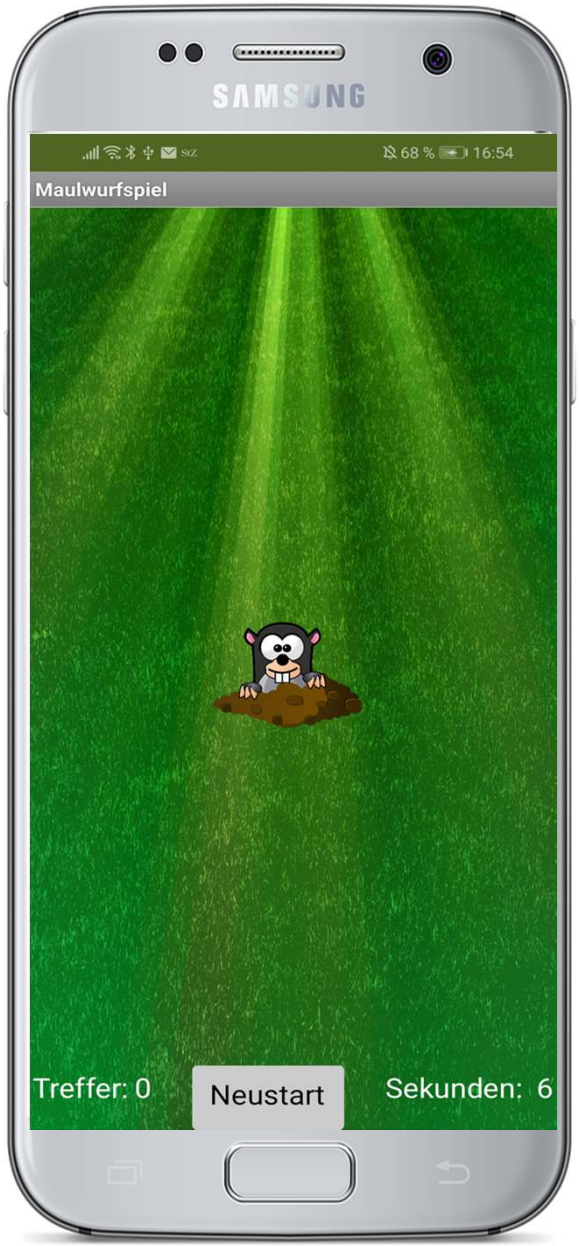
Projekt 3: Maulwurfspiel

Ziel

Wir möchten ein kleines Reaktionsspiel auf dem Handy.

Dazu erstellen wir eine App, auf der sich ein kleiner Maulwurf an verschiedenen Orten zeigt. Wenn man den Maulwurf mit dem Finger berührt bekommt man einen Punkt.

Später werden wir diese App noch erweitern und verschönern.



Teil 1: Grafik, Sound und Grundfunktion

Im ersten Teil programmieren wir unsere App so, dass die Grafiken für die Wiese und den Maulwurf auf dem Bildschirm dargestellt werden. Wenn wir den Maulwurf mit dem Finger berühren wird ein Sound abgespielt.

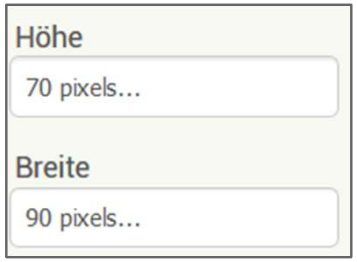
Anleitung (1)

1. Richte die Benutzeroberfläche wie folgt ein:
   a. Lege ein neues Projekt an. Nenne das Projekt *Maulwurf* oder so ähnlich.
   b. Setze den Titel von *Screen1* auf *Maulwurfspiel* und die *Bildschirmausrichtung* auf *Hochformat*.
   c. Lade die Datei ***Maulwurf.zip*** von www.markuskaupp.de/mobile-apps-ag und entpacke sie.
   d. Im Bereich *Medien* (unter den *Komponenten*) kann man Dateien hochladen. Lade die Sounddatei und die beiden Bilder aus der zip-Datei hoch.

   e. Ziehe aus der Palette eine *Zeichenfläche* auf das Handy im *Betrachter*. Setze die Höhe und die Breite der Zeichenfläche jeweils auf *Fülle alles*...

Anleitung (2)

2. Jetzt fügen wir die Bilder und den Sound dazu. Mache dazu folgendes:
   a. Wähle in den Komponenten den Screen1 auf und setze das *HintergrundBild* auf *wiese.jpg*. Setze die Hintergrundfarbe der *Zeichenfläche1* auf *Keine*.
   b. Im Bereich *Palette* gibt es unter *Zeichnen und Animation* die *ZeichenAnimation*. Ziehe eine *ZeichenAnimation* irgendwo auf die *Zeichenfläche1*.
   c. Benenne die *ZeichenAnimation1* um in *Maulwurf*. Passe die Eigenschaften von Maulwurf dann so um:
      - Höhe: 70 Pixel
      - Breite: 90 Pixel
      - Bild: Maulwurf.png



   d. Im Bereich *Palette* gibt unter *Medien* die *Tonwiedergabe*. Ziehe eine *Tonwiedergabe* auf das Handy. Sie erscheint dann unter dem Handy bei *nicht sichtbare Komponenten*.
   e. Benenne *Tonwiedergabe1* um in *TrefferTon*. Setze die Quelle von *TrefferTon* auf *treffer.wav*.

