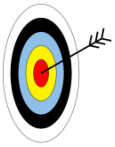
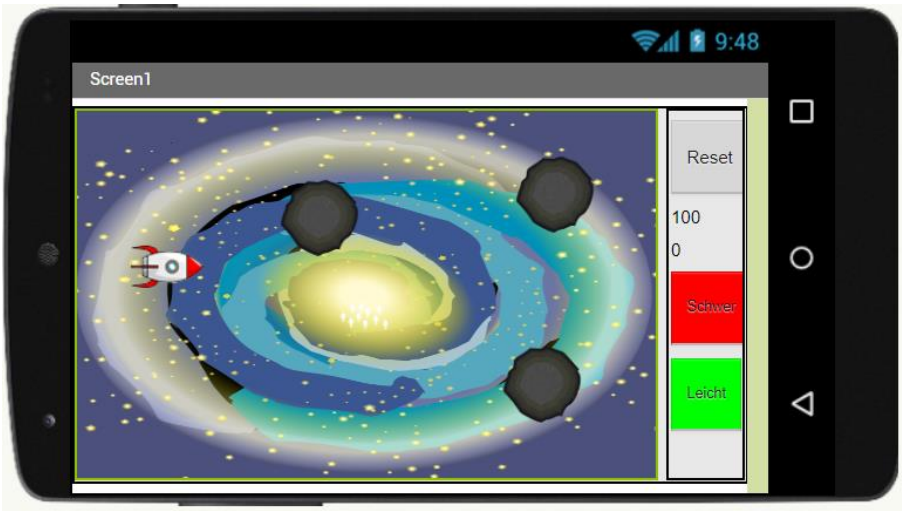
2b Asteroids



Ihr habt euch für **Asteroids** entschieden. Ziel des Spiels ist, ein **Raumschiff**  [1] durch den **Weltraum**  [2] zu bewegen und dabei kleinen **Asteroiden**  [3] auszuweichen. Dieses Arbeitsblatt wird euch helfen,...

- einen **Weltraum** mit **Rakete** und **Asteroiden** darzustellen,
- die **Steuerung** der **Rakete** zu entwerfen,
- die **Bewegung** der **Asteroiden** umzusetzen und
- **Punkte hochzuzählen** für die Zeit, die eure Rakete überlebt.

Schwierigkeitsgrad: 



[4]

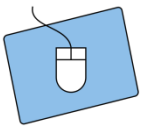
Während des gesamten Moduls geben euch die Arbeitsblätter Hinweise zur Umsetzung. Achtet dabei einfach auf die folgenden Symbole, die...

- euer Arbeiten strukturieren und Teilziele aufzeigen, 
- euch Hilfen geben, Wichtiges, Schwieriges, etc. kennzeichnen und 
- die Arbeitsaufträge und Aktionen beinhalten. 

2b Asteroids

Anlegen des Projekts

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



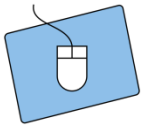
1. Erstellt euer Projekt, und gebt ihm einen passenden Namen.
2. Verbindet den App Inventor wie auf dem ersten Arbeitsblatt beschrieben mit dem Tablet/Smartphone.

Der Aufbau eurer App

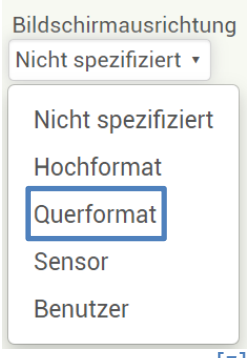
Als Nächstes müsst ihr euch Gedanken darüber machen, wie eure App aussehen soll. Eure App braucht Platz für folgende Dinge (die ihr der Reihe nach einbauen werdet):

- eine **Spielfläche**, auf der die Rakete sich bewegen kann,
- eine **Reset-Taste**, mit der ihr das Spiel neustartet,
- eine **Punkteanzeige**, die eure aktuellen Punkte hochzählt.

