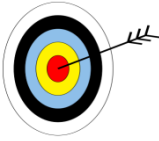



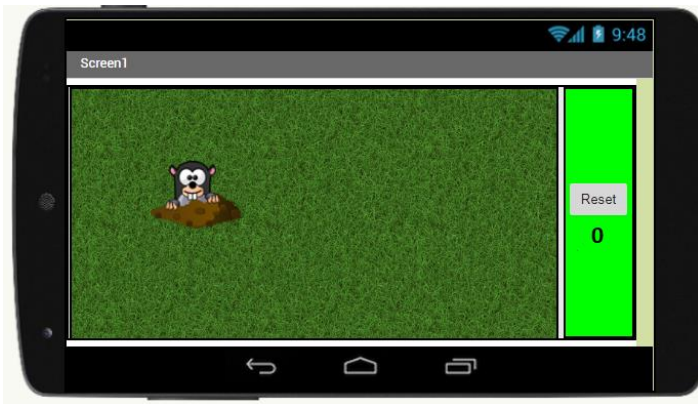


## Blatt 4a – MoleMash



Ihr habt euch also für **MoleMash** entschieden. Dies ist ein lustiges Spiel, bei dem es darum geht, den **Maulwurf**  zu fangen, der an verschiedenen Positionen des Bildschirms erscheint. Dieses Arbeitsblatt wird euch dabei helfen, eine App zu erstellen, die...

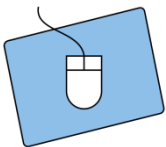
- einen  auf einem Spielfeld erscheinen lässt.
- es dem Spieler ermöglicht, den  durch Anklicken zu fangen.
- Punkte hochzählt, wenn dies geschafft wurde.



Schwierigkeit: ★★

## Anlegen des Projekts

Wie bei jeder neuen App müsst ihr zunächst ein neues Projekt anlegen.



- 1.) Erstellt jetzt euer Projekt, und gebt ihm einen ansprechenden Namen.
- 2.) Verbindet den App Inventor wieder, wie auf dem ersten Arbeitsblatt beschrieben, mit dem Smartphone/Tablet oder dem Emulator.

## Der Aufbau eurer App

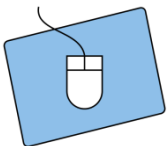
Als Nächstes müsst ihr überlegen, wie eure App aussehen soll. Eure App braucht Platz für die folgenden Dinge (die ihr in den nächsten Abschnitten der Reihe nach einbauen werdet):

- eine **Spielfläche**, auf der sich der Maulwurf bewegen kann,
- einen **Reset-Knopf**, mit dem ihr das Spiel neuanfangen könnt, und
- eine **Punkteanzeige**, die eure aktuellen Punkte hochzählt.

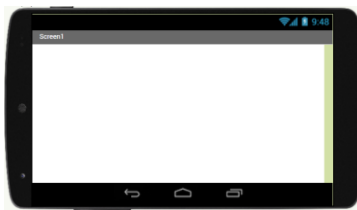
Das sind schon recht viele Dinge. Damit diese alle Platz haben, sollt ihr zunächst die Ausrichtung eurer Screens von **Unspecified** in **Landscape** ändern:



Mit **Landscape** ist bei Apps eine bestimmte Ausrichtung des Bildschirms gemeint, nämlich die horizontale Ausrichtung. Möchte man eine vertikale Ausrichtung haben, dann wählt man **Portrait**.