Anleitung (3)

3. Schreibe jetzt den Code, der den Sound abspielt, wenn der Maulwurf angetippt wird. Beim Antippen wird das Ereignis *DrueckeRunter* in der ZeichenAnimation Maulwurf ausgelöst. Der Ton soll als Reaktion auf dieses Ereignis abgespielt werden. Das Ganze soll dann so aussehen:

4. **Fertig!** Lade die App herunter und teste sie..

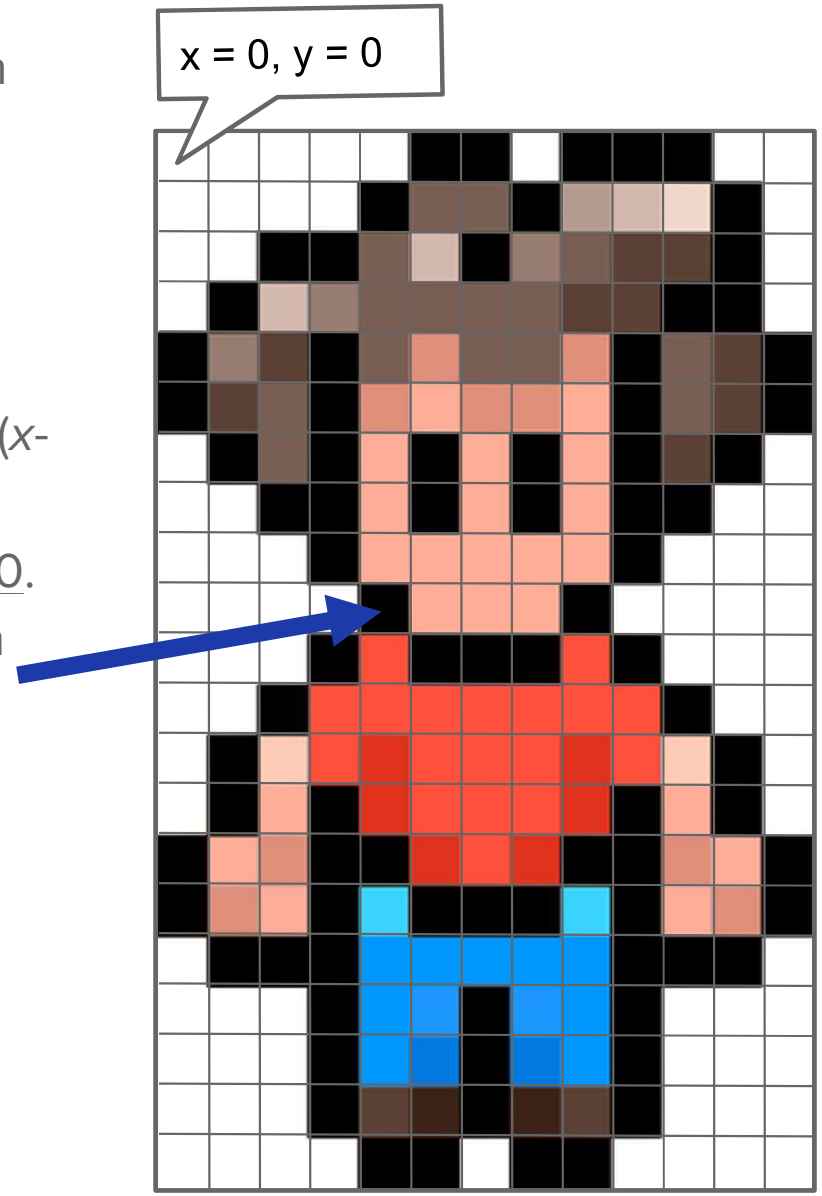
Teil 2: Zufällige Bewegung

Jetzt programmieren wir den Maulwurf so, dass er ständig die Position wechselt. Er soll sich immer nach einer festen Zeit bewegen und außerdem auch, wenn er gefangen wurde.

Gut zu wissen: Bildschirm und Pixel

Bildschirme bestehen aus einzelnen Punkten, die in allen Farben leuchten können. Diese Bildpunkte nennt man **Pixel**.

Die Pixel sind in einem Raster (also tabellenförmig) angeordnet. Jedes Pixel des Bildschirms hat eine Position. Die Position ist gegeben durch die Spalte (*x*-Position) und die Zeile (*y*-Position). Man zählt die Zeilen und Spalten von links oben und beginnt bei 0. Bei dem Männchen ist z.B. das Pixel an der Position *x = 4*, *y=9* schwarz.