Das sind schon recht viele Sachen. Damit diese alle Platz haben, solltet ihr zunächst die **Bildschirmausrichtung** (Screen1) von **Nicht spezifiziert** in **Querformat** ändern.



1. Wählt unter den **Komponenten Screen1** aus.
2. Sucht unter den **Eigenschaften** den Eintrag **Bildschirmausrichtung**.
3. Ändert den Wert von **Nicht spezifiziert** in **Querformat**.



[5]

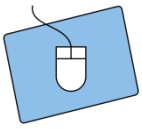
Die Bestandteile eurer App

Nun könnt ihr anfangen, die einzelnen Elemente eurer App hinzuzufügen und diese mit Hilfe der vorgestellten **Anordnungen** sinnvoll zu platzieren.

- Für die **Spielfläche** braucht ihr eine **Zeichenfläche**; diese findet ihr unter **Zeichnen und Animation**.

Eine Zeichenfläche könnt ihr euch wie eine Leinwand vorstellen, auf der sich Objekte wie z. B. die Rakete bewegen können.

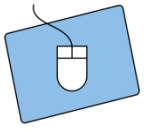
- Für die **Reset-Taste** braucht ihr eine **Taste** mit dem Text „Reset“.
- Für die **Punkteanzeige** benötigt ihr eine **Bezeichnung** mit dem Text „0“.



1. Macht euch Gedanken darüber, wie ihr die Elemente positionieren wollt.
2. Zieht die **Anordnungen**, die ihr für euer Layout braucht, in den Betrachter hinein.
3. Fügt eurer App die drei Bestandteile **Zeichenfläche, Taste, Bezeichnung** hinzu.

Der erste Test

Nachdem ihr die Bestandteile sinnvoll angeordnet habt, könnt ihr direkt testen, wie das Ganze auf dem Tablet/Smartphone aussieht.

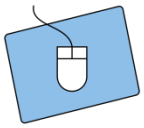


1. Startet dazu die App auf dem Tablet/Smartphone, wenn ihr dies noch nicht gemacht habt.
2. Gefällt euch die Anordnung? Wenn ja, dann könnt ihr weitermachen. Ansonsten überarbeitet sie einfach nochmal.

Lass eure App auf dem Tablet/Smartphone laufen. Der App Inventor aktualisiert alle Änderungen, die ihr vornehmt, und zeigt euch diese direkt an, ohne dass ihr die App neustarten müsst. Sollte dies einmal nicht der Fall sein, dann setzt die Verbindung zurück.

Umbenennen nicht vergessen 😉

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



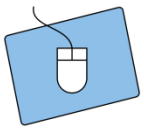
Gebt den Bestandteilen eurer App sinnvolle Namen.

Ein Hintergrund für eure Spielfläche

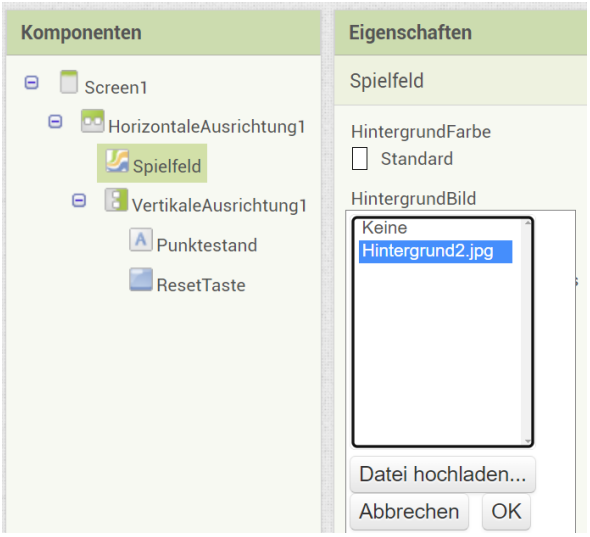
Als Nächstes sollt ihr den Weltraum darstellen, in dem sich die Rakete bewegen kann. Ladet dazu einen der Hintergründe aus dem folgenden Ordner hoch:

Desktop / InfoSphere goes Android / Asteroids

Im Anschluss müsst ihr eurer Zeichenfläche den Hintergrund noch zuweisen:

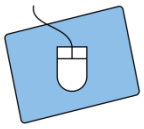


1. Wählt unter den **Komponenten** die **Zeichenfläche** aus.
2. Klickt unter den **Eigenschaften** auf **Hintergrundbild**, und weist der Zeichenfläche einen der Hintergründe zu.