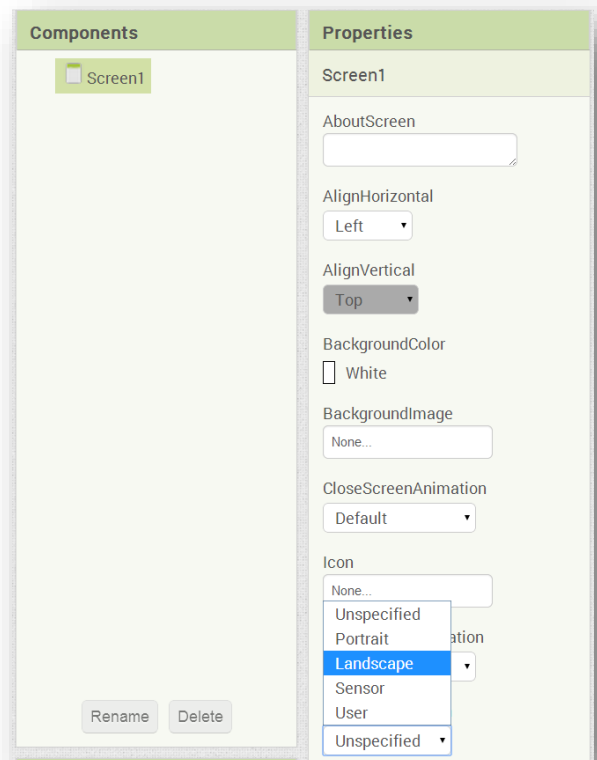
- 1.) Wählt euren *Screen1* unter Components aus.
- 2.) Sucht unter Properties den Eintrag *ScreenOrientation*.
- 3.) Ändert den Wert von *Unspecified* in *Landscape*.



Landscape



Portrait



### Die Bestandteile eurer App

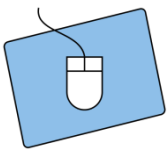
Nun könnt ihr anfangen, die einzelnen Elemente eurer App hinzuzufügen und diese mit Hilfe der vorgestellten **Screen Arrangements** sinnvoll anzuordnen.

Für die **Spielfläche** braucht ihr ein **Canvas**. Dieses findet ihr in der *Palette* unter *Drawing and Animation*.

Ein Canvas ist eine Leinwand. Auf dieser können sich Sachen bewegen, also perfekt für den Maulwurf.



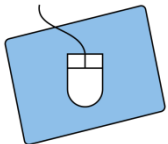
- Für den **Reset-Knopf** braucht ihr einen **Button** mit dem Text „Reset“.
- Für die **Punkteanzeige** braucht ihr ein **Label** mit dem Text „0“.



- 1.) Macht euch Gedanken darüber, wie ihr die Elemente positionieren wollt.
- 2.) Zieht die Screen Arrangements, die ihr für euer Layout braucht, in die App hinein.
- 3.) Fügt die Bestandteile (Canvas, Button und Label) eurer App hinzu.

### Der erste Test

Nachdem ihr die Bestandteile sinnvoll angeordnet habt, könnt ihr direkt einmal testen, wie das Ganze auf eurem Smartphone/Tablet bzw. im Emulator aussieht.



- 1.) Startet dazu die App auf dem Smartphone/Tablet oder im Emulator, wenn ihr dies noch nicht gemacht habt.
- 2.) Gefällt euch die Anordnung? Wenn ja, könnt ihr weitermachen, ansonsten überarbeitet sie einfach nochmal.



Lasst die App einfach auf dem Smartphone/Tablet oder im Emulator laufen. Der App Inventor aktualisiert alle Änderungen, die ihr vornehmt, und zeigt sie euch direkt an, ohne dass ihr die App neustarten müsst.

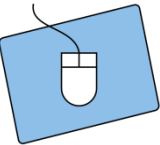
### Umbenennen nicht vergessen ;)

Da ihr jetzt die ersten Bestandteile eurer App fertig habt, ist es an der Zeit, diese wieder mit sinnvollen Namen zu versehen, damit ihr sie im Blocks-Editor besser unterscheiden könnt.

