to Spielfeld.Height

when Schmetterling.TouchDown
do
  call sndSchmetterling.Play
  set global AnzahlSchmetterling to (get global AnzahlSchmetterling + 1)
  set lblZahlSchmetterlinge.Text to get global AnzahlSchmetterling
  if (get global AnzahlSchmetterling ≥ 3)
  then
    set Fliege.Visible to false
    set Schmetterling.Visible to false
    set Spielfeld.FontSize to 50
    set Spielfeld.TextAlignment to 1
    call Spielfeld.DrawText
      text "Verloren"
      x (Spielfeld.Width / 2)
      y 150
  else
    call Schmetterling.MoveTo
      x random integer from 1 to Spielfeld.Width
      y random integer from 1 to Spielfeld.Height
  
```



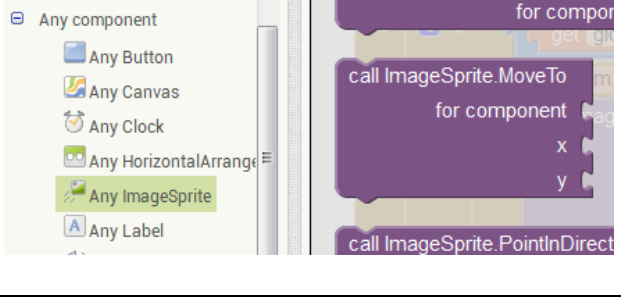
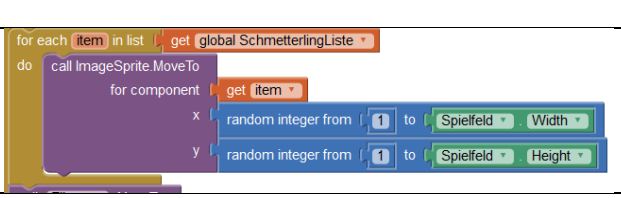
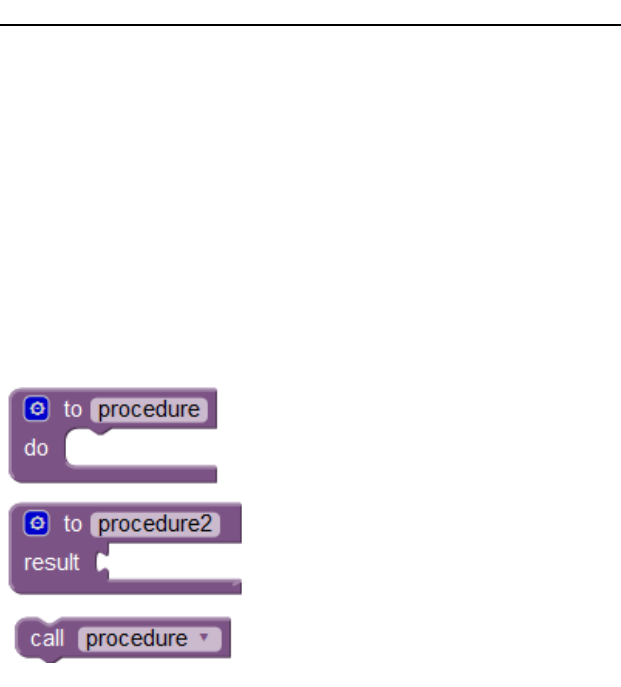
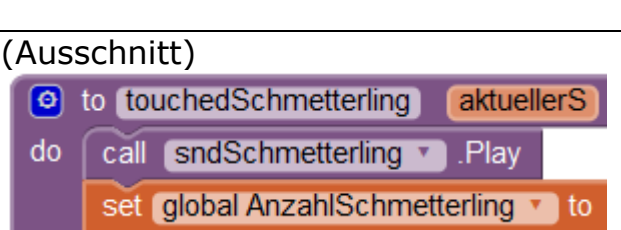
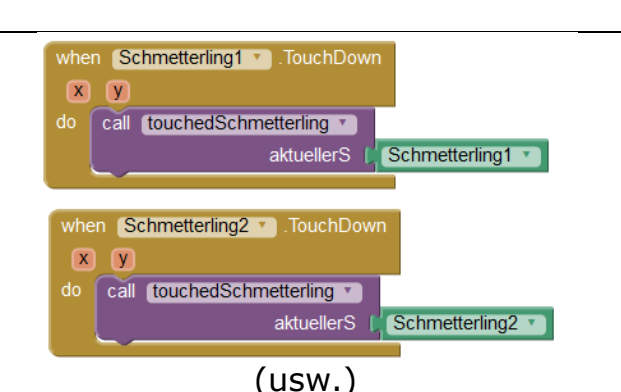
(vgl. TdIFliege05.aia)

## 06-Viele Schmetterlinge... - Listen und Prozeduren

*Aufgabe:* Um den Spielverlauf zu erschweren, werden nun (im Designer) einige weitere Schmetterlinge eingefügt und mit Schmetterling1, Schmetterling2 usw. benannt. In unserem Beispiel soll es insgesamt sieben Schmetterlinge geben.

Da alle Schmetterlinge die gleichen Eigenschaften haben und bei den gleichen Ereignissen die gleichen Methoden ausführen sollen, bietet es sich an, mit allen Schmetterlingen eine Liste zu erstellen.	
Dazu wird zunächst eine leere Liste als globale Variable benötigt.	initialize global SchmetterlingListe to create empty list
Beim erstmaligen Initialisieren des Bildschirms "Screen1" wird die Liste mit den Schmetterlingen gefüllt.	