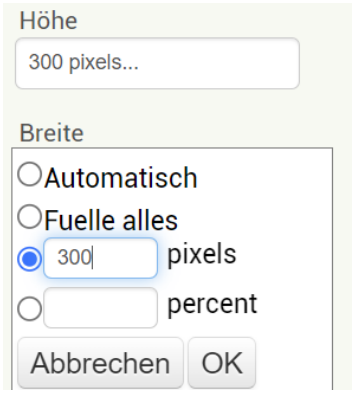


[6]

Wie ihr schon von Blatt 1 wisst, muss man die Größe beim App Inventor manchmal per Hand einstellen.



3. Ihr habt immer noch eure **Zeichenfläche** (die in der Abbildung schon in Spielfeld umbenannt ist) ausgewählt. Sucht in den **Eigenschaften** nach **Breite** und **Höhe**, und setzt beide auf 300 Pixel.
4. Testet jetzt eure App, und ändert gegebenenfalls die Anzahl der Pixel, damit es auf dem Tablet/Smartphone gut aussieht.

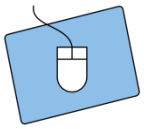


[7]

Auf der nächsten Seite geht es weiter mit der Rakete.

Die Rakete

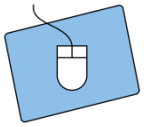
Als Nächstes soll die Rakete auf das Spielfeld.



1. Wechselt in der Palette zu **Zeichnen und Animation**.
2. Zieht eine **ZeichenAnimation** direkt in den Weltraum-Hintergrund.
3. Ändert den Namen der ZeichenAnimation in **Rakete**.
4. Ladet nun aus dem Ordner, aus dem ihr auch den Hintergrund habt (S. 4), die Rakete hoch, und weist sie der ZeichenAnimation zu.

Eine **ZeichenAnimation** ist ein besonderes Bild. Im Gegensatz zu einem normalen Bild kann die ZeichenAnimation sich auf dem Spielfeld bewegen.

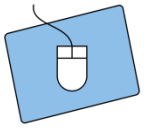
Außerdem hat eure Rakete eine **Position** auf dem Spielfeld. Diese könnt ihr in den Eigenschaften als **X- und Y-Koordinaten** ablesen bzw. verändern.



1. Wählt die Rakete unter den Komponenten aus.
2. Ändert unter den Eigenschaften die X- und Y-Koordinaten, und sucht euch eine schöne Startposition für die Rakete aus.

Die Asteroiden

Neben der Rakete braucht ihr noch zwei weitere **ZeichenAnimationen** für die **Asteroiden**.



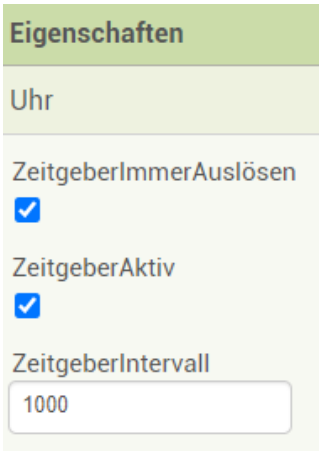
1. Verfahrt genau wie im vorherigen Abschnitt, und erstellt die ZeichenAnimationen an einer sinnvollen Position.
2. Testet das Gesamtbild eurer App.

