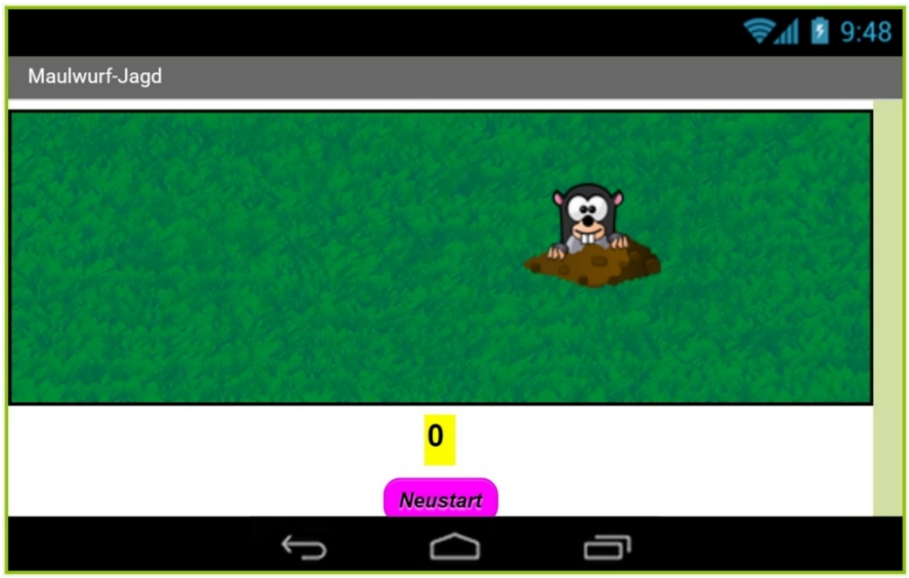
MoleMash

Fang den Maulwurf

In diesem Spiel geht es darum, Maulwürfe zu fangen, die fröhlich auf dem Bildschirm herumspringen. Wenn du Maulwürfe nicht so magst, kannst du dir natürlich ein anderes Tier oder einen Gegenstand aussuchen. Das Design der App ist ganz dir überlassen.

Das Ziel des Spiels ist, in einer bestimmten Zeit (z.B. eine Minute) so oft wie möglich auf den Maulwurf zu tippen. Der Punktestand erhöht sich jedes Mal und wird auch im Spiel angezeigt. Die verbleibende Zeit sollte auch angezeigt werden. Ist die Zeit abgelaufen, so friert das Spiel ein, und der Punktestand kann nicht weiter erhöht werden. Bei einem Klick auf die Neustart-Taste beginnt das Spiel von vorne.



1. Design: Wie soll die App aussehen?

a) Lege ein neues Projekt an und gib ihm einen Namen (z.B. *MoleMash*). Die App wird nachher unter diesem Namen auf dem Smartphone/Tablet erscheinen.

b) Entscheide dich, ob das Spiel im Quer- oder Hochformat gespielt werden soll. Das Format kann angepasst werden, indem man unter *Komponenten* auf *Screen1* klickt und unter *Bildschirmausrichtung* die Bildschirmausrichtung *Hochformat* oder *Querformat* auswählt.

c) Lade ein Bild und einen Ton hoch: Klicke dazu unten bei *Medien* auf *Datei Hochladen*. Einige Bilder und Töne findest du auf dem Desktop in den Ordnern *Bilder* bzw. *Sounds*.

d) Ziehe folgende Bildschirmelemente aus der Palette auf die Bildschirmfläche:

- 1 *Taste* (für Neustart)
- 2 *Bezeichnungen* (für die Anzeige der Punkte und der Zeit)
- 1 *Zeichenfläche* (zu finden unter *Zeichnen und Animation*)

Die Zeichenfläche ist die Hauptkomponente für das Spiel und sollte **so viel Platz wie möglich** einnehmen.
Um die Elemente schöner zu anzuordnen, kannst du *Ausrichtungen* verwenden (in der Palette zu finden unter *Anordnung*). Außerdem

kann für jedes Element unter *Eigenschaften* eine Höhe und eine Breite bestimmt werden. Benenne die Elemente so wie im Bild.