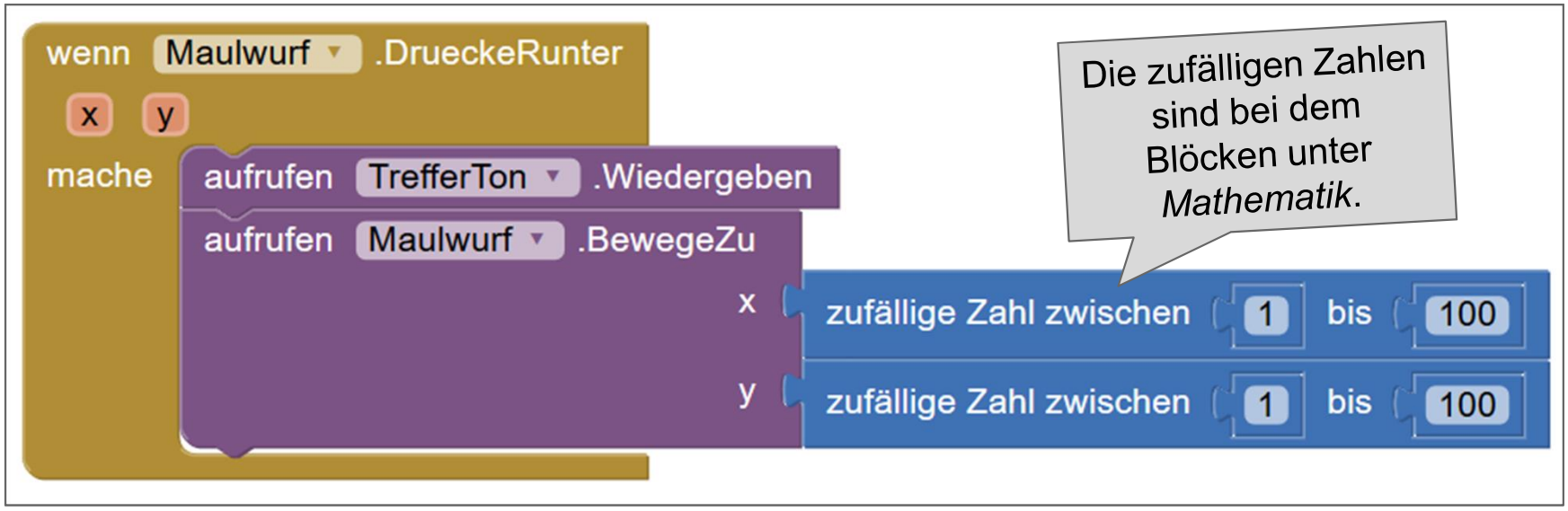


Die Anzahl der Spalten und Zeilen der Pixel ist die **Auflösung** des Bildschirms. Höhere Auflösung bedeutet mehr Pixel. Mehr Pixel sorgen für ein schärferes Bild.

Mein Handy hat eine Auflösung von 1.080 x 2.340 Pixel.

Anleitung (1)

1. Wähle in den Komponenten den Maulwurf aus und setze die Eigenschaften *X* und *Y* jeweils auf 0. Die linke obere Ecke des Maulwurfbilds ist nun in der linken oberen Ecke des Bildschirms.
2. Wechsle in die Ansicht *Blöcke*.
3. Der Maulwurf soll sich jetzt immer an eine zufällige Stelle bewegen, wenn er erwischt wurde. Dazu bewegen wir den Maulwurf an eine Position mit zufälligem *x*- und *y*-Wert.



Anleitung (2)

4. Bis jetzt würde unser Maulwurf immer nur in den *X*- und *Y*-Bereichen von 1 bis 100 platziert. Unser Bildschirm ist aber größer. Weil Handys unterschiedliche Auflösungen haben, wissen wir aber nicht, welche Positionen erlaubt sind.

   Es gibt aber Blöcke, die uns die Höhe und die Breite der Zeichenfläche liefern. Ersetze die Obergrenze für die Zufallszahlen also durch die Breite bzw. die Höhe der Zeichenfläche. Dann nutzt der Maulwurf den ganzen Bildschirm aus.

Anleitung (3)

5. Der Maulwurf wechselt jetzt schon immer die Position, wenn wir ihn antippen. Er soll sich aber zusätzlich immer nach einer bestimmten Zeit bewegen. Baue dafür einen Timer ein, und zwar so:
    a. Wechsle in den Designer
    b. Ziehe im Bereich *Palette* aus dem Abschnitt *Sensoren* eine *Uhr* auf das Handy im Betrachter. Die *Uhr1* erscheint bei den *nicht sichtbaren Komponenten*.
    c. Benenne *Uhr1* um in *MaulwurfUhr* und setze bei den Eigenschaften das *ZeitgeberIntervall* auf *800*. Der Wert ist die Zeit, nach der sich der Timer meldet, in Millisekunden. Unser Wert entspricht also 0,8 Sekunden.
    d. Wechsle in die Ansicht *Blöcke* und füge einen Block ein, der auf das Ereignis *MaulwurfUhr.Zeitgeber* reagiert. Da unsere Uhr sich alle 0,8 Sekunden meldet, wird dieses Ereignis auch alle 0,8 Sekunden ausgelöst.
    e. Dupliziere den *Maulwurf.Bewegezu*-Block aus dem *DrueckeRunter*-Ereignis und füge ihn ein.

Anleitung (4)

5. **Fertig!** Lade die App runter und versuche den Maulwurf zu fangen. Falls es dir zu schnell geht, kannst du das *ZeitgeberIntervall* auch erhöhen.

Teil 3: Punktezähler und Zeit

Die App soll jetzt so erweitert werden, dass die Anzahl der Treffer und die Zeit angezeigt wird.

