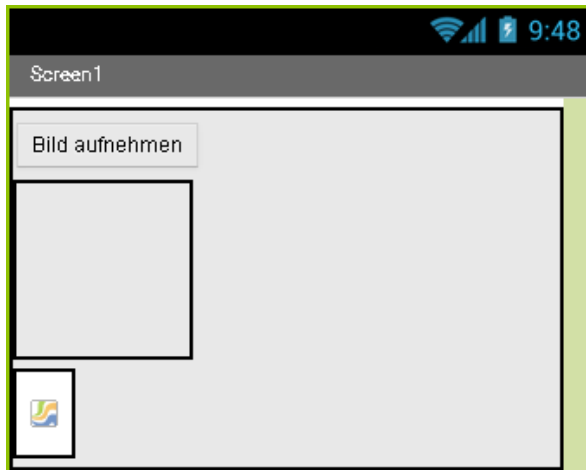


0 Ein Foto soll es machen

So kann es aussehen



Das soll passieren

Zuerst soll per Knopfdruck mit der Kamera ein Foto geschossen werden. Dieses geschossene Bild soll dann als Hintergrund der Leinwand gesetzt werden.

Außerdem soll mit den *Screen Arrangements* gleich in der Vertikalen ein bisschen für Ordnung sorgen.

Neue Komponenten

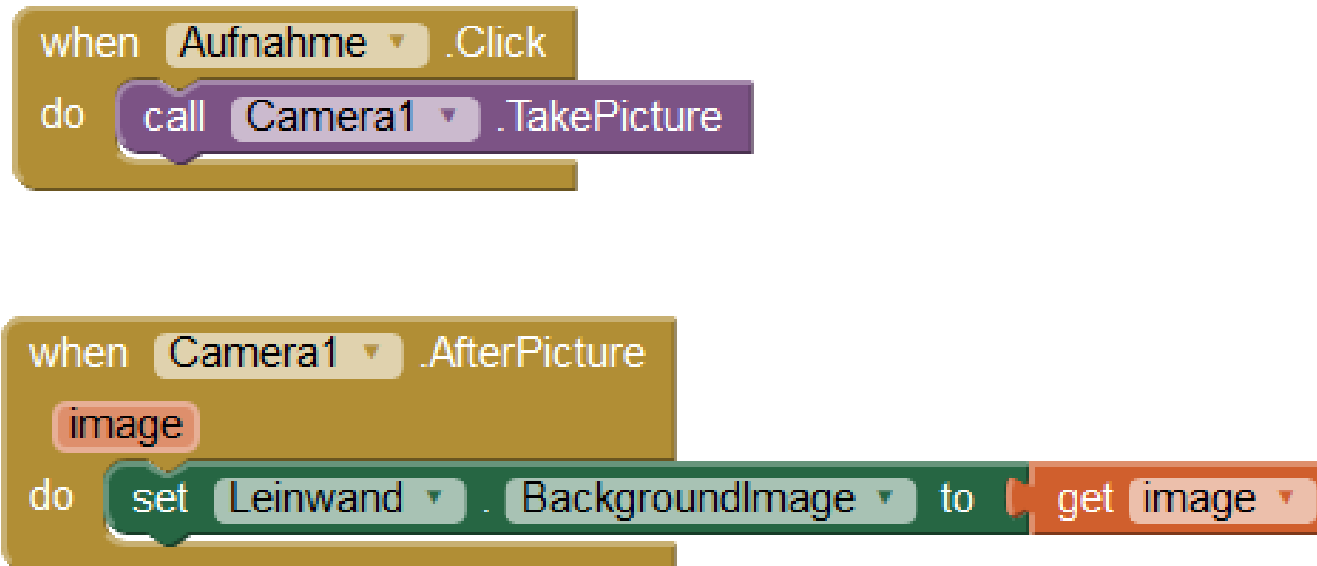
1 x Button „Bild aufnehmen“
 1 x Leinwand (Canvas)
 1 x Kamera (Camera)
 1 x VerticalArrangement

Tipps

Die benötigten Bausteine kannst du bei den neuen Komponenten (Button, Leinwand, Kamera) finden.