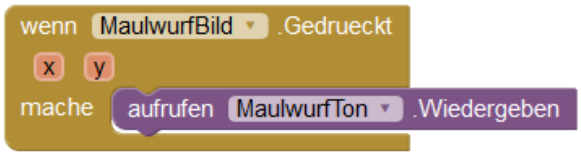
e) Verbinde dein Smartphone über die MIT-AI2-Companion-App und teste deine Oberfläche.
Wenn dein Spiel im Querformat gespielt wird, kannst du **Strg + F11** drücken, um den Emulator zu drehen.

2. Der Maulwurf

a) Ziehe eine *ZeichenAnimation* aus der Palette (unter *Zeichnen und Animation*) auf die *Zeichenfläche*. Benenne sie in *MaulwurfBild* um. Wähle unter *Bild* das in 2.1 c) hochgeladene Bild aus. Passe die Größe über die Felder *Höhe* und *Breite* an.

b) Ziehe ein *Tonwiedergabe*-Element aus der Palette (unter *Medien*) auf die Benutzer-oberfläche. Benenne es in *MaulwurfTon* um. Weise ihm dann unter *Quelle* die in 2.1 c) hochgeladene Ton-Datei zu.

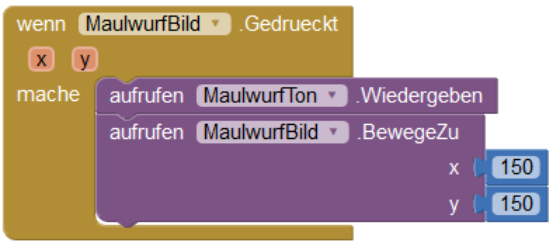
c) Wechsle oben rechts zur *Blöcke*-Ansicht. Klicke (unter *Screen1*) auf *MaulwurfBild* und ziehe die Funktion wenn **MaulwurfBild.Gedrueckt** auf die Arbeitsfläche.
Dann klicke auf das Tonwiedergabe-Element *MaulwurfTon* und ziehe die Funktion aufrufen **MaulwurfTon.Wiedergeben** auf die Arbeitsfläche. Hänge sie in den *Gedrueckt*-Block ein.



3. Bewegung

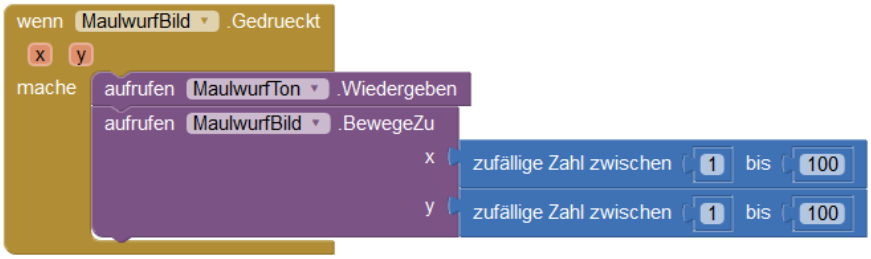
Das Bild soll bei Berührung an eine zufällige Position auf der *Zeichenfläche* springen.

a) Klicke unter *Screen1* → *Zeichenfläche1* → *MaulwurfBild* auf die Funktion aufrufen **MaulwurfBild.BewegeZu** und hänge sie in den *Gedrueckt*-Block ein.
Dieser Baustein lässt das MaulwurfBild an eine andere Position springen, die du über die x- und y-Koordinate festlegen kannst.



b) Teste die App für verschiedene x- und y-Koordinaten.

c) Um das Bild an eine **zufällige** Position springen zu lassen, kannst du statt der festen Zahlenwerte die Funktion **zufällige Zahl** verwenden. Diese findest du unter *Eingebaut* → *Mathematik*. Die Funktion generiert eine Zufallszahl zwischen zwei Werten (anfänglich zwischen 1 und 100).
Ersetze die festen Werte für die x- und y-Koordinaten mit dem Aufruf dieser Funktion und teste die App.



Das ist die Basisversion von MoleMash. Nun bist du gefragt! Überlege dir, wie du das Spiel spannender machen kannst. Im Folgenden findest du eine Reihe von Ideen inklusive Beschreibung, wie du diese umsetzen kannst.