Eine Uhr für das Spiel

Jetzt braucht ihr noch eine **Uhr**, die ihr in der **Palette** in der Kategorie **Sensoren** findet. Die Uhr misst die Zeit, die seit Spielbeginn vergangen ist.

Eine **Uhr** ist für viele Sachen gut. Ihr könnt z. B. ein **ZeitgeberIntervall** einstellen. Dieses löst, nach Ablauf der eingestellten Zeit, immer wieder die Funktion **Uhr.Zeitgeber** aus.

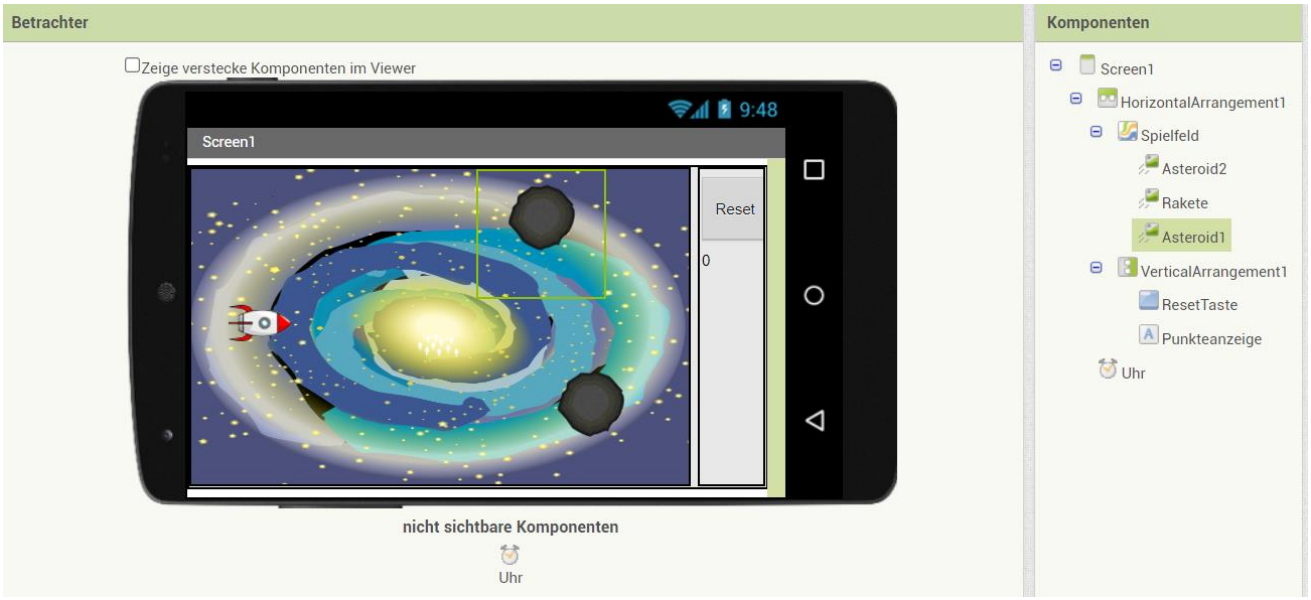
1. Zieht eine **Uhr** in eure App.
2. Wählt die Uhr unter den **Komponenten** aus, und ändert in den **Eigenschaften** den Wert des **ZeitgeberIntervalls** in 1000 ms (1000 ms entsprechen 1 Sekunde).



[8]

Ein Zwischenfazit

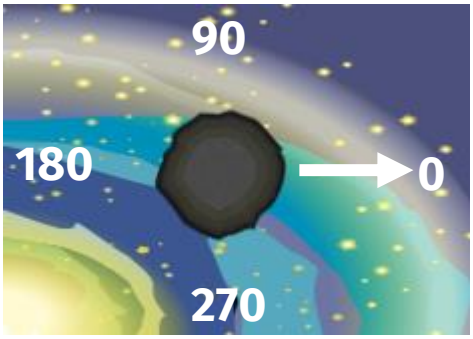
So oder so ähnlich sollte eure App jetzt aussehen. Falls ihr noch Fragen habt, sprecht mit einem Betreuer oder einer Betreuerin.