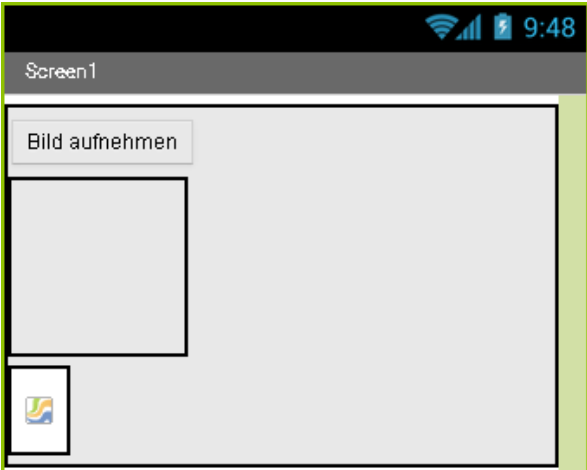
0 Ein Foto soll es machen

Lösung



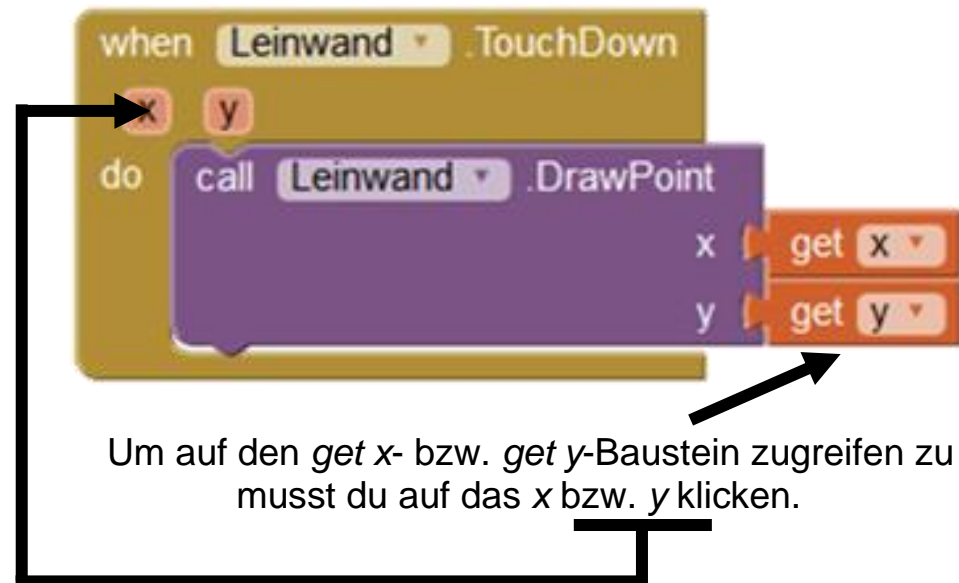
Um auf den get-image-Baustein zugreifen zu können, musst du auf image klicken.

1.1 Punkte sind der Anfang

So kann es aussehen	Das soll passieren
	<p>Bis jetzt lässt sich mit der App ein Foto schießen und auf der Leinwand (Canvas) anzeigen.</p> <p>Nun soll aber ein Punkt auf der Leinwand gezeichnet werden, sobald über den Touchscreen die Leinwand berührt wird.</p>
Neue Komponenten	Tipps
Keine	Die benötigten Bausteine befinden sich im <i>Block Editor</i> unter <i>Leinwand</i> .

1.1 Punkte sind der Anfang

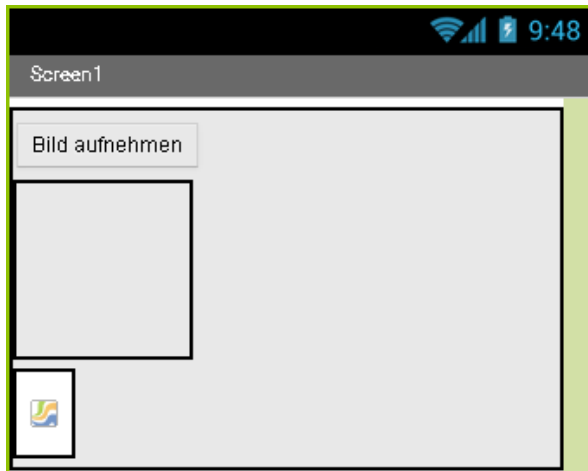
Lösung



Um auf den *get x*- bzw. *get y*-Baustein zugreifen zu können, musst du auf das *x* bzw. *y* klicken.