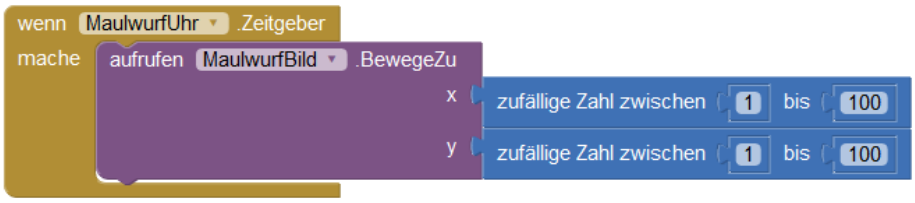
4. Erweiterungsideen

4.1 Der Maulwurf springt von alleine

a) Ziehe eine *Uhr* aus der Palette (*Sensoren*) auf die Benutzeroberfläche und nenne sie *MaulwurfUhr*.

b) Ziehe im *Blöcke*-Editor unter *MaulwurfUhr* die Funktion wenn **MaulwurfUhr.Zeitgeber** auf die Arbeitsfläche und lass den Maulwurf so springen, wie du es in 2.3 c) bereits getan hast.

Die *Uhr* führt in regelmäßigen Zeitabständen den Inhalt des **Zeitgeber**-Blocks aus. Diesen Zeitabstand kannst du im *Designer*-Editor bei den Eigenschaften der *MaulwurfUhr* unter *Zeitgeberintervall* einstellen.
Willst du den Maulwurf zum Beispiel alle 100 ms springen lassen, so musst du für das Intervall den Wert 100 wählen.



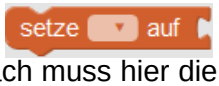
4.2 Punktezähler

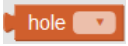
Benutze die in 1 d) erstellte *ScoreBezeichnung*, um die Anzahl der getroffenen Maulwürfe anzuzeigen. Für die Anzahl brauchst du eine Variable, die jedes Mal, wenn der Maulwurf getroffen wird, um 1 erhöht wird.

a) Klicke unter *Eingebaut* → *Variablen* auf

Benenne die Variable von „Name“ in „Score“ um und gib ihr den Startwert 0.

b) Jedes Mal, wenn der Maulwurf getroffen wird, sollte die Variable um 1 erhöht werden. Dies funktioniert nach dem Schema $Score = Score + 1$

Klicke auf *Eingebaut* → *Variablen* und ziehe 

in den Arbeitsbereich. Wähle die eben erstellte Variable *Score* aus. Danach muss hier die obige Gleichung eingehängt werden: Den **+**-Baustein findest du unter *Eingebaut* → *Mathematik*. Den Wert der Variablen kannst du über *Eingebaut* → *Variablen* → 

auslesen:

c) Die Bezeichnung soll nun die in global **Score** gespeicherte Punktezahl anzeigen. Klicke dazu auf *Screen1* → *ScoreBezeichnung* und ziehe setze **ScoreBezeichnung.Text** auf in die Arbeitsfläche. Hänge dann den Wert der Variablen über hole **global Score** ein.

d) Füge beide Blöcke in den Block **MaulwurfBild.Gedrueckt** ein.

4.3 Zeitmessung

Eine Zeitanzeige soll zeigen, wie viele Sekunden seit Spielstart schon verstrichen sind.

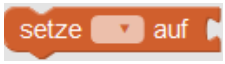
a) Ziehe eine *Uhr* aus der Palette (unter *Sensoren*) auf die Benutzeroberfläche und nenne sie *SpielZähler*. Wechsle zum *Blöcke*-Editor.

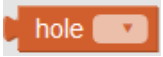
b) Erstelle eine Variable (*Eingebaut* → *Variablen*) und nenne sie *Zeit*. Diese soll die verstreichenden Sekunden speichern.
Gib ihr den Startwert 0.

c) Ziehe den Block wenn **SpielZähler.Zeitgeber** auf die Arbeitsfläche.
Der Inhalt in diesem Block wird standardmäßig alle 1000 ms (also ein Mal pro Sekunde) ausgeführt – perfekt für einen Sekundenzähler.