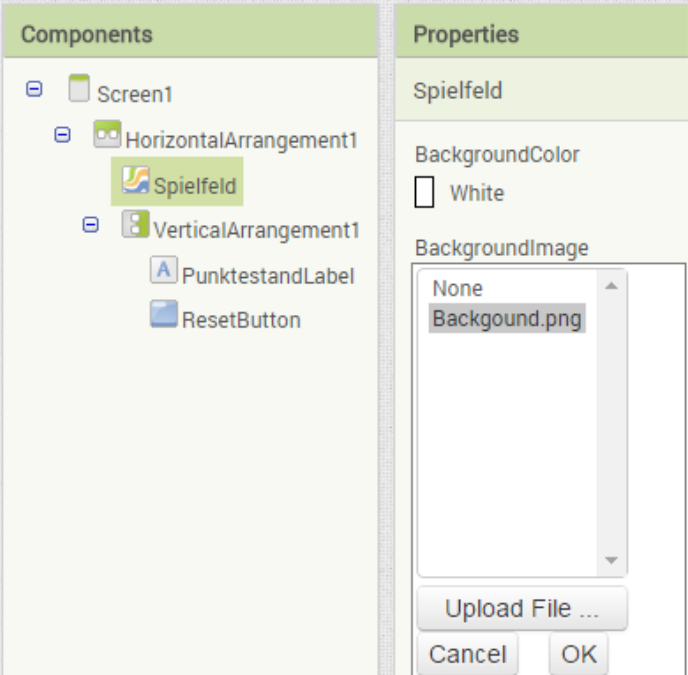
### Ein Hintergrund für eure Spielfläche

Damit der Maulwurf sich auf der Spielfläche wohlfühlt, soll diese eine Wiese als Hintergrund bekommen. Ladet dazu einen (von drei möglichen) Hintergründen aus dem Materialien-Ordner hoch.

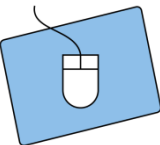
Anschließend müsst ihr eurem Canvas den Hintergrund nur noch zuweisen:



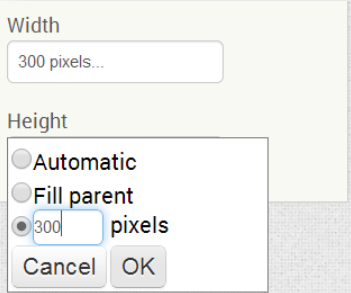
- 1.) Wählt das Canvas unter Components aus.
- 2.) Klickt unter Properties auf *BackgroundImage*, und weist dem Canvas das **background.png** als Hintergrundbild zu.



Wie wir euch bereits erklärt haben, müsst ihr beim App Inventor manchmal die Größe per Hand einstellen.

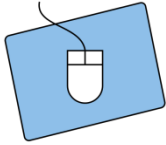


- 1.) Ihr habt immer noch euer Canvas ausgewählt.
- 2.) Sucht in den Properties nach **Width** (Breite) und **Height** (Höhe), und setzt diese jeweils auf 300 Pixel.
- 3.) Testet jetzt eure App, und ändert gegebenenfalls die Anzahl der Pixel, damit es auf dem Ausgabe-Medium gut aussieht.



### Der Maulwurf

Als Nächstes soll der Maulwurf auf die Wiese.

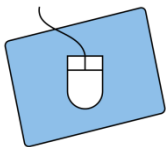


- 1.) Wechselt in der Palette zu *Drawing and Animation*.
- 2.) Zieht ein **ImageSprite** direkt auf die Wiese (euer Canvas).
- 3.) Ändert den Namen des ImageSprites.
- 4.) Ladet nun aus demselben Ordner den Maulwurf hoch, und weist das Bild dem ImageSprite zu.



Ein **ImageSprite** ist ein besonderes Bild. Dieses kann sich, im Gegensatz zu einem normalen Image, auf einem Canvas bewegen.

Außerdem hat euer Maulwurf eine Position auf der Wiese. Diese könnt ihr in den Properties als **x- und y-Koordinaten** ablesen bzw. verändern.



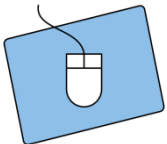
- 1.) Wählt den Maulwurf unter Components aus.
- 2.) Ändert unter Properties die x- und y-Koordinaten, und sucht euch eine schöne Position für euren Maulwurf aus.