Bei jedem Timer-Ereignis müssen nun alle Schmetterlinge an einen zufälligen anderen Ort verschoben werden.	
Um alle Elemente einer Liste zu durchlaufen gibt es eine zugehörige Kontrollstruktur	for each item in list do

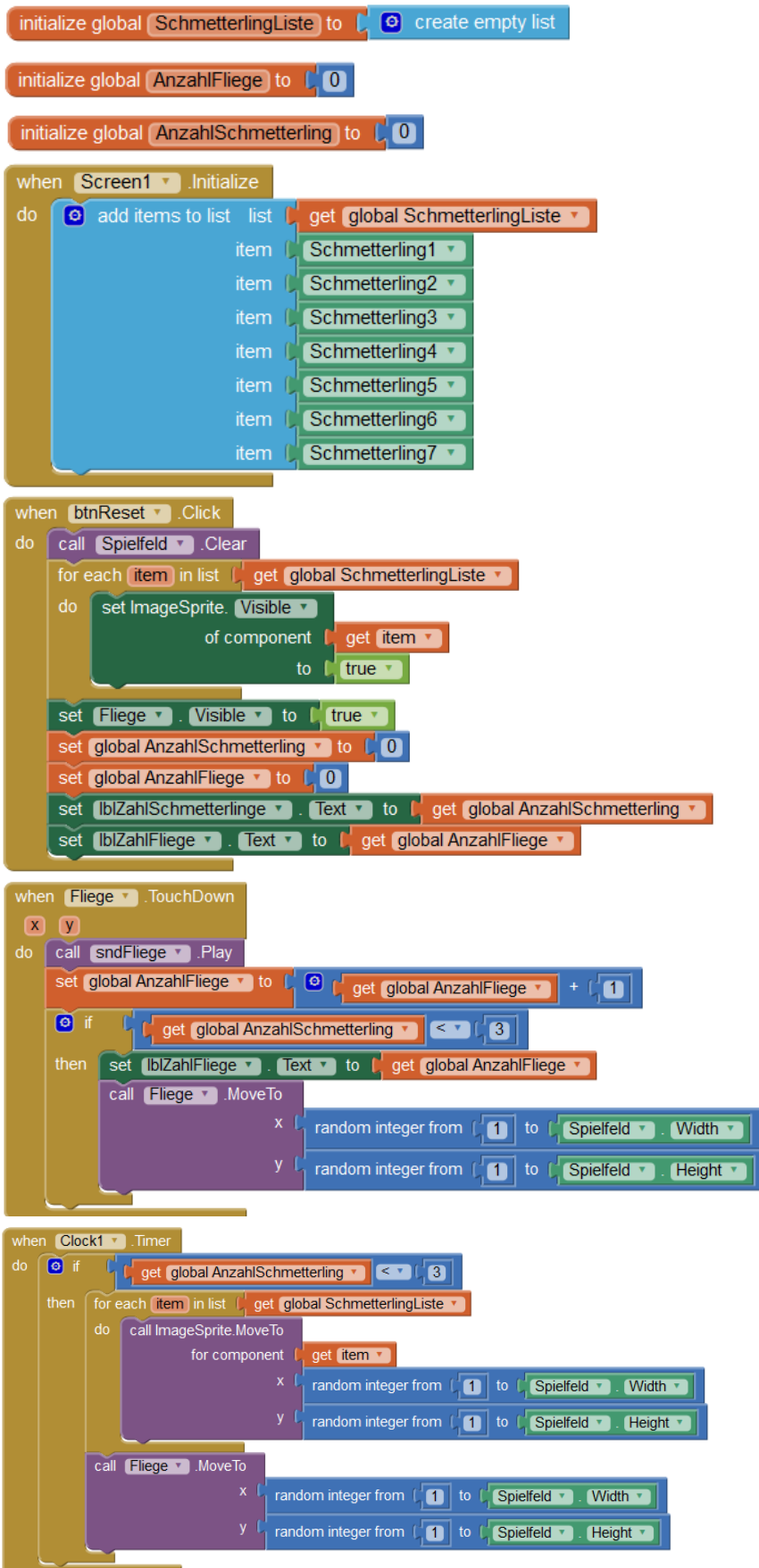


<p>Um ein ImageSprite aus einer Liste zu verwenden, gibt es "Any component". Hier verwendet man dann z. B. die Methode "ImageSprite.MoveTo"</p>	
<p>Im Timer-Ereignis sieht dieser Block dann wie nebenstehend aus</p>	
<p>Bei Berührung eines (beliebigen) Schmetterlings, soll ein Sound abgespielt werden, der entsprechende Schmetterling seine Position verändern und die Anzahl mitgezählt werden.</p> <p>Es empfiehlt sich eine eigene Prozedur dafür zu schreiben, die für jeden Schmetterling aufgerufen wird.