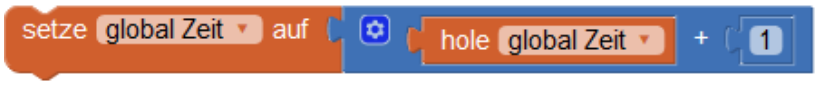
d) Bei jedem Aufruf des Blocks wenn **SpielZähler.Zeitgeber** soll nun die Variable *Zeit* um 1 erhöht werden. Das funktioniert wieder nach dem Schema:

$$Zeit = Zeit + 1$$

Klicke dazu auf *Eingebaut* → *Variablen* und ziehe 

auf den Arbeitsbereich. Wähle *global Zeit* aus. Dann sollte hier die obige Gleichung eingehängt werden.
Den **+**-Baustein findest du unter *Eingebaut* → *Mathematik*. Den aktuellen Wert der Variablen kannst du über *Eingebaut* → *Variablen* → 

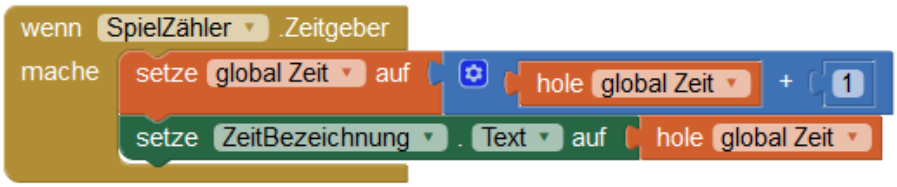
holen.



Falls du anstatt eines Sekundenzählers lieber einen Countdown hättest, musst du nur den Anfangswert der Variable *Zeit* auf den Startwert des Countdowns setzen und dann den Wert herunterzählen, also einmal pro Sekunde die Zahl 1 vom aktuellen *Zeit*-Wert abziehen:

$$Zeit = Zeit - 1$$

e) Um die Zeit auch anzuzeigen, wähle eine *Bezeichnung* von deiner Benutzeroberfläche (hier *ZeitBezeichnung*) und setze den Text auf den Wert der Variablen *Zeit*.



4.4 Neustartknopf

Damit man die App nicht jedes Mal neu starten muss, wenn man ein neues Spiel anfangen will, benötigst du einen Neustartknopf. Wird der gedrückt, so wird der Zustand des Spiels auf die Anfangswerte zurückgesetzt. Dazu gehört der Punktestand (*Score*) und die Uhr (*Zeit*).

a) Klicke im *Blöcke*-Editor unter *Screen1* → *NeustartTaste* auf wenn **NeustartTaste.Klick**.

b) Setze in diesem Block nun alle Werte auf ihren Anfangswert, wie zum Beispiel die Punkte, die verstrichene Zeit oder ggf. auch die Position des Maulwurfs.

4.5 Spielende

Ist das Spielende erreicht, soll das Spiel angehalten werden.

a) Um den Spielfluss zu stoppen, brauchst du als erstes eine Variable:
Klicke im *Blocks*-Editor unter *Eingebaut* → *Variablen* auf

Nenne die Variable *Spielen*.
Gib ihr den Wert **wahr** (*Eingebaut* → *Logik,* **wahr** & **falsch** sind **keine** Text-Bausteine).

Diese Variable gibt an, ob das Spiel gerade läuft (*Spielen* = **wahr**) oder schon beendet ist (*Spielen* = **falsch**).

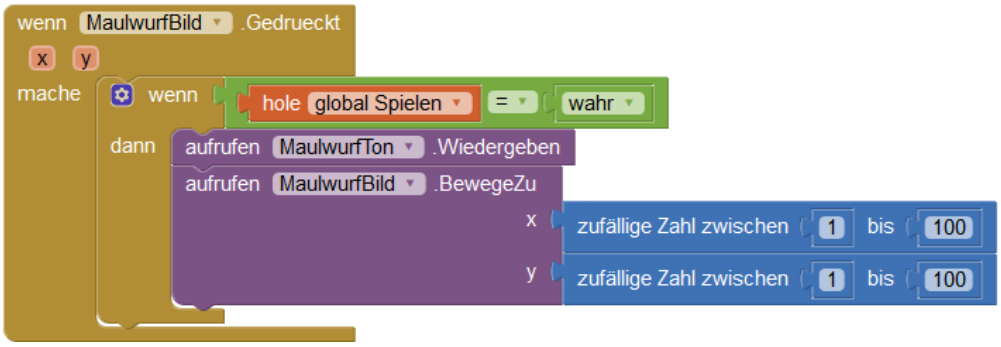
b) Alle Aktionen, die während des Spiels möglich sind, sollen nun von dieser Variable abhängig gemacht werden und nur ausgeführt werden, wenn *Spielen* = **wahr** ist, also das Spiel noch läuft.
Dies lässt sich mit **wenn**-Abfragen realisieren. Diese überprüfen eine Bedingung und führen Anweisungen nur dann aus, wenn die Bedingung **wahr** ist.

c) Klicke auf *Eingebaut* → *Logik* → =, um die Bedingung zu bauen. Fülle den Vergleich mit der Variablen *Spielen* und dem *Logik*-Wert **wahr**.