Anleitung (1)

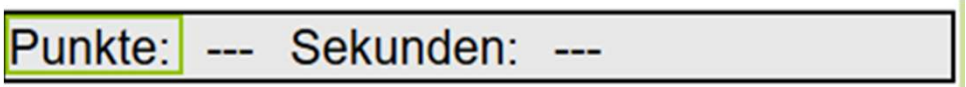
1. Im ersten Schritt platzieren wir Bezeichnung-Elemente auf dem Bildschirm. Da schreiben wir später die Zeit und die Punkte rein. Gehe dazu wie folgt vor:

   a. Platziere eine *HorizontaleAusrichtung* (aus der *Palette*, Abschnitt *Anordnung*) unter der Zeichenfläche. DIe Zeichenfläche wird dadurch nach oben geschoben.

   b. Setze die Breite der *HorizontaleAusrichtung* auf *Fuelle alles...*

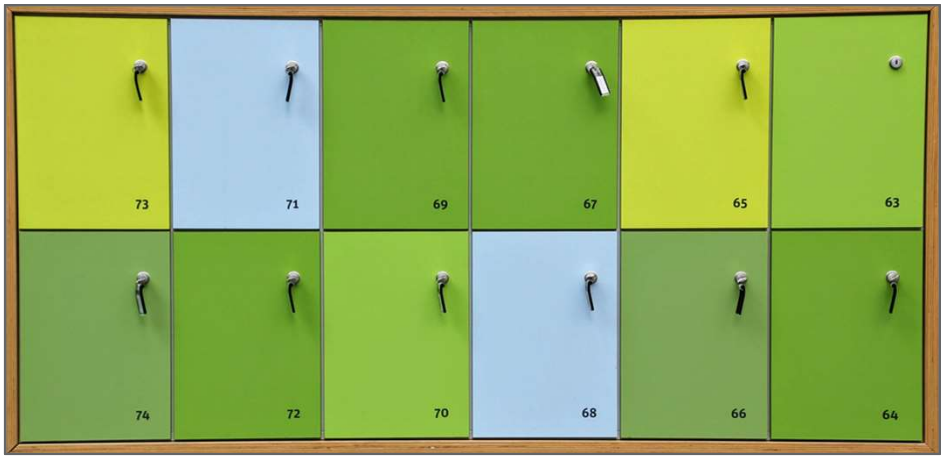
   c. Füge in die *HorizontaleAusrichtung* vier *Bezeichnung*-Elemente ein.

   d. Benenne das zweite und das vierte Bezeichnungselement um in *Punkte* bzw. *Sekunden*. Setze den Text dieser beiden Bezeichnungen jeweils auf ---.

   e. Setze den Text der verbleibenden beiden Bezeichnungen sinnvoll. Es könnte z.B. so aussehen. 

   f. Setze die *Schriftgröße* für alle vier Bezeichnungen auf *18* und die *TextFarbe* auf *weiß*.

Gut zu wissen: Variablen



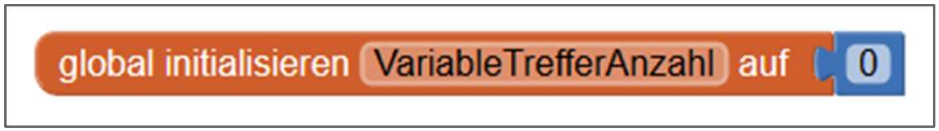
In Computer-Programmen und -Apps muss man sich oft Werte merken. Deswegen haben Computer und Handys **Hauptspeicher** (engl. **RAM**). Dort können die Programme ihre Daten speichern.

Den Hauptspeicher kann man sich vorstellen, wie einen Schließfachschrank mit nummerierten Schließfächern; nur dass in den Fächern halt Zahlen abgelegt werden können.

Wenn man sich beim Programmieren einen Wert merken möchte, legt man eine **Variable** an. Jede Variable belegt ein oder mehrere Speicherfächer. Große Zahlen brauchen mehr Platz, kleine weniger.

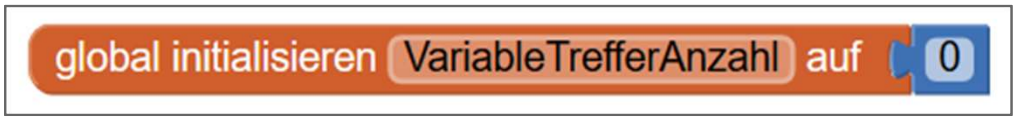
Jede Variable hat auch einen **Variablennamen**. Über diesen Namen können wir sie einfach ansprechen.

Beispiel: Anlegen der Variablen mit dem Namen *VariableTrefferAnzahl*.