</p> <p>Im Bereich Built-in gibt es Procedures - ohne Rückgabewert, mit Rückgabewert bzw. der Aufruf. Über das Zahnrad können Parameter eingebunden werden.</p>	
<p>Eine Prozedur namens "touchedSchmetterling" mit dem aktuell berührten Schmetterling als Parameter wird erzeugt.</p>	<p>(Ausschnitt)</p> 
<p>Der Prozeduraufruf erfolgt für jeden Schmetterling</p>	 <p>(usw.)</p>

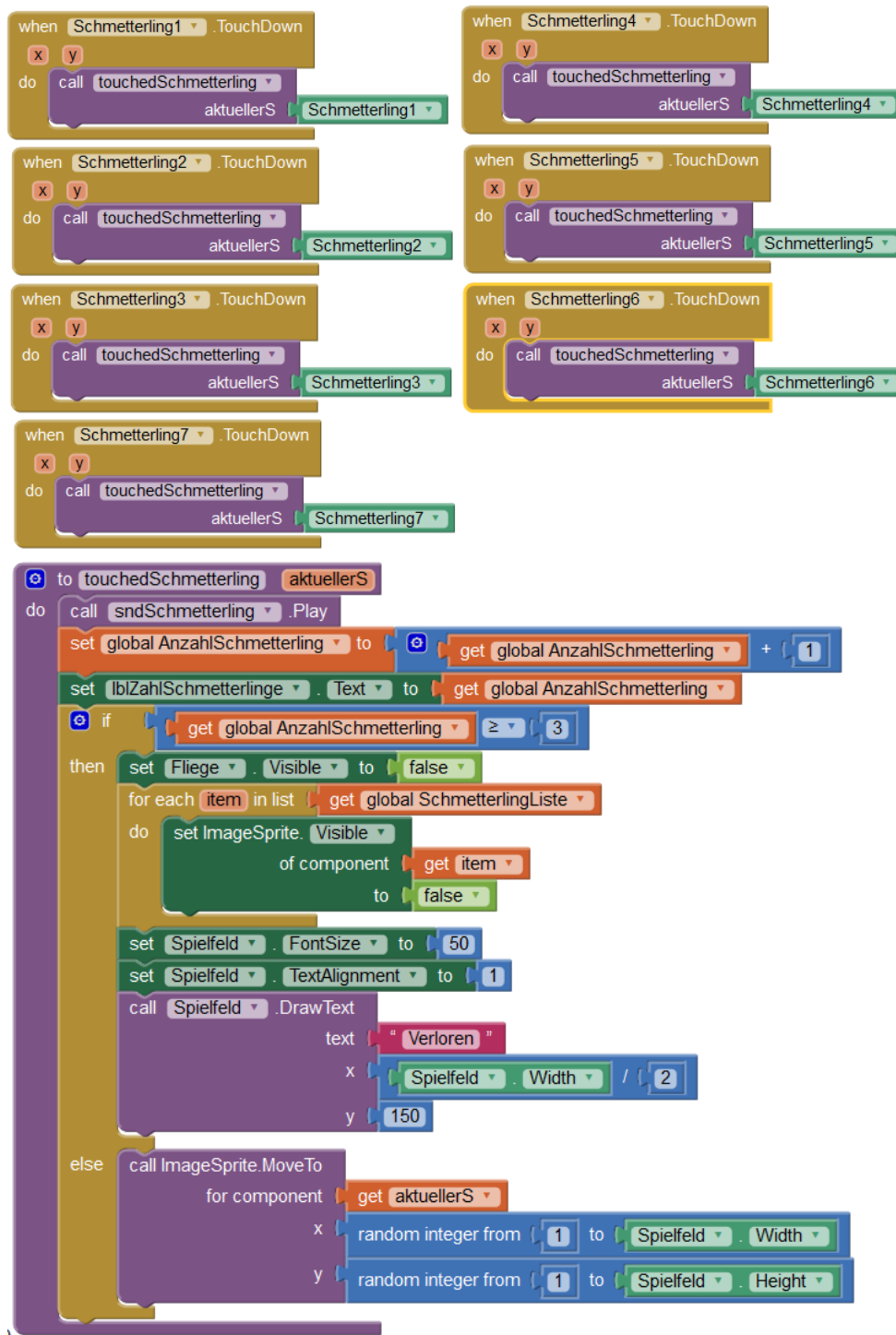




## Alle Blöcke im Überblick:







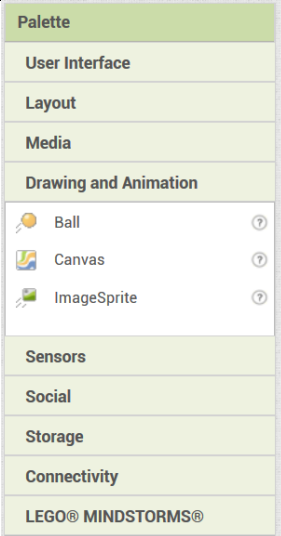
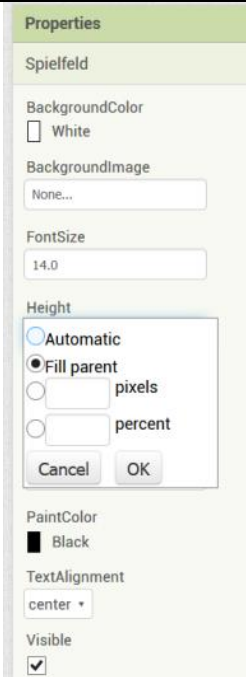
(vgl. TdiFliege06.aia)



## Die App „Pong“

In dieser App rollt ein Ball über den Bildschirm. Am unteren Ende befindet sich ein „Paddle“, das mit Wischbewegungen gesteuert werden kann. Das Paddle soll den Ball davon abhalten, den Boden zu berühren.