## Ein Timer für den Maulwurf

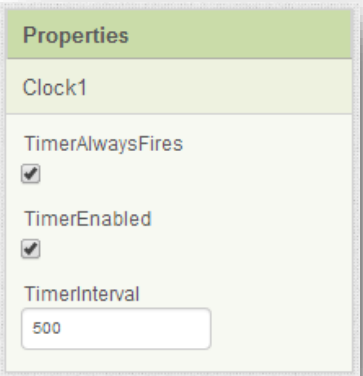
Als letzten Bestandteil eurer App braucht ihr eine **Clock (Uhr)**. Diese braucht ihr, damit sich der Maulwurf bewegen kann. Die Clock findet ihr in der Palette unter **Sensors**.



Eine **Clock** ist für viele Sachen gut, z. B. könnt ihr ein **TimerInterval** einstellen. Dieses löst nach Ablauf der eingestellten Zeit immer wieder die Funktion **Clock.Timer** aus. Das ist hilfreich, wenn sich der Maulwurf nach einer gewissen Zeit an eine neue Position bewegen soll.



- 1.) Zieht eine **Clock** in eure App.
- 2.) Wählt diese unter Components aus, und ändert den Wert für *TimerInterval* in 500. (500 Millisekunden entsprechen 0,5 Sekunden.)



## Ein kleines Zwischenfazit

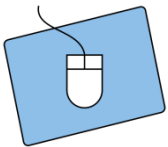
So oder so ähnlich sollte eure App jetzt aussehen. Falls ihr noch Fragen habt, spricht einfach kurz mit einem Betreuer oder einer Betreuerin.



## Die Bewegung des Maulwurfs

Als Erstes soll sich der Maulwurf zufällig über das Spielfeld bewegen. Dazu müssen zwei Dinge umgesetzt werden:

- 1.) die Bewegung des Maulwurfs an eine **zufällige** neue Position,
- 2.) das **Zeitintervall**, das vorgibt, wann diese Bewegung ausgelöst wird.



- 1.) Wechselt in den Blocks-Editor.
- 2.) Wählt bei eurem **Maulwurf** den Block **call Maulwurf.MoveTo** aus.

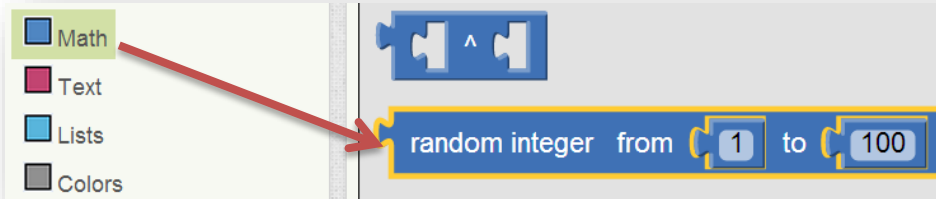


- 3.) Dieser Block bewegt euren Maulwurf an neue **x- und y-Koordinaten**, die ihr später an diesen Block anhängen werdet.
- 4.) Um die Bewegung auszulösen, braucht ihr die **Uhr**. Benutzt deren Funktion **when Uhr.Timer**, die alle 500 ms (also so, wie ihr es eingestellt habt) aufgerufen wird.
- 5.) Kombiniert beide Blöcke.
- 6.) Vorsicht! Nicht ausprobieren! Ihr habt noch offene Puzzle-Teile. Damit es richtig funktioniert, fehlen euch nämlich noch die **x- und y-Koordinaten** der neuen Position. Diese fügt ihr auf der nächsten Seite hinzu.



## Blatt 4a – MoleMash

Die neue Position (x, y) des Maulwurfs soll **zufällig** sein. Um das erreichen zu können, besitzt der App Inventor die Möglichkeit, eine **Zufallszahl** zu generieren. Diese findet ihr im **Blocks-Editor** unter **Built-In** → **Math**.