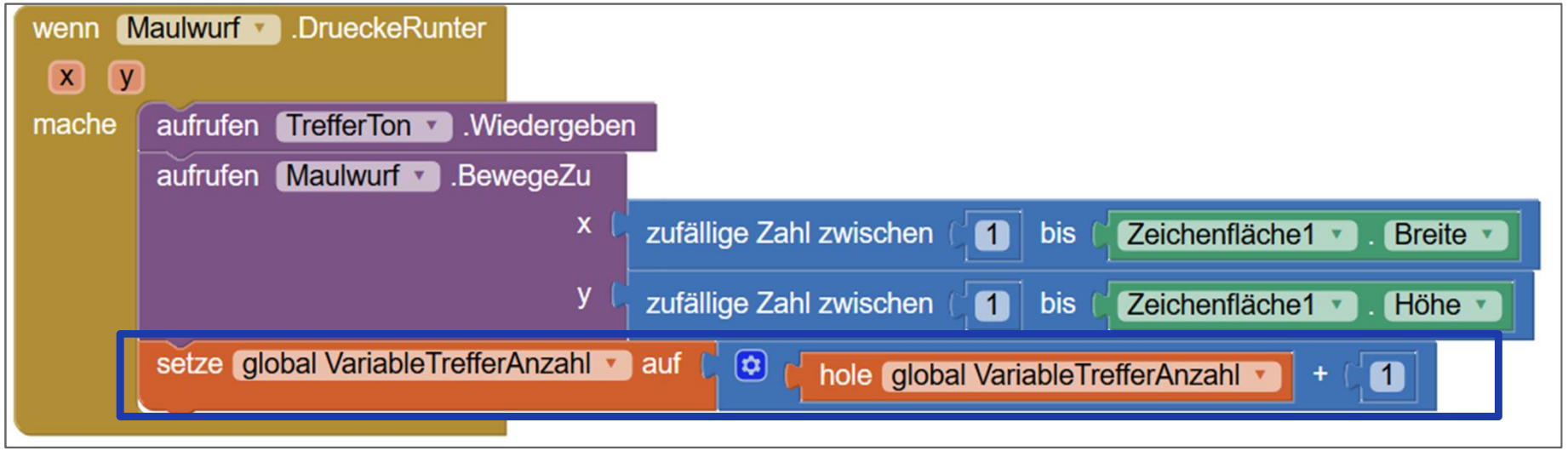


Anleitung (2)

2. Lege eine neue Variable an, in der wir uns die Anzahl der Treffer merken können. Den entsprechenden Block findest du bei den Blöcken im Abschnitt *Variablen*. Weil wir am Anfang noch Null Treffer haben, schreiben wir in die Variable den Wert 0.



3. Jetzt zählen wir zu dem Wert in der Variablen immer eins dazu, wenn der Maulwurf getroffen wird.



Anleitung (3)

4. Schau dir die eingefügten Blöcke nochmal genau an. Das werden wir jetzt noch öfter so oder so ähnlich machen. Der Block macht nacheinander drei Sachen:

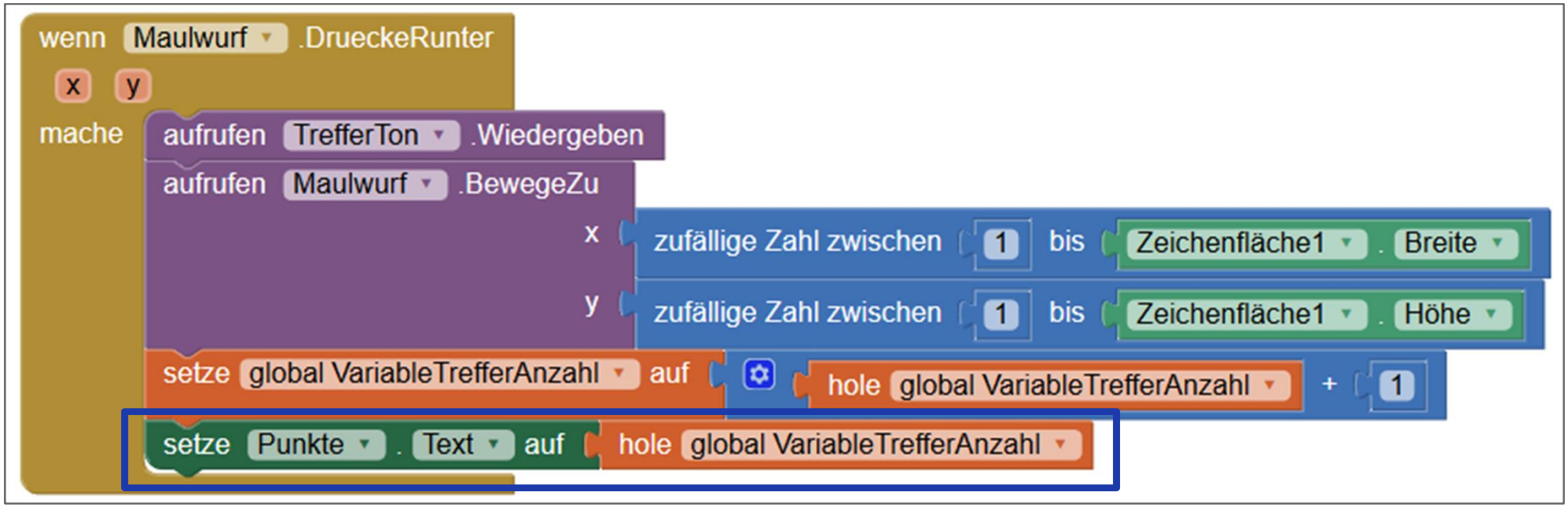
   - Der Block 

holt erstmal den aktuellen Wert der Variablen aus dem Speicher. Beim ersten Aufruf steht da eine 0 drin.
   - Zu dem gelesenen Variablenwert addieren wir mit dem blauen Block den Wert 1 dazu. 

   - Das Ergebnis der Addition schreiben wir jetzt mit dem orangen setze-Block wieder in die Variable. Der Wert, der in der Variablen steht, ist jetzt um 1 höher als vorher.