### Erstes Design


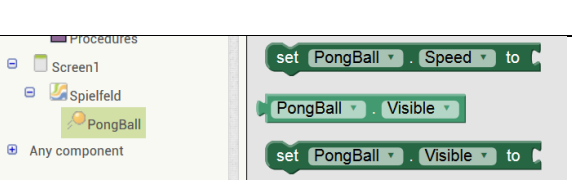
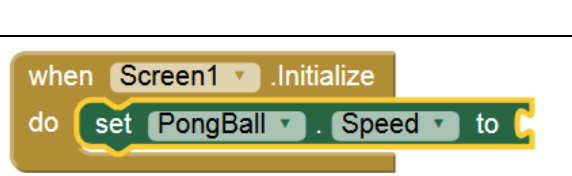
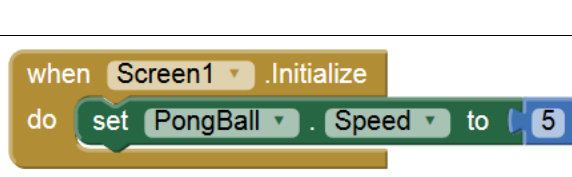
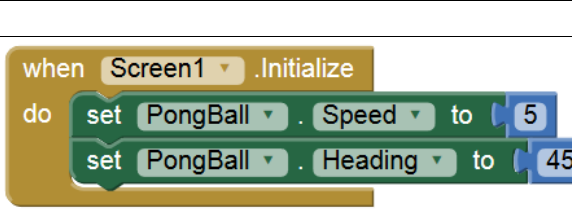
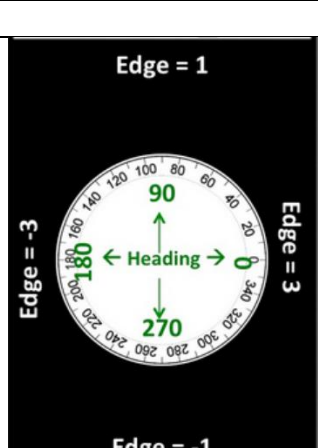
<p>Zu Beginn wird nur ein Ball benötigt. Dieser befindet sich unter „Drawing and Animation“. Damit dieser sinnvoll programmiert werden kann, muss er auf einer „Canvas“ liegen.</p>	
<p>Die Attributwerte der Canvas – umbenannt in „Spielfeld“ werden auf maximal möglich, also auf „Fill Parent“ gestellt.</p>	

### 01 - Erstes Programmieren: Bewegen des Balls

*Aufgabe:* Der erste Programmierschritt ist das „Herumrollen“ des Balls. Dies soll zunächst schon beim Starten der App und nicht erst nach einem Ereignis geschehen.

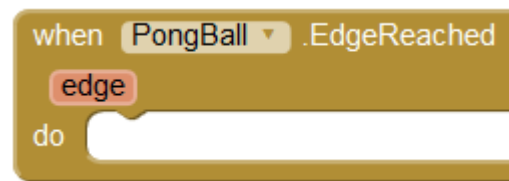
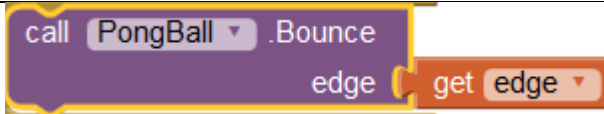


Alles, was beim erstmaligen Aufruf eines Bildschirms (screen) geschehen soll, wird in „initialize“ festgelegt.

<p>Aus dem Blockbereich „Screen1“ wird „When Screen1.Initialize“ in den Viewer gezogen.</p>	
<p>Als nächstes müssen wir dem Ball eine Geschwindigkeit und eine Richtung geben. Dies geschieht mit den Attributwerten „speed“ und „heading“. Wir scrollen also im Blockbereich „PongBall“ (so hieß unser Ball) nach unten, bis wir „set PongBall.speed to“ finden.</p>	
<p>Diesen Block ziehen wir in den Viewer.</p>	
<p>Im Blockbereich „Math“ finden wir Zahlen. Wir ziehen sie in den Viewer und ändern den Wert ab. Die Zahl gibt an, um wie viel Pixel der Ball pro Zeitintervall „wandert“.</p>	
<p>Den Attributwert „Heading“ können wir nun einfacher setzen. Wir kopieren mit STRG C „set PongBall.Speed to 5“ und fügen es mit STRG V wieder ein. Anschließend ändern wir „Speed“ über den kleinen Pfeil zu „Heading“ und stellen einen entsprechenden Wert ein.</p>	
<p>Bei heading ist rechts 0, oben 90, links 180 und unten 270.</p>	

## 02-Abprallen des Balls vom Rand

*Aufgabe:* Der Ball bewegt sich zu Beginn in die richtige Richtung, prallt dann jedoch nicht von der Bande ab. Dies gilt es zu berichtigen.

Das hierfür zu programmierende Ereignis heißt „EdgeReached“.	
Die Ränder (edge) sind nummeriert. Die zugehörigen Nummern sind auch weiter oben bei „heading“ zu sehen.	Oben: Edge = 1 Links: Edge = -3 Rechts: Edge = 3 Unten: Edge = -1
Um den Ball vom Rand abprallen zu lassen, wird die Methode „call PongBall.Bounce“ verwendet. Diese Methode liefert den aktuellen Rand unter „edge“ mit.	

## 03-Einfügen eines Paddles

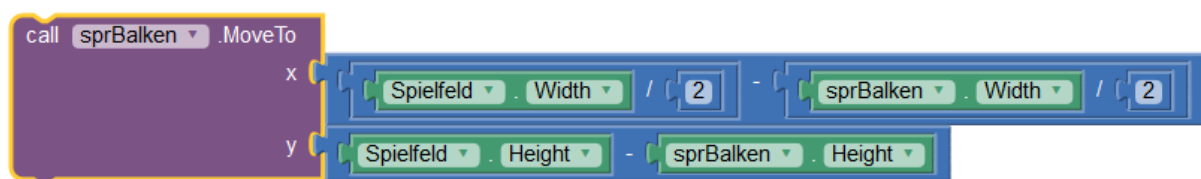
*Aufgabe:* Um den Ball mit Hilfe eines Paddles, hier Balken genannt, steuern zu können, wird ein solcher benötigt.