[9]

Die Bewegungsrichtung der Asteroiden

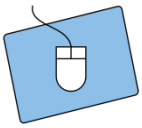
Die Bewegungsrichtung (im App Inventor kurz **Richtung**) einer ZeichenAnimation orientiert sich anhand eines Kreises. In der Abbildung rechts seht ihr, dass bei **0** die Flugrichtung **rechts** ist. Möglich sind alle Werte von 0 bis 359.



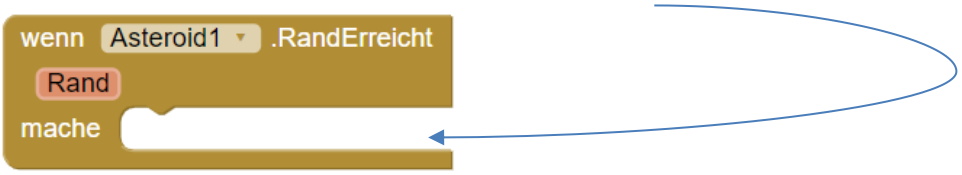
[10]

Als Erstes sollen sich die Asteroiden zufällig über die Spielfläche bewegen. Dazu müssen die folgenden beiden Bedingungen umgesetzt werden:

1. Die Asteroiden suchen sich zufällig ein neues Ziel aus.
2. Die Asteroiden prallen von den Rändern ab und fliegen weiter.

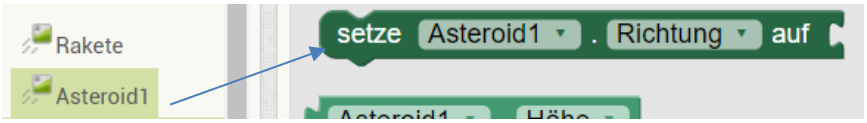


1. Jeder Asteroid hat eine **voreingestellte Bewegungsrichtung**. Diese Voreinstellung könnt ihr im **Designer-Editor** unter den **Einstellungen** nachsehen. Im App Inventor wird die voreingestellte Bewegungsrichtung kurz **Richtung** genannt.
2. Wechselt in den **Blöcke-Editor**.
3. Wenn der Asteroid gegen einen Rand prallt, soll die Flugrichtung geändert werden. Dazu haben alle ZeichenAnimationen folgende Funktion:
   Wenn Asteroid1 den Rand erreicht, dann wird etwas gemacht.



[11]

4. Sobald der Rand erreicht wird, soll die **Flugrichtung** geändert werden. Dazu gibt es den folgenden Block:



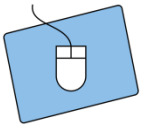
[12]

5. Kombiniert beide Blöcke, und hängt außerdem eine neue **Flugrichtung als Zahl** (Mathematik) an. Diese Zahl soll **zufällig** sein.
6. Wiederholt diese Schritte für den zweiten Asteroiden, und testet eure App.

Damit die Asteroiden sich richtig bewegen, fehlt noch etwas. Was das ist, das erfahrt ihr auf der nächsten Seite.

Die Geschwindigkeit der Asteroiden

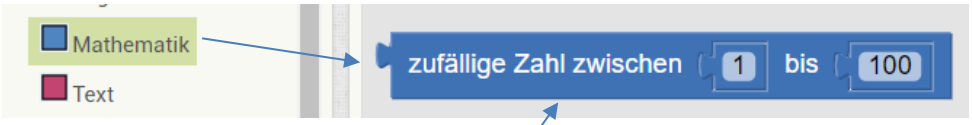
Bisher habt ihr nur die Flugrichtung der Asteroiden festgelegt. Was also noch fehlt, ist die Geschwindigkeit. Diese könnt ihr im **Designer-Editor** unter den **Einstellungen** zunächst **voreinstellen**.