Anleitung (4)

5. Das Zählen der Treffer funktioniert jetzt schon. Wir können den Wert aber bisher noch nirgends sehen. Zeige deshalb jetzt den neuen Wert der Variablen in dem passenden *Bezeichnung*-Element an:



6. Lade die App herunter und probiere sie aus. Jetzt bekommst du die Treffer angezeigt.

Anleitung (5)

7. Jetzt soll zusätzlich auch noch die Zeit hochzählen. Dazu brauchen wir eine zweite Uhr und eine zweite Variable. Mache also folgendes:
    a. Ziehe in der Designer-Ansicht eine zweite Uhr auf das Handy. Nenne die Uhr *SekundenUhr* und lasse das *ZeitgeberIntervall* auf *1000*.
    b. Lege in der *Blöcke*-Ansicht eine neue Variable an. Sie soll *SekundenZähler* heißen und beim Start der App den Wert 0 bekommen.
    c. Immer wenn das Ereignis *SekundenUhr.Zeitgeber* auftritt soll der Wert der Variablen *SekundenZähler* im 1 erhöht werden. Außerdem soll der neue Wert in das Bezeichnung-Element *Sekunden* geschrieben werden.
8. **Fertig!** Die App zählt jetzt Treffer und Sekunden. Viel Spaß beim Ausprobieren.

Teil 4: Spielende und Neustart

Die App soll so erweitert werden, dass man pro Runde nur noch 20 Sekunden Zeit hat. Dann soll der Text **GAME OVER** erscheinen.

Mit einem Button soll man ein neues Spiel starten können.

Anleitung (1)

1. Um das Spiel nach 20 Sekunden zu beenden, schauen wir uns einfach jedes Mal, wenn sich die *SekundenUhr* meldet, ob der *SekundenZähler* schon bei 20 ist. Das kannst du wie folgt machen:

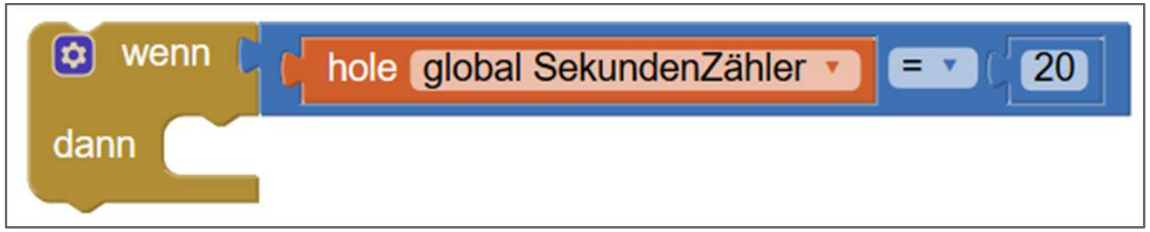
   a. Baue in das Ereignis *SekundenUhr.Zeitgeber* einen **wenn-dann-Block** ein. Der Block prüft, ob eine bestimmte Bedingung erfüllt ist. Nur wenn die Bedingung erfüllt ist, führt er seine Blöcke aus.

   b. Stöpsle beim *wenn*-Teil des *wenn-dann*-Blocks einen *Vergleich* an. Der Vergleich sagt uns, ob zwei Zahlen gleich sind.

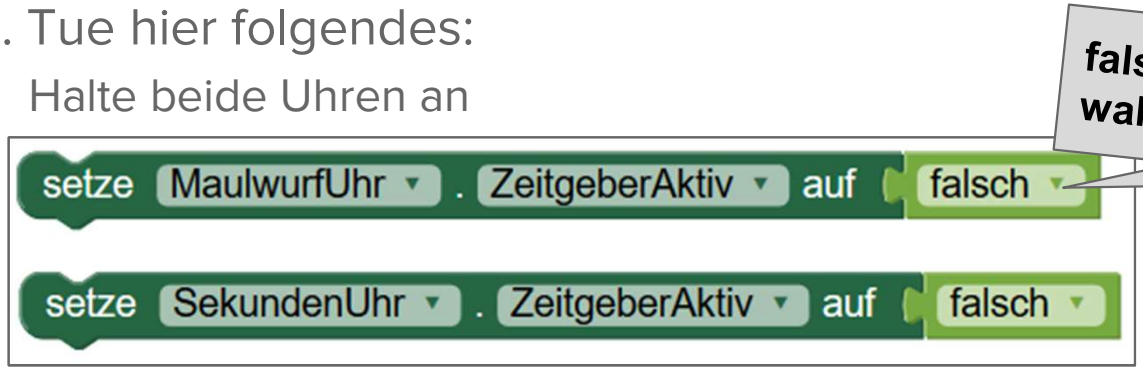
   

   c. Wir wollen schauen, ob die Variable *SekundenZähler* schon gleich 20 ist. Deswegen sollte das alles zusammen so aussehen:

Anleitung (2)