Im *Designermodus* wird ein neues Sprite hinzugefügt, das Aussehen über ein Bild hochgeladen. Sinnvolle Werte für die Breite und Höhe ein müssen eingestellt werden (z. B. 90 px auf 20 px)

Im *Block-Modus* wird dann im Ereignis „Screen1.initialize“ die Position des Balkens bestimmt. Dieser soll sich mittig am unteren Rand der Leinwand befinden.

Die Position (x/y) eines Sprites ist immer der Punkt links/oben des Sprites.

Die Methode, um einen Sprite an eine bestimmte Position zu fügen, heißt "MoveTo".



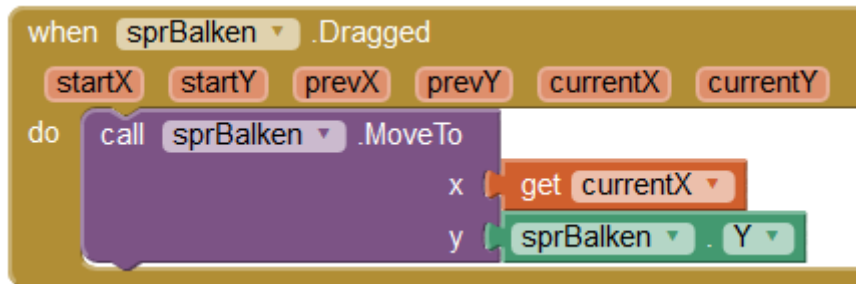


## 04-Bewegen des Balkens

*Aufgabe:* Der Balken soll durch Ziehen mit dem Finger nach links bzw. rechts bewegt werden können.

Das hierfür benötigte Ereignis heißt „Dragged“.

Die Attribute `currentX` bzw. `currentY` des "Zieh-Ereignisses" geben dabei die aktuelle Position des Sprites an.



## 05-Steuern des Balls mit Hilfe des Balkens

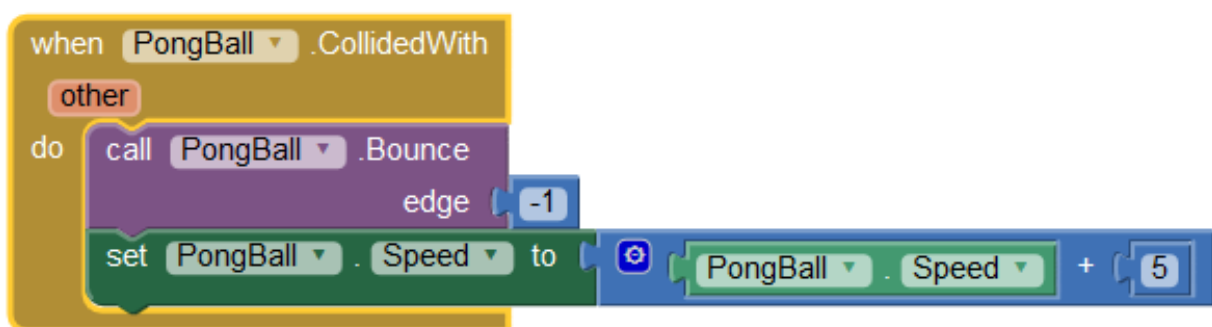
*Aufgabe:* Trifft der Ball auf den Balken, so soll dieser nicht durch ihn hindurchgehen, sondern sofort von ihm abprallen, außerdem soll die Geschwindigkeit des Balls (z. B. um 5) erhöht werden.

Das benötigte Ereignis heißt „CollidedWith“.

Die Attributwerte von „Heading“ und „Speed“ sind anzupassen.

Für Heading gilt: Der Ball kommt von oben, würde der Ball bis zum Rand kommen, wäre `edge = -1`. Mit diesen Informationen lässt sich die Methode "Bounce" aufrufen.

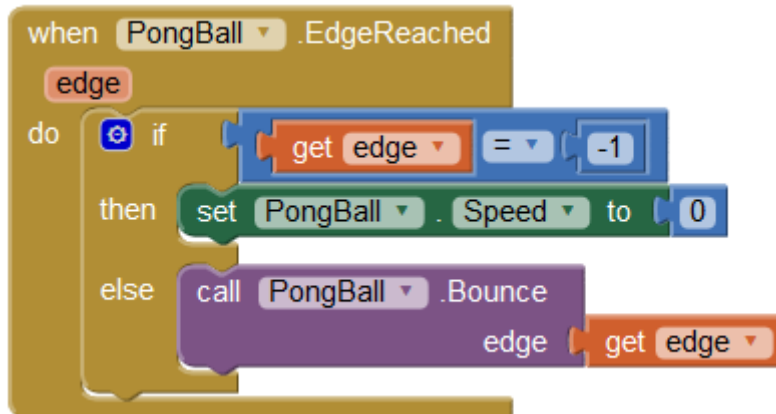
Für Speed gilt: Die Geschwindigkeit soll um 5 erhöht werden.