1. Setzt einen beliebigen Wert für die Geschwindigkeit, und testet eure App.
2. Beobachtet das Verhalten der Asteroiden, und stellt eine passende Geschwindigkeit ein.

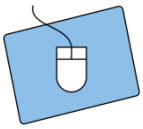
Die zufällige Flugrichtung der Asteroiden

Damit die Flugrichtung sich bei jeder Kollision ändert und auch immer zufällig ist, braucht ihr eine **Zufallszahl**. Die entsprechende Funktion findet ihr im **Blöcke-Editor** unter **Eingebaut → Mathematik**.



[13]

Diese Funktion wählt eine zufällige Zahl im Bereich von a (hier 0) bis b (hier 100) aus.



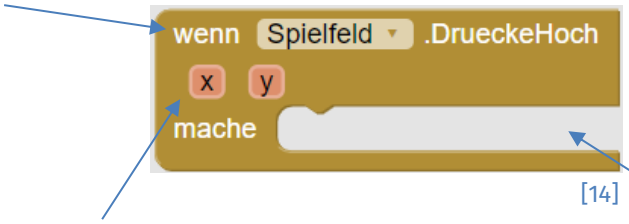
3. Überlegt:
   - An welcher Stelle müsst ihr die zufällige Zahl einbauen?
   - In welchem Bereich muss die Zufallszahl liegen?
4. Versucht, dies selbstständig umzusetzen, und testet eure App dabei regelmäßig. Zögert nicht, eine*n Betreuer*in bei Fragen anzusprechen.

Die Rakete steuern

Die Steuerung der Rakete soll durch **Berührung des Bildschirms** erfolgen und setzt sich ebenfalls aus zwei Dingen zusammen:

1. Die Koordinaten der Berührung müssen erkannt werden.
2. Die Rakete muss sich zu diesen Koordinaten ausrichten (Richtung) und schließlich in diese Richtung bewegen (Geschwindigkeit).

Die Berührung kann über das **Spielfeld** erkannt werden. Hierzu gibt es folgende Funktion:
Wenn das **Spielfeld gedrückt** wurden, dann…

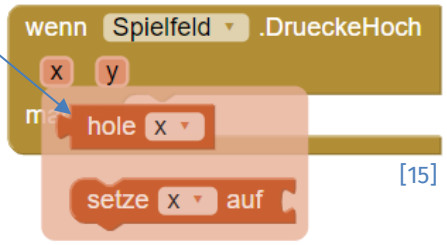


[14]

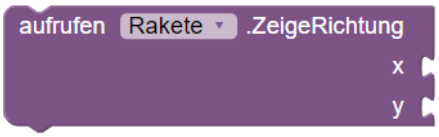
erkennt die Funktion die **X-** und **Y-Koordinaten** und kann mit diesen etwas **machen**.

Um zu erreichen, dass die Rakete in Richtung der Koordinate fliegt, an der ihr den Finger wieder vom Bildschirm hochgehoben habt, müsst ihr die Funktion **Wenn-Spielfeld.DrueckeHoch** und nicht **Wenn-Spielfeld.Gedrueckt** verwenden.

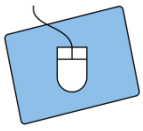
Um die Koordinaten nutzen zu können, müsst ihr den Mauszeiger über die entsprechende Koordinate bewegen. Wählt dann den **hole**-Block aus.



[15]

Jetzt braucht ihr die Funktion 

[16], an die ihr zwei Koordinaten anhängen könnt. Ihr findet diese in den Funktionen der Rakete.

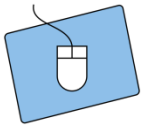


1. Benutzt die obige Funktion, und ändert die **Flugrichtung** der Rakete auf die **X- und Y-Koordinate** der Berührung.
2. Setzt die **Geschwindigkeit** der Rakete in den Einstellungen (Designer-Editor).
3. Testet eure App. Wenn die Steuerung funktioniert, könnt ihr direkt weiterarbeiten.

Super, wieder ein gutes