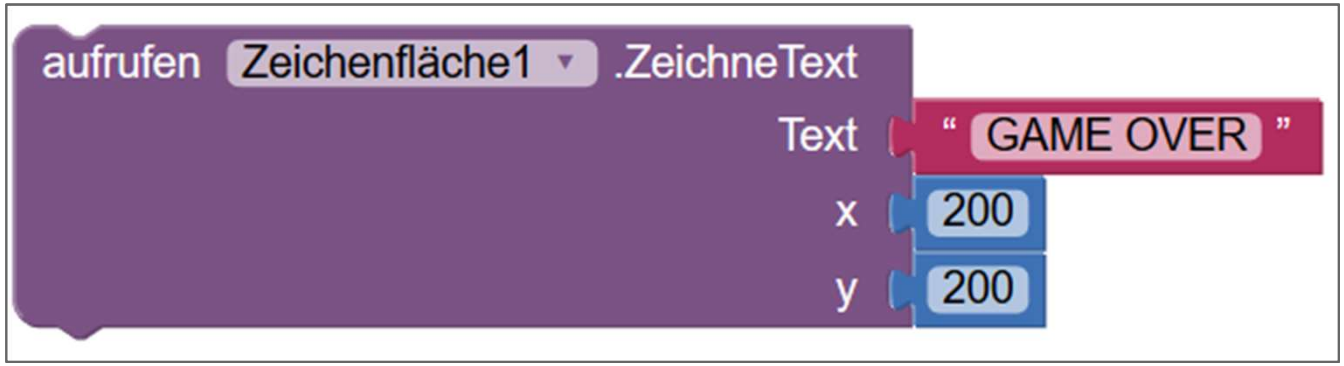
2. Jetzt müssen wir nur noch sagen, was nach den 20 Sekunden getan werden soll. Tue hier folgendes:
   a. Halte beide Uhren an



falsch bedeutet hier "*ausgeschaltet*".
wahr bedeutet "*eingeschaltet*".

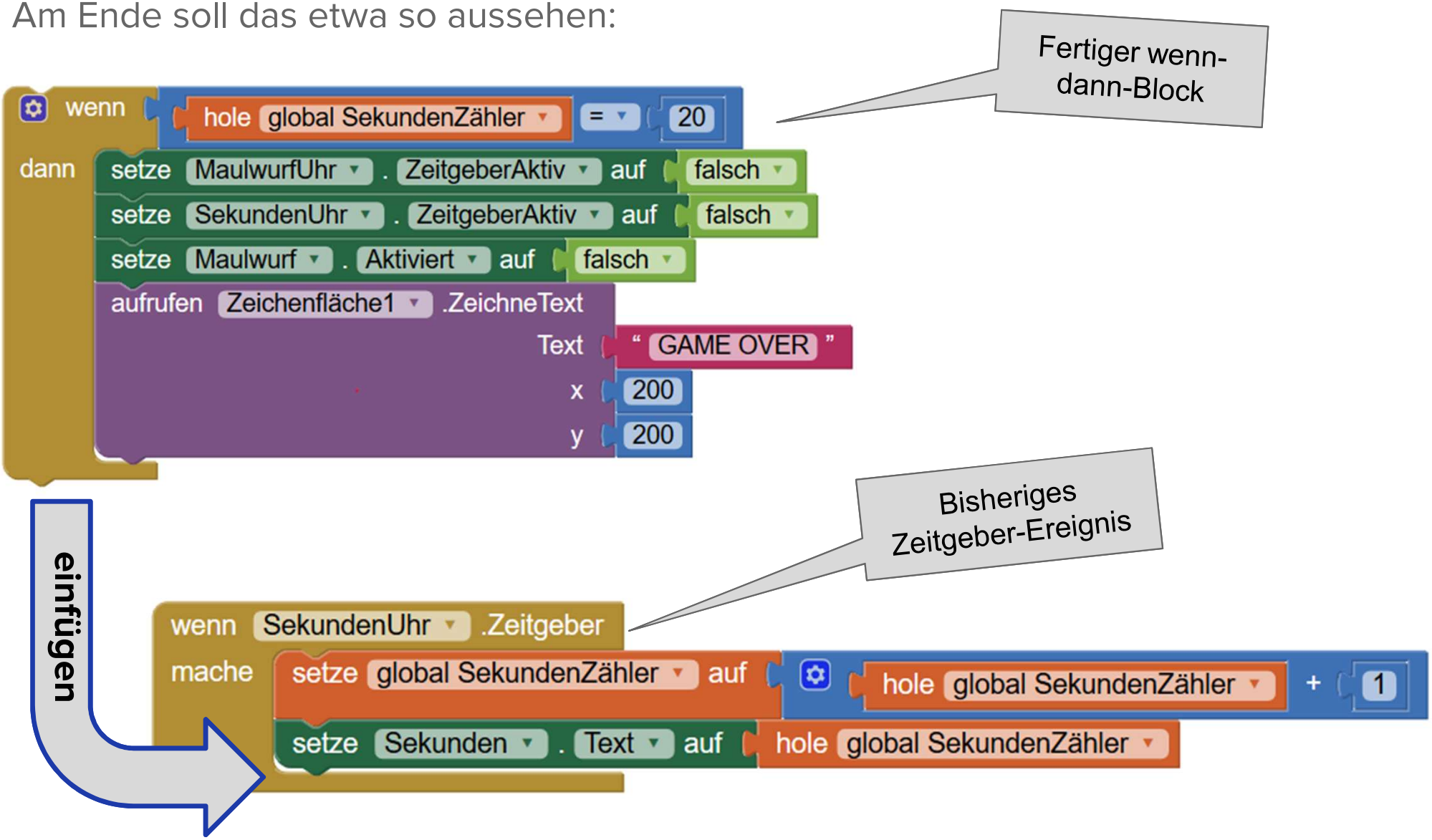
   b. Deaktiviere den *Maulwurf*, damit er nicht mehr gedrückt werden kann.

   c. Schreibe den Text *Game Over* auf die Zeichenfläche. Die Textgröße und -farbe kannst du bei den Eigenschaften der Zeichenfläche im *Designer* einstellen.