## 06-Stop, falls unterer Rand berührt

Aufgabe: Wird der Ball nicht mit dem Balken getroffen und erreicht somit den unteren Rand, so soll er sich nicht mehr bewegen.

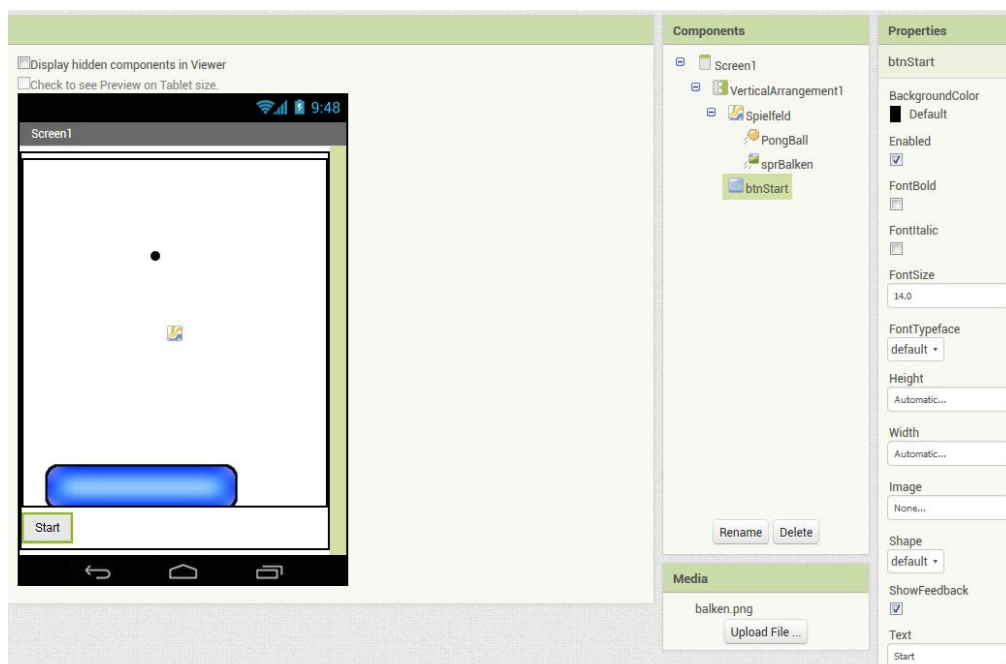
Im Ereignis "EdgeReached" wird, falls edge der untere Rand ist, die Geschwindigkeit auf Null gesetzt.



## 07-Erstellen eines Buttons, Positionieren des Balls und Pause vor Druck auf Button

Aufgabe: Zu Beginn des Spiels soll sich der Ball an einer Startposition oberhalb des Balkens befinden. Der eigentliche Spielbeginn soll durch einen Button auslösbar sein. Zusätzlich soll, falls der Ball nicht vom Balken getroffen wurde, nicht nur die Geschwindigkeit auf Null gesetzt, sondern der Ball wieder auf die Startposition gesetzt werden.

Hinzufügen des Buttons im Designermodus:





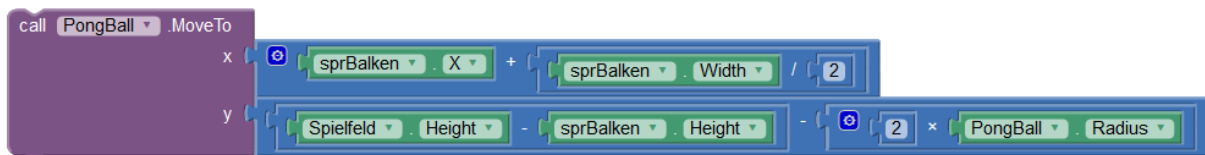
Positionieren des Balls im Blockmodus:

Mögliche Koordinaten (mittig über den Balken) der Startposition des Balls sind:

$x = \text{Balken.X} + \text{Balken.Width}/2;$

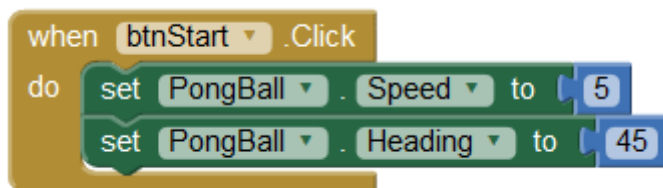
$y = \text{Spielfeld.Height} - \text{Balken.Height} - 2 * \text{PongBall.Radius}$

Die Position sollte zum einen in Initialize, zum anderen nachdem der Ball nicht vom Balken getroffen wurde, gesetzt werden.



Warten auf Button-Druck im Blockmodus:

Der Ball soll sich erst nach Drücken des Startbuttons bewegen. Deshalb muss das Setzen der Geschwindigkeit und der Richtung aus Initialize rauszunehmen und in das entsprechende Ereignis zu verschieben.