1.2 Viele Punkte machen einen Strich

So kann es aussehen



Das soll passieren

Nun können auf die Leinwand und damit auf das geschossene Foto Punkte gemalt werden.

Es fehlen nun noch Striche. Dies ist die neue Aufgabe für dich.

Der erste Druck auf den Touchscreen soll den Start des Striches bilden, durch das Ziehen auf dem Touchscreen soll der Strich gezeichnet werden. Lässt man den Touchscreen los, wird der Strich beendet.

Neue Komponenten

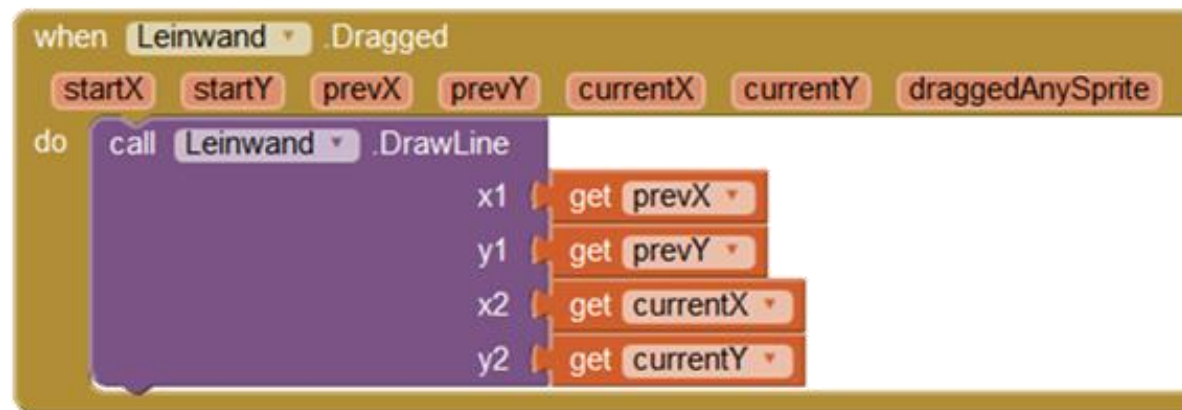
Keine

Tipps

Die benötigten Bausteine befinden sich im *Block Editor* unter *Leinwand*.

1.2 Viele Punkte machen einen Strich

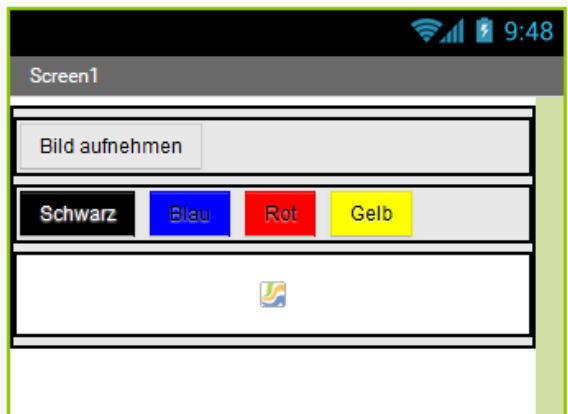
Lösung



Verwendet man für $x1$ und $y1$ die Bausteine *startX* und *startY*, dann ergibt sich ein toller Effekt.
(Wobei dies die gestellte Aufgabe nicht lösen würde ... vielleicht ein Extra für deine App?)

2.1 Es werde bunt!

So kann es aussehen



Das soll passieren

Nun soll Farbe ins Spiel oder vielmehr in die App kommen.

Die Aufgabe lautet, dass sich, wenn du auf einen Button für eine Farbe klickst, die Farbe auf der Leinwand ändert.