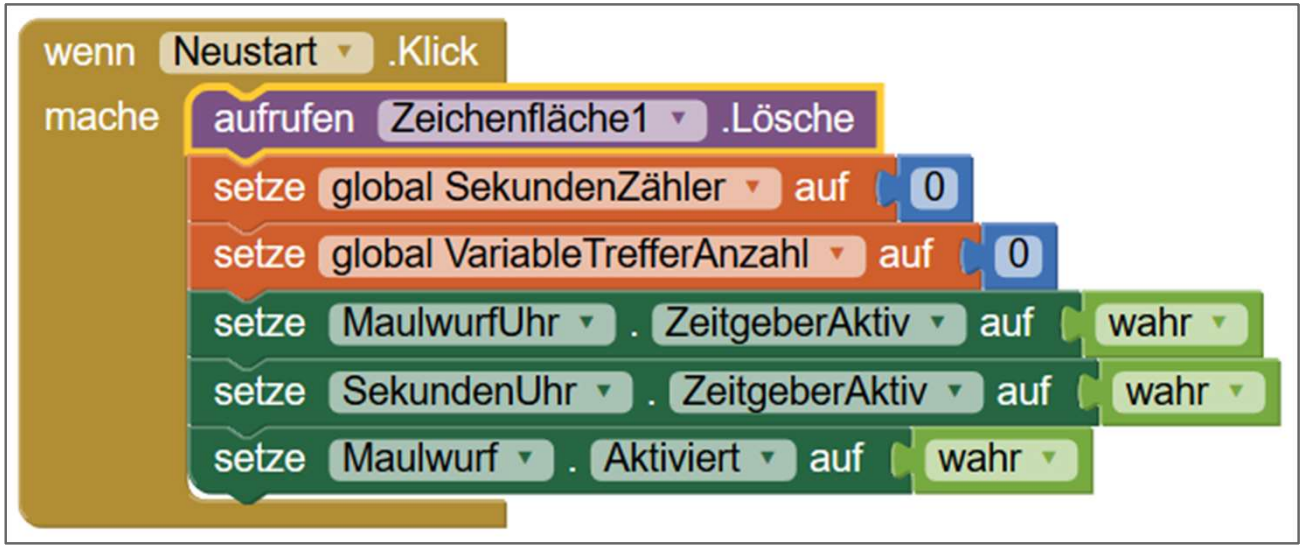
Anleitung (3)

Am Ende soll das etwa so aussehen:



Anleitung (4)

3. Zum Schluss bauen wir jetzt noch einen Button ein, mit dem wir das Spiel neu starten können. Das geht so:
   a. Füge im Designer zwischen den Bezeichnungselementen für die Punkte und die Zeit einen Button ein.
   b. Nenne den Button *Neustart*. Setze seine Breite auf Fuelle alles… und seinen Text auf *Neues Spiel*
   c. Wechsle in die Ansicht *Blöcke*.
   d. Wenn der Button *Neustart* geklickt wird, müssen wir die Zeichenfläche mit dem *Game Over* wieder löschen, die Zeit und die Treffer auf 0 setzen und die Uhren und den Maulwurf wieder aktivieren



Anleitung (5)

4. **Fertig!** Lade die App herunter und führe sie aus.

Erweiterungen

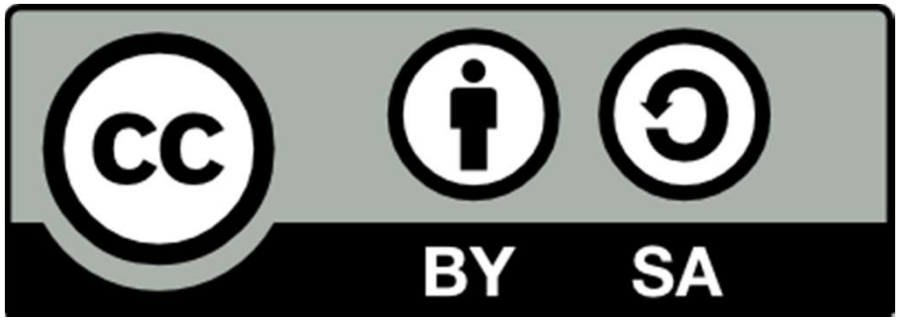
Funktionalität

- Füge noch einen zweiten Maulwurf ein. Verwende dazu ein weiteres *ZeichenAnimation*-Objekt mit dem Maulwurfbild.
- Füge einen Igel ein. Er soll sich verhalten wie ein Maulwurf. Wenn man ihn anklickt, gibt es aber Abzug. Ein passendes Bild findest du auf Google oder in der Bilddatenbank www.pixabay.com.
- Mache das Verhalten des Maulwurfs unberechenbarer. Dazu kannst du im *Zeitgeber*-Ereignis das *ZeitgeberIntervall* der *MaulwurfUhr* auf einen Zufallswert setzen.

Design

- Du kannst die App schnell verändern, wenn du die Bilder austauscht. Überlege dir ein anderes Tier und einen anderen Hintergrund (z.B. eine Maus in einem Käse). Schau, ob du in der Bilddatenbank www.pixabay.com passende Bilder für deine Idee findest und binde die Bilder ein.
- Suche andere Sounds, die zu deiner neuen App passen und binde sie ein.
- Suche eine Sounddatei mit einem Applaus und spiele den Applaus ab, wenn das Game Over erscheint.

Lizenz



Urheber: Markus Kaupp