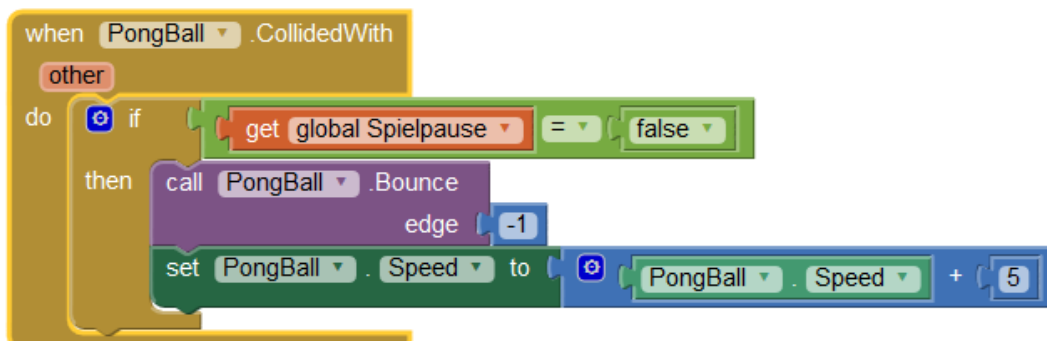
Nun ist jedoch noch zu bedenken, dass auf Grund der Startposition des Balls dies gleichzeitig eine Kollision mit dem Balken bedeutet und das entsprechende Ereignis aufgerufen wird. Dies bewirkt, dass sich der Ball doch wieder bewegt. Um dies zu verhindern, fügen wir eine neue Variable ein, die Auskunft über den aktuellen Spielstand geben soll: Spielpause wahr/falsch.

Überlegung, an welchen Stellen die Variable "Spielpause" gesetzt werden muss:

True: In Initialize und wenn der Ball den unteren Rand erreicht

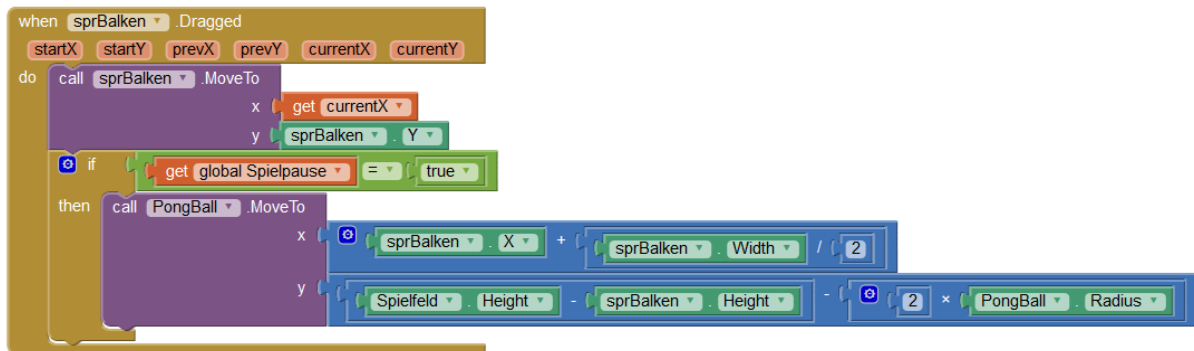
False: Nach Drücken des Startbuttons

Im Kollisionsereignis ist dann zu überprüfen, ob gerade Pause ist oder nicht.

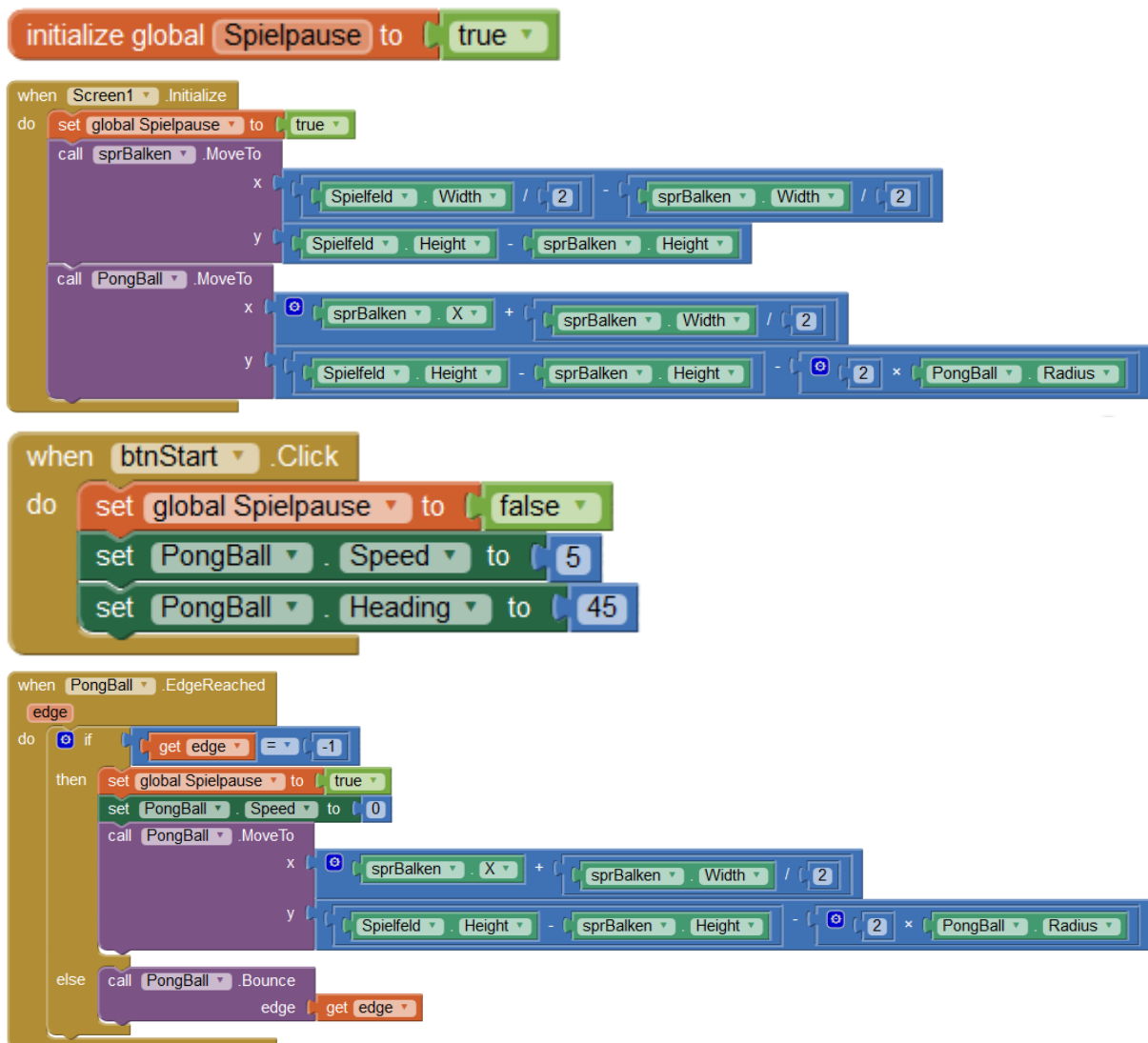




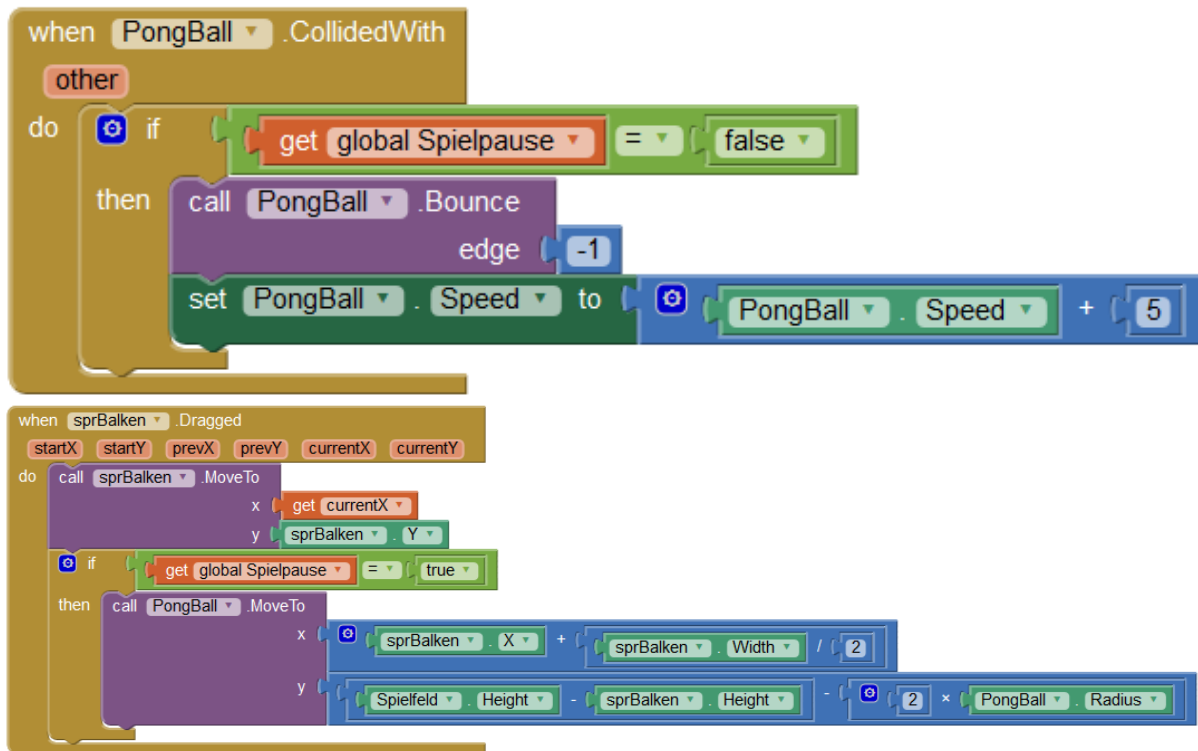
Befindet sich der Ball bei Spielpause direkt oberhalb des Balkens, so wäre es schön, wenn dieser sich beim Verschieben des Balkens mitbewegen würde. Das Ereignis `sprBalken.Dragged` ist also entsprechend anzupassen.



## 08-Gesamtübersicht aller Blöcke







## 09-Anregungen

Die App lässt sich beliebig erweitern, z. B. durch Einfügen eines Highscores, Beschränkung auf eine bestimmte Zeit - der Fantasie der Schüler ist hier freier Lauf gelassen ;-)

### Weitere Tutorien und Dokumentationen

<http://appinventor.mit.edu/explore/ai2/tutorials.html>

<http://appinventor.mit.edu/explore/library.html>

<http://www.appinventor.org/book2> (Nach unten scrollen - Buch ist online)

### Quellennachweis

#### Bildmaterial:

Blumenwiese:  
Schmetterling:  
Fliege:

[www.flickr.com/Ilona\\_Kuckuck](http://www.flickr.com/Ilona_Kuckuck)  
[www.pixabay.com](http://www.pixabay.com) (kein Bildnachweis notwendig)  
[www.pixabay.com](http://www.pixabay.com) (kein Bildnachweis notwendig)

#### Tonmaterial:

Fliegensound:

[www.cayzland-music.de](http://www.cayzland-music.de)  
(kostenlos für nichtgewerbliche Schulprojekte)

Schmetterlingsound:

selbst erstellt