Für den Anfang wären Schwarz, Blau, Rot und Gelb eine gute Idee, du kannst aber auch andere Farben verwenden. Damit diese schön nebeneinander angeordnet werden können, benötigst du die *HorizontalArrangement-Komponente*.

Neue Komponenten

4 x Button (für die Farben)
1 x HorizontalArrangement (für die Buttons)

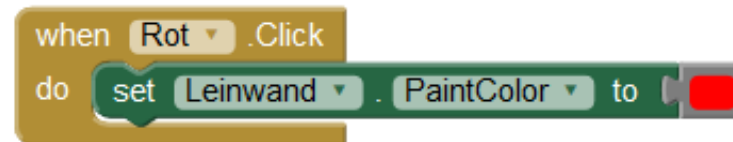
Tipps

Sobald du im *Block Editor* die Programmierung für einen Button fertig hast, kannst du dessen Programmierung ganz einfach kopieren und so vervielfältigen. Klicke dazu auf die Programmierung des Buttons und wähle *Duplicate* aus. Dann musst du nur noch die Bezeichnung des Buttons und die Farbe anpassen.

2.1 Es werde bunt!

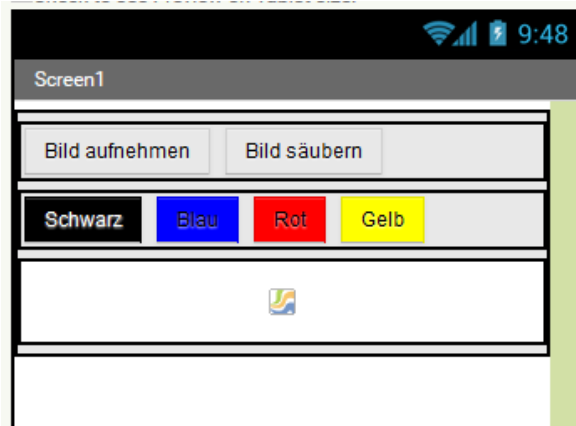
Lösung

Suche im *Block Editor* einen Farb-Button und ziehe den Click-Baustein auf die Arbeitsfläche. Im Auswahlmenü der Leinwand findest du den unteren grünen Baustein, der in die Lücke des Click-Bausteins gehört. Unter *Colors* findest du Bausteine für die Farben. Ziehe die passende Farbe in die hintere Lücke.



2.2 Jeder macht mal Fehler ...

So kann es aussehen



Das soll passieren

Es wäre sehr praktisch, wenn man das übermalte Foto wieder säubern könnte.
Füge dazu einen Button hinzu.

Neue Komponenten

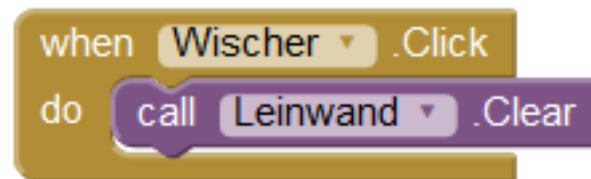
1 x Button *Bild säubern*

Tipps

Keine

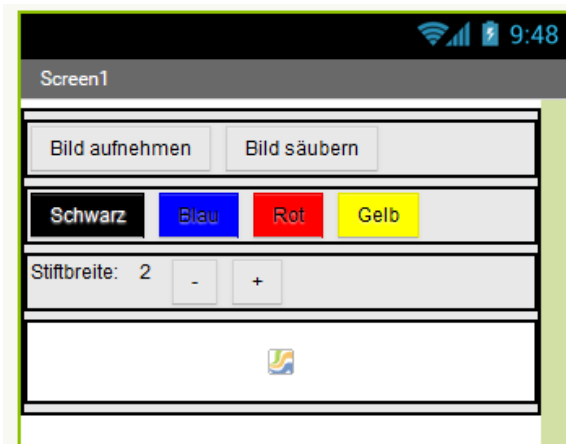
2.2 Jeder macht mal Fehler ...

Lösung



2.3 Mal größer und mal kleiner

So kann es aussehen



Das soll passieren

Dir wird aufgefallen sein, dass die Striche bisher sehr dünn sind.