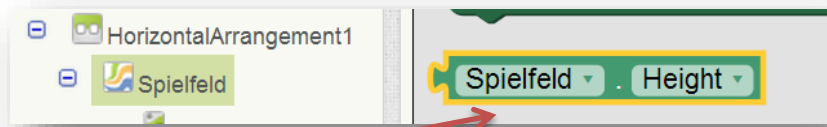


Die Funktion wählt eine **zufällige ganze Zahl** im Bereich von **a** bis **b** aus, hier von 0 bis 100.

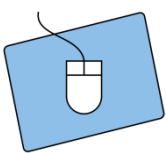
**Der Bereich der Zufallszahlen**

Euer Bereich für die Zahlen soll so groß sein wie die Wiese.

Die **Größe der Wiese** erhaltet ihr folgendermaßen:



**Spielfeld.Height** gibt euch die Höhe (y-Koordinate) zurück, und **Spielfeld.Width** liefert euch die Breite (x-Koordinate).

**So sehen die Blöcke kombiniert aus (für die Höhe)**

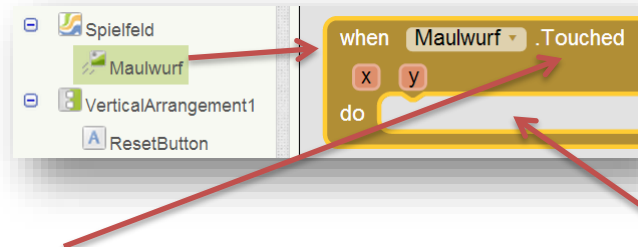
- 1.) Erstellt nun die zufällige Bewegung für die Höhe.
- 2.) Benutzt das Wissen aus den vorherigen Schritte, um die Funktion mit den fehlenden **x- und y-Koordinaten** zu erweitern.
- 3.) Testet die Bewegung! 😊

*Super! Bis hier ist es schon ein ganzes Stück Arbeit gewesen. Jetzt kommt der Endspurt ;)*

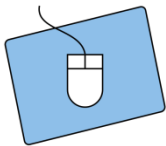


## Jetzt den Maulwurf fangen

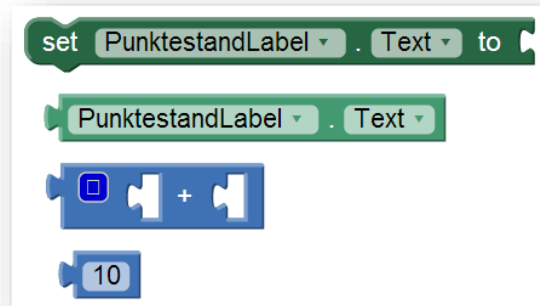
Um den Maulwurf zu fangen, muss man ihn antippen, bevor er sich wieder bewegt hat. Ob man das geschafft hat, lässt sich mit folgender Funktion feststellen:



Wenn der Maulwurf **berührt** wurde, dann wird der Code im Bereich **do** ausgeführt. Das Ziel ist es, den Maulwurf zu treffen und dadurch bspw. **10 Punkte hochzuzählen**. Dazu braucht ihr die Bausteine im folgenden Kasten:



- 1.) Sucht die Bausteine raus.
- 2.) Macht euch Gedanken, wie ihr sie kombinieren könnt.
- 3.) Fügt sie in die Funktion **when Maulwurf.Touched** ein.
- 4.) Testet euer Programm.



## Der Reset-Knopf

Als Letztes müsst ihr dafür sorgen, dass das Spiel **zurückgesetzt** wird. Dazu braucht ihr den **Reset-Knopf**.

Wenn der Reset-Knopf gedrückt wird, dann sollen die Grundeinstellungen wiederhergestellt werden.

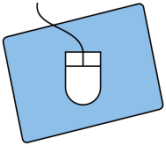


### Die Grundeinstellungen

Jetzt müsst ihr euch kurz überlegen, was die Grundeinstellungen sind. Dazu habt ihr folgende Hinweise:

- 1.) Der **Punktestand** hat eine Grundeinstellung.
- 2.) Die **Position des Maulwurfs** kann (muss aber nicht) auf den Startwert zurückgesetzt werden.