Die Bedingung ist also erfüllt, wenn *Spielen* = **wahr** ist.

d) Ziehe eine **wenn**-Abfrage (*Eingebaut* → *Steuerung*) auf die Arbeitsfläche und stecke oben die eben erstellte Bedingung ein. Alles in dem **dann**-Teil unterhalb der Bedingung wird nur dann ausgeführt, wenn *Spielen* = **wahr** ist, also das Spiel gerade läuft.

e) Um nun beispielsweise den Maulwurf nur noch dann hüpfen zu lassen, wenn das Spiel auch gerade läuft, musst du den **MaulwurfBild.BewegeZu**- und sinnvollerweise auch den **MaulwurfTon.Wiedergeben**-Aufruf im **dann**-Teil vom **wenn**-Block unterbringen:

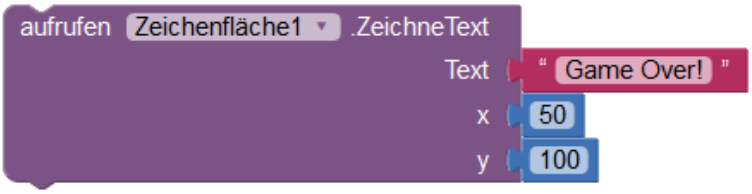


f) Jetzt musst du dir nur noch überlegen, wann das Spiel eigentlich enden soll.
Erstelle dir dafür dann analog eine **wenn**-Abfrage mit einer Bedingung, die im **dann**-Block die Variable *Spielen* = **falsch** setzt. Baue die neue Abfrage an passender Stelle in dein Programm ein.

4.6 Text auf dem Bildschirm anzeigen

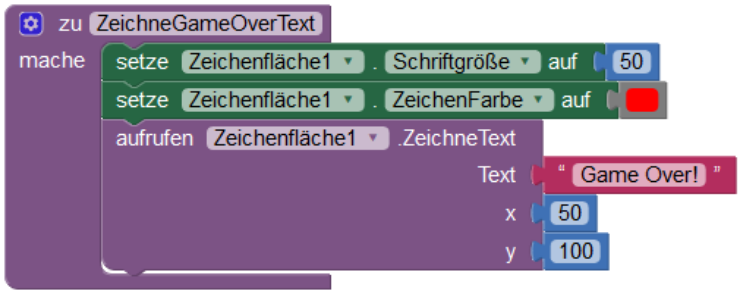
Ein Text, wie z.B. „*Game Over*“ soll bei Spielende auf dem Bildschirm angezeigt werden.

a) Klicke auf *Screen1* → *Zeichenfläche1* → **Zeichenfläche1.ZeichneText**. Fülle den **Text**-Steckplatz mit einem Text-Block (*Eingebaut*→ *Text*).
Die Koordinaten **x** und **y** geben die Position an, an der der Text erscheinen soll.



b) Um Größe und Schriftfarbe zu verändern, musst du über den **ZeichneText**-Aufruf die Anweisung setze **Zeichenfläche1.Schriftgröße** auf und setze **Zeichenfläche1.Zeichenfarbe** auf einfügen. Farben findest du unter *Eingebaut* → *Farben*.

c) Erstelle nun eigenen Block, genannt *Prozedur*, indem du den Block zu **Prozedur** mache (*Eingebaut* → *Proze-dur*) auf die Arbeitsfläche ziehst.
Benenne sie von „Prozedur“ um in beispielsweise „ZeichneGameOver-Text“ und hänge dort alle eben erstellten Blöcke ein.