Sorge nun dafür, dass man die Stiftbreite verändern kann. Wenn du auf „+“ klickst, soll die Stiftbreite erhöht werden, durch Klicken auf „-“ verringert.

Wenn sich die Stiftbreite verändert, soll der neue Wert auch angezeigt werden.

Der Startwert der Stiftbreite soll 2 sein.

Neue Komponenten

2 x Label (*Stiftbreite* und *Wert der Stiftbreite*)

2 x Buttons (+ und -)

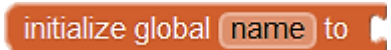
Tipps

Du brauchst eine Variable für die Stiftbreite.

2.3 Mal größer und mal kleiner

Lösung

Damit der Stift (dein Finger) in verschiedenen Stiftbreiten malen kann, musst du ihm zunächst eine Stiftbreite zuweisen. Dazu musst du eine **Variable** anlegen. Gehe dazu im Bereich *Blocks* auf die Kategorie *Variables* und ziehe nebenstehenden Baustein in den Viewer.

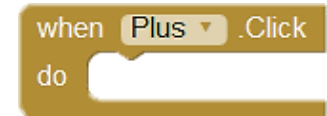


Wenn du auf *name* klickst, kannst du den Namen der Variable ändern. Rechts an dem Baustein kannst du den Startwert festlegen. Ändere nun den Namen der Variable auf *Stiftbreite* und hänge den nebenstehenden Baustein (du findest ihn in der *Math*-Kategorie) an die rechte Lücke.



Ändere anschließend den Startwert auf 2.

Um beim Klicken auf den *Plus*-Button die Stiftbreite zu vergrößern, ziehe nebenstehenden Baustein in den Arbeitsbereich.



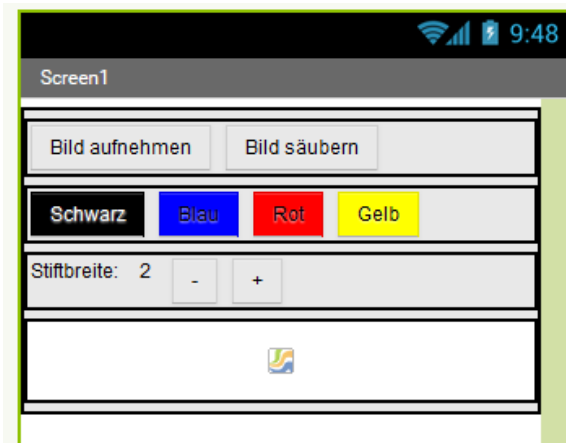
Anschließend musst du dafür sorgen, dass die Variable *Stiftbreite* um 1 erhöht wird und sich die Stiftbreiten bei der Leinwand und dem Label anpassen. Dafür brauchst du die folgenden Bausteine.

Suche sie zusammen und füge sie in dieser Reihenfolge in den Baustein *when Plus.Click do* ein. Die Programmierung für den *Minus*-Button verläuft entsprechend.



2.4 Die Stiftbreite und ihre Grenzen

So kann es aussehen



Das soll passieren

Bisher kann die Stiftbreite beliebig groß werden und sogar kleiner als 1. Das sollte aber nicht so sein.

Um das zu verhindern, benötigst du Bedingungen. Sie legen den Geltungsbereich für bestimmten Aktionen fest.

Ändere deine App so ab, dass die Stiftbreite nicht kleiner als 1 und nicht größer als 6 werden kann.

Neue Komponenten

Tipps

Keine

Du brauchst *if-then*-Bedingungsbausteine.