## Blatt 4a – MoleMash



- 1.) Überlegt euch, was alles zurückgesetzt werden muss.
- 2.) Setzt eure Ideen in der Funktion **when ResetButton.Click** um.



Gratulation ☺

*Damit habt ihr ein voll funktionsfähiges MoleMash programmiert. Auf der nächsten Seite findet ihr noch Tipps, Hinweise und Anregungen, wie ihr das Spiel erweitern könnt. Ihr könntet bspw. alle drei Hintergründe und Maulwürfe einbauen.*

*Falls ihr lieber ein neues Spiel programmieren wollt, dann wendet euch an die Betreuerinnen und Betreuer.*

## Erweiterungen

Hier sind einige Ideen, um das Spiel noch zu erweitern:

### **Mehrere Maulwürfe und Hintergründe zur Auswahl**

Um diesen Effekt zu erzielen, könnt ihr für jeden Hintergrund und jeden Maulwurf einen eigenen Button anlegen. Wenn ihr diesen Button drückt, soll sich das jeweilige Bild ändern:



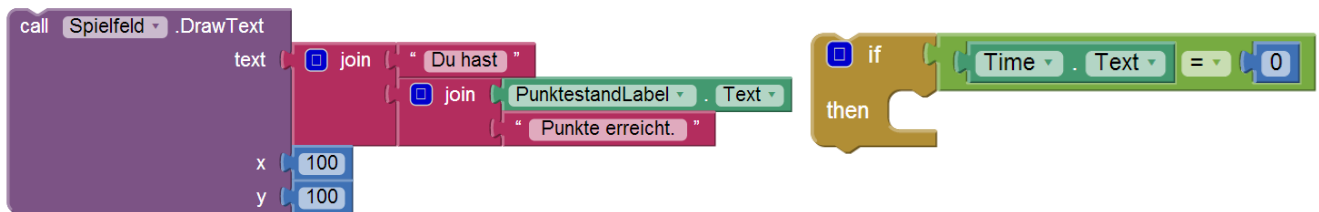
### **Ein zweiter (böser) Maulwurf auf dem Spielfeld**

Wie wäre es mit einem zweiten Maulwurf, der negative Punkte gibt, wenn man ihn anklickt?

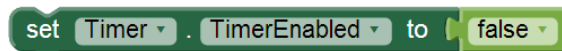
- Dazu braucht ihr ein neues ImageSprite.
- Außerdem müsst ihr anschließend Maulwurf2 genau wie den ersten Maulwurf zufällig bewegen und bei einem Treffer 10 Punkte abziehen.

### **Ein Spielende...**

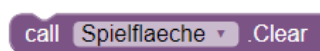
wäre noch sehr schön. Dazu könnt ihr ein neues Label hinzufügen, das bei 100 anfängt. Jedes Mal, wenn der Timer auslöst, reduziert ihr den Wert um 1. Wenn der Wert 0 erreicht wird, könnt ihr bspw. das Spiel resetten oder einen Text ausgeben, der sagt, wie viele Punkte der Spieler bekommen hat.



Außerdem wäre es wichtig, den Timer auszuschalten, damit der Maulwurf stehenbleibt.



Tipp: Wenn ihr den Reset-Knopf drückt, muss der Timer wieder eingeschaltet und die Schrift vom Bildschirm entfernt werden. Schaut euch dazu einmal die Funktionen vom Spielfeld an, denn ihr wollt es **clearen**.



### Quellenverzeichnis:



– Quelle: pixabay.com, Autor: OpenClipartVectors (CC0)



– Quelle: pixabay.com, Autor: geralt (CC0)



– Quelle: InfoSphere

**Alle weiteren Grafiken** – Quelle: Screenshots des MIT App Inventors: (<http://appinventor.mit.edu/explore/>)