Verwende den Block aufrufen ZeichneGameOverText (*Eingebaut* → *Prozeduren*), um die *Prozedur* aufzurufen, damit der Text angezeigt wird..
Hinweis: Eine *Prozedur* fasst mehrere Operationen zusammen, die mit einem einzelnen Aufruf von überall ausgeführt werden können. Auf diese Weise muss nicht jede Operation einzeln eingefügt werden, was das Programm übersichtlicher macht.

d) Der erstellte Text bleibt solange auf dem Bildschirm stehen, bis er mit dem Aufruf

aufrufen Zeichenfläche1 .Lösche (*Screen1* → *Zeichenfläche1*) gelöscht wird.

Setze diesen Block an eine geeignete Stelle in deinem Programm (falls du den Neustartknopf implementiert hast, etwa in den wenn **NeustartTaste.Klick**-Block).

5. Freie Arbeit

Erweitere dein Spiel nach deiner Vorstellung und deinen Wünschen. Was fehlt, um es wirklich optisch ansprechend, interessant, spaßig oder anspruchsvoll zu machen?
Für den Fall, dass dir auf Anhieb nichts einfällt, gibt es im Folgenden eine kleine Liste mit Anregungen und Hinweisen.

Anregungen:

- **Schönere Zeitanzeige:**
  Die Funktionen **Modulo**- und **Quotient** unter *Eingebaut* → *Mathematik* ermöglichen dir eine Zeitanzeige im *„Minuten : Sekunden“*-Format.

- **Mehr Maulwürfe:**
  Du kannst mehr Maulwürfe einfügen, indem du mehr *ZeichenAnimationen* auf die *Zeichenfläche* ziehst. Im *Blöcke*-Editor unter *Irgendeine Komponente* findest du Funktionen, die die Arbeit mit mehreren *ZeichenAnimationen* einfacher machen.
  Du kannst die Anzahl der Maulwürfe auch während des Spiels ändern, indem du das Attribut *Sichtbar* der *ZeichenAnimationen* veränderst.

- **Unberechenbares Verhalten der Maulwürfe:**
  Du könntest nach jedem Sprung das Intervall der *MaulwurfUhr* zufällig anpassen.

- **Minuspunkte durch Igel**
  Du kannst Igel einfügen, die sich ähnlich verhalten wie die Maulwürfe, aber Minuspunkte geben, wenn man sie berührt (Aua).

-
- **Verschiedene Schwierigkeitsgrade oder Spielmodi:**
  Ihr könnt einige Tasten und globale Variablen einfügen, über die sich das Verhalten des Maulwurfs oder auch die Punktevergabe ändern lässt.

- **Automatische Geschwindigkeitsanpassung:**
  Du kannst die Geschwindigkeit automatisch anpassen lassen, wenn der Spieler einen Maulwurf trifft oder verfehlt. Je nachdem wieviel schneller bzw. langsamer das Spiel pro Treffer bzw. Fehlschlag wird, sind besonders einfache oder besonders anspruchsvoll Schwierigkeitsgrade möglich.

-
- **Punktespeicherung:**
  Du kannst mit einer *TinyDatenbank* eine Highscoreliste speichern und wieder auslesen. Den Inhalt der Highscoreliste kannst du über eine zusätzliche Bezeichnung anzeigen, wenn du ihn mit den Bausteinen unter *Eingebaut* → *Text* formatierst.
  Um alles übersichtlich zu halten, kannst du auch Bildschirmelemente (auch *Ausrichtungen*) über das Attribut *Sichtbar* ein- und ausblenden.

- **Leben / alternatives Spielende:**
  Du kannst mit einer weiteren globalen Variable und **wenn-dann**-Abfragen dem Maulwurf eine bestimme Anzahl an Leben geben und das Spiel enden lassen, sobald alle Leben weg sind, anstatt nach einer festen Zeit.
  Oder du gibst dem Spieler nur eine begrenzte Anzahl von Versuchen mit denen er möglichst viele Maulwürfe treffen soll.

- **„Laufende“ Maulwürfe:**
  Du kannst die Maulwürfe auch laufen anstatt springen lassen, indem du die Werte *Richtung* und *Geschwindigkeit* der *ZeichenAnimationen* anpasst. Die Methode aufrufen **ZeichenAnimation.Abprall** lässt in Kombination mit wenn **ZeichenAnimation.RandErreicht** eine *ZeichenAnimation* von einer Kante der Zeichenfläche abprallen.

Viel Spaß bei der Maulwurf-Jagd!!