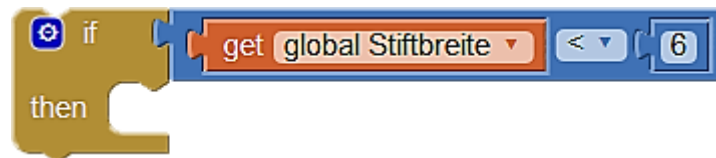
2.4 Die Stiftbreite und ihre Grenzen

Lösung

Für den *Minus*-Button soll zum Beispiel gelten: Wenn (*if*) die Stiftbreite größer als 1 ist, dann (*then*) wird der Wert der Stiftbreite verringert und die Stiftbreite der Leinwand sowie der Text des Labels aktualisiert. Wenn die Stiftbreite 1 oder kleiner ist, soll dies aber nicht passieren. Um das zu erreichen, braucht man im Block Editor diese Bausteine:



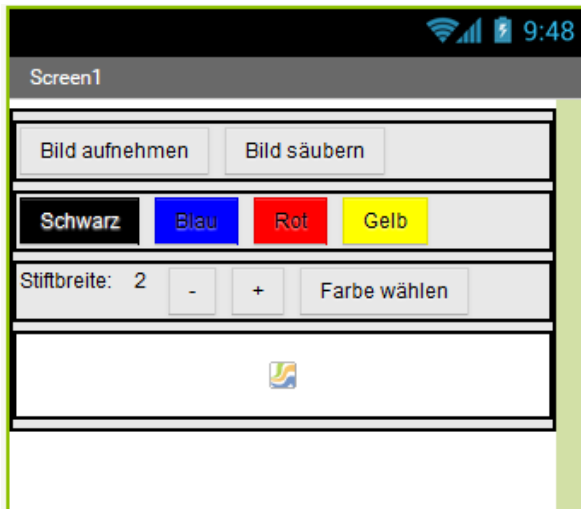
Für den *Plus*-Button benötigst du auch den *if-then*-Baustein, nur das die Bedingung eine andere ist. Damit die Stiftbreite nicht größer als 6 wird, müsste man die Bedingung wie folgt gestalten (natürlich kann man auch einen anderen Wert als 6 festlegen, je nachdem welche Stiftbreite man erlauben möchte):



Füge die Bedingungen für den *Plus*- und *Minus*-Button ein.

3.1 Noch mehr Farben! (Teil 1)

So kann es aussehen



Das soll passieren

Die Auswahl an Farben soll größer werden. Sorge dafür, dass eine Liste mit Farbnamen erscheint, aus der man die gewünschte Farbe auswählen kann. Dazu brauchst du Folgendes:

1. Eine Liste mit verschiedenen Namen von Farben.
2. In der App muss ein *ListPicker* angezeigt werden, auf den man klicken kann, um eine Farbe auszuwählen. Außerdem musst du ihm sagen, aus was (den Farben) ausgewählt werden soll. Dazu benötigst du folgenden Baustein:

```
when Farbe_auswählen ▾ .BeforePicking
do
```

Neue Komponenten

1 x *ListPicker*: Farbe wählen

Hinweis: Die Buttons zur Farbauswahl kannst du löschen oder zur Schnellauswahl behalten.

Tipps

- Listen kann man mit dem Baustein *make-a-list* erstellen.
- Wenn du mehr Felder in der Liste brauchst, klicke auf das Zahnrad im blauen Kasten des Bausteins.

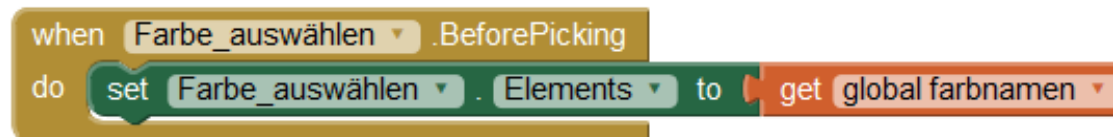
3.1 Noch mehr Farben! (Teil 1)

Lösung

Schritt 1 sollte folgende Blöcke liefern (es können natürlich auch andere Farben sein):

