


### (3) Maulwurf-Spiel (Mole Mash)

Jetzt programmieren wir das Maulwurf-Spiel. Das Ziel ist den Maulwurf zu erwischen, während dieser hin und her hüpft.

Wir zählen, wie oft das klappt (Trefferpunkte) innerhalb von 10 Sekunden Spielzeit.



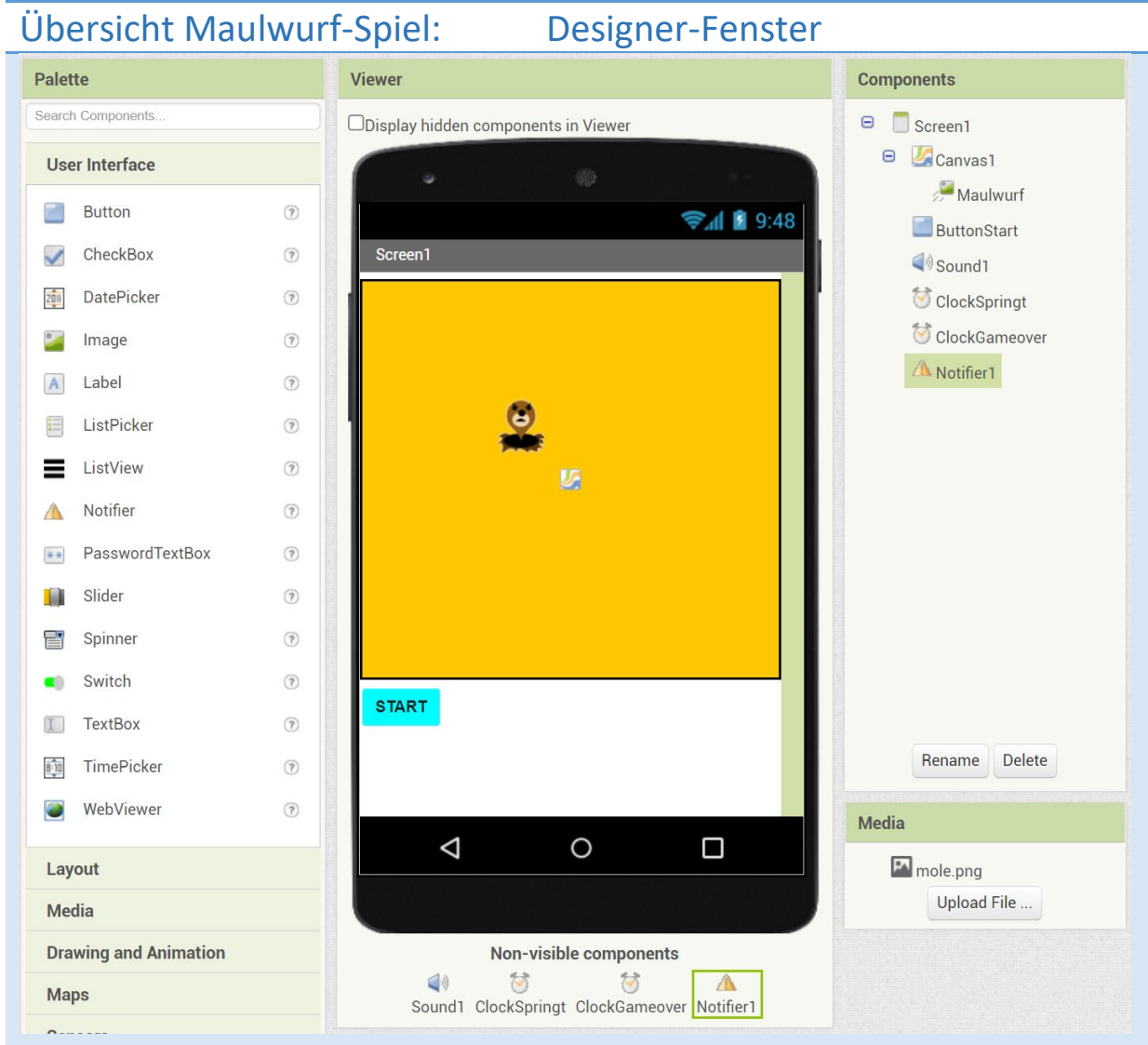
Wir werden einige neue Komponenten und Blöcke brauchen:

<b>TIPPS:</b> 	<b>COMPONENTS</b>	
	<b>Canvas</b>	Leinwand. Hier benutzt als Spielfläche.
	<b>Clock:</b>	Zeitschaltuhr oder Timer. Zählt ein Intervall und gibt am Ende ein Tickerzeichen aus. Kann immer wieder neu mit dem Zählen starten.
	<b>Sprite:</b>	Eine Spielfigur, welche in deinem Spiel ständig bewegt.
	<b>Notifier:</b>	Zeigt die BenutzerIn eine Meldung am Ende des Spiels an.
	<b>BLOCKS</b>	
	<b>Text join:</b>	Hängt alle Eingaben zusammen, um eine einzige Textzeile zu bilden.

Starte ein neues Projekt und nenne dieses MaulwurfSpiel\_einfach. Füge die folgenden Komponenten im Viewer (Designer-Fenster) ein. Du musst jede Komponente umbenennen und gewisse Eigenschaften anpassen.

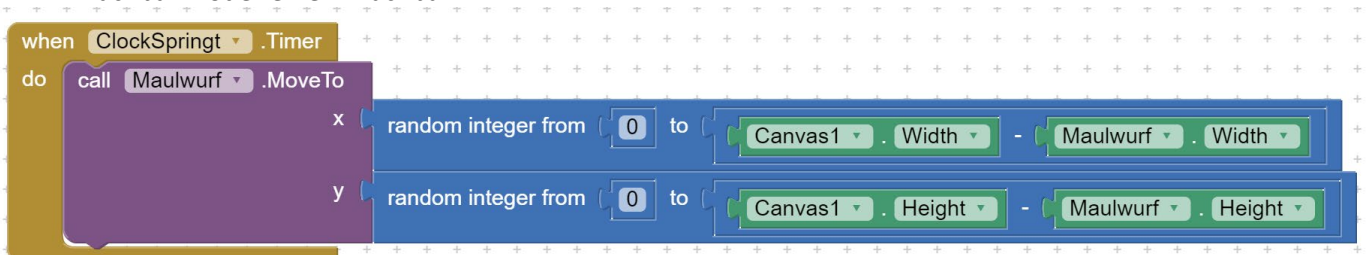
Komponenten	Palette (Kategorie)	Umbenennen in	Wofür plus Eigenschaften
<b>Canvas</b>	Drawing and Animation	Canvas1	Spielfläche <u>Eigenschaften:</u> -BackgroundColor: <Nach Geschmack> -Height: 300 Pixels -Width: fill parent
<b>ImageSprite</b>	Drawing and Animation	Maulwurf Bmk: platzieren innerhalb von Canvas	Spielfigur <u>Eigenschaften:</u> Picture: mole.png
<b>Button</b>	User Interface	ButtonStart	Anzeige «START» Man drückt diesen Knopf, um das Spiel zu starten
<b>Sound</b>	Media	Sound1	Vibriert, wenn der Maulwurf berührt wird
<b>Clock</b>	Sensors	ClockSpringt	Zählt ein Zeitintervall kontinuierlich <u>Eigenschaften:</u> -TimerAlwaysFires: ja -TimerEnabled: nein -TimerInterval: 500
<b>Clock</b>	Sensors	ClockGameOver	Zählt ein Zeitintervall kontinuierlich <u>Eigenschaften:</u> -TimerAlwaysFires: ja -TimerEnabled: nein -TimerInterval: 10000
<b>Notifier</b>	User Interface	Notifier1	Zeigt die Trefferpunkte am Ende des Spiels.

Überprüfe die Zusammenfassung des Designer-Teils unten. Hast du alles programmiert?



Jetzt müssen wir die entsprechenden Aktionen im Blocks-Fenster programmieren. Denk daran, deine App immer wieder zu testen.

- Wir beginnen damit, das Hüpfen des Maulwurfs zu programmieren. Er soll nach jedem Intervall von ClockSpringt zufällig innerhalb des Canvas springen. Schau auf der linken Seite unter ClockSpringt, Maulwurf und Math nach den benötigten Aktionen und baue sie wie unten gezeigt. Wie soll die Programmierung sicherstellen, dass der Maulwurf innerhalb des Canvas bleibt? Diskutiere mit einer Nachbarin oder einem Nachbar.



- Wir brauchen auch einen Variablen, um die "Trefferpunkte" (Getroffen) zu zählen. Am Anfang hat man null Punkte.



3. In den Funktionen von Canvas (Spielfläche) findest du einen Kontrollblock, der aktiviert wird, wenn der Spieler den Bildschirm berührt. Da kann man überprüfen, ob der Spieler den Maulwurf erwischt hat, und dann die Trefferpunkte erhöhen.

```

when Canvas1 .Touched
do
  if get touchedAnySprite
  then
    set global Getroffen to get global Getroffen + 1
  
```

4. Wenn die SpielerIn den Maulwurf fängt, soll das Telefon vibrieren. Füge die passende Aktion hinzu.

```

when Maulwurf .Touched
do
  call Sound1 .Vibrate
  100
  
```

5. Am Ende der Spielzeit (ClockGameOver) wollen wir alle Clocks anhalten und die Gesamtpunktzahl mit einem Notifier anzeigen. Finde die Aktionen unter den entsprechenden Blöcken, und baue sie wie unten gezeigt. Finde heraus, was der String <br> im Text Join Block ändert.

```

when ClockGameOver .Timer
do
  set ClockSpringt .TimerEnabled to false
  set ClockGameOver .TimerEnabled to false
  call Notifier1 .ShowDialog
  message join "Getroffen: "
  get global Getroffen
  "<br>"
  title "Game over"
  buttonText "OK"
  
```

6. Denk jetzt darüber nach, wie man das Spiel starten und immer wieder neu starten kann. Sicher muss man die Trefferpunkte zurücksetzen, und die Clocks einschalten (TimerEnabled auf True setzen, siehe Snapshot unten).

```

set ClockSpringt .TimerEnabled to true
  
```

Den Startbutton hast du bereits in dem Designer eingebaut. Programmiere jetzt alle Aktionen, die du brauchst!

Alles parat? Zeit zum Testen und Korrigieren. Überprüfe deinen Code mit der Übersicht auf der nächsten Seite.

## Übersicht Maulwurf-Spiel: Blocks-Fenster

```

when ClockSpringt.Timer
do
  call Maulwurf.MoveTo
  x random integer from 0 to Canvas1.Width - Maulwurf.Width
  y random integer from 0 to Canvas1.Height - Maulwurf.Height

initialize global Getroffen to 0

when ButtonStart.Click
do
  set global Getroffen to 0
  set ClockSpringt.TimerEnabled to true
  set ClockGameOver.TimerEnabled to true

when Canvas1.Touched
x y touchedAnySprite
do
  if get touchedAnySprite
  then
    set global Getroffen to get global Getroffen + 1

when ClockGameOver.Timer
do
  set ClockSpringt.TimerEnabled to false
  set ClockGameOver.TimerEnabled to false
  call Notifier1.ShowDialog
  message join " Getroffen: " get global Getroffen " <br> "
  title " Game over "
  buttonText " OK "

when Maulwurf.Touched
x y
do
  call Sound1.Vibrate
  milliseconds 100
  
```

### HERAUSFORDERUNGEN :

- (1) Baue eine zweite Variable, Verpasst, und zähle auch die verpassten Punkte (wenn man den Maulwurf beim Drücken des Bildschirms nicht erwischt). Zeige am Ende des Spiels sowohl "Getroffen" als auch "Verpasst" Punkte mit dem Notifier an.
- (2) Erweitern Sie das Spiel, sodass das Intervall zwischen zwei Sprüngen je nach Leistung des Spielers variiert. Zum Beispiel: kürzer, wenn es mehr Treffer als Fehler gibt, und umgekehrt.
- (3) Und noch weitere Ideen von dir....

Die originelle detaillierte Anweisung für das «Maulwurf-Spiel» Tutorial findet man unten:

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

Oder runterladen das Kapitel:

<http://www.appinventor.org/apps/molemash/molemash.pdf>

### BEMERKUNG

Das Original-Maulwurf-Tutorial verwendet Beschriftungen (Labels), Tabellenanordnungen (Table Arrangements) und Prozeduren (Procedures), die sicherlich interessante Ressourcen sind, aber insgesamt länger zum Programmieren brauchen. Diese vereinfachte Version ist gedacht für kurze Workshops.

## ANHANG

## Lösung zur Herausforderung (1)

The image displays a collection of Scratch code blocks for a game challenge solution, organized into several functional groups:

- Initial Positioning:** A `when ClockSpringt .Timer` block triggers a `do` loop that calls `Maulwurf .MoveTo`. The x and y coordinates are determined by `random integer from 0 to Canvas1 .Width - Maulwurf .Width` and `random integer from 0 to Canvas1 .Height - Maulwurf .Height`, respectively.
- Global Variables:** Two `initialize global` blocks set `Getroffen` and `Verpasst` to `0`.
- Game Start:** A `when ButtonStart .Click` block sets `global Getroffen` and `global Verpasst` to `0`, and enables the `ClockSpringt .TimerEnabled` and `ClockGameOver .TimerEnabled` to `true`.
- Collision Detection:** A `when Canvas1 .Touched` block with `x` and `y` coordinates uses `touchedAnySprite`. An `if` block checks `get touchedAnySprite`. If true, it sets `global Getroffen` to `get global Getroffen + 1`. If false, it sets `global Verpasst` to `get global Verpasst + 1`.
- Game Over:** A `when ClockGameOver .Timer` block sets `ClockSpringt .TimerEnabled` and `ClockGameOver .TimerEnabled` to `false`. It then calls `Notifier1 .ShowMessageDialog` with a `message` block containing a `join` of: `" Getroffen: "`, `get global Getroffen`, `" <br> "`, `" Verpasst: "`, `get global Verpasst`, `" <br> <br> "`, and `" Nochmals starten? "`. The `title` is `" Game over "` and the `buttonText` is `" OK "`.
- Audio Feedback:** A `when Maulwurf .Touched` block with `x` and `y` coordinates calls `Sound1 .Vibrate` for `100` milliseconds.