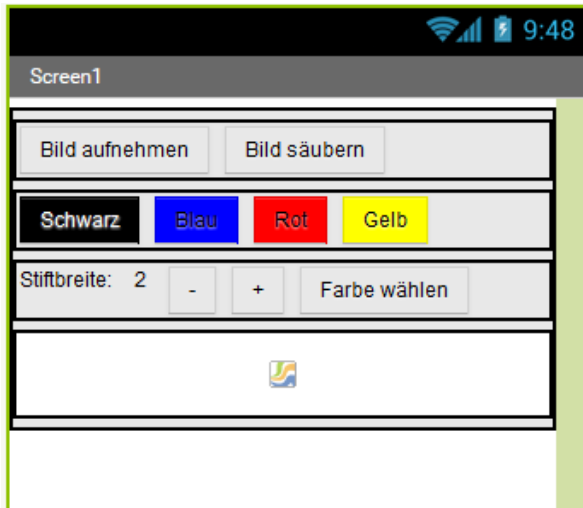


Schritt 2 sollte folgende Blöcke liefern:



3.2 Noch mehr Farben! (Teil 2)

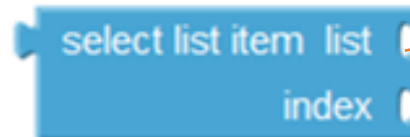
So kann es aussehen



Das soll passieren

Jetzt musst du nun noch dafür sorgen, dass sich auch die Stiftfarbe ändert, wenn man eine Farbe auswählt.

1. Du benötigst zuerst eine weitere Liste. Diesmal soll sie die ausgewählten Farb-Bausteine enthalten.
2. Damit die Farbe der Leinwand richtig gesetzt wird, brauchst du folgenden Baustein:



Aus der oben genannten Liste wird das Element mit dem Index (= Nummer) ausgewählt, den du unten angibst.

Neue Komponenten

Keine

Tipps

- Du musst die Farben in der Liste in der gleichen Reihenfolge anordnen wie die Farbnamen.
- Die Eigenschaft *SelectionIndex* vom *ListPicker* gibt an, welches Element vom Benutzer ausgewählt wurde.

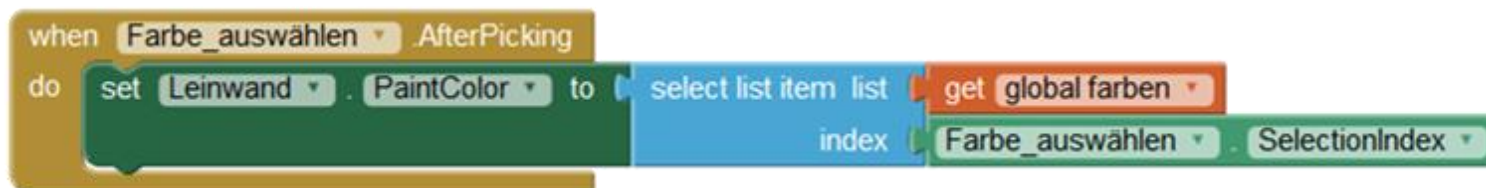
3.2 Noch mehr Farben! (Teil 2)

Lösung

Schritt 1 sollte folgende Blöcke liefern (es können natürlich auch andere Farben sein, sie müssen aber die gleiche Reihenfolge aufweisen wie die Farbnamen):



Schritt 2 sollte folgende Blöcke liefern:



3.3 Teile deine App mit der Welt

So kann es aussehen



Das soll passieren

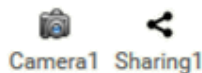
Damit auch deine Freundinnen und Freunde deine Meisterwerke bewundern können, benötigt deine App noch einen Button zum Teilen.

Füge einen solchen Button zu deiner App hinzu. Wenn man auf den *Teilen*-Button klickt, soll die Komponente *Sharing1* aufgerufen werden und eine Datei teilen. Zuerst muss dafür das *Canvas* (= Leinwand) aber noch unter einem beliebigen Dateinamen (name.jpg) abgespeichert werden.

Neue Komponenten

1 x Button *Teilen*
1 x *Sharing*

Non-visible components



Tipps

Auf deinem Smartphone wird automatisch eine Liste mit verschiedenen Apps aufgerufen, mit denen du das Bild verschicken kannst, beispielsweise WhatsApp und Gmail. Aber auch das Speichern in der Dropbox ist möglich, wenn du die entsprechende App installiert hast. Darum brauchst du dich nicht mehr kümmern.

3.3 Teile deine App mit der Welt

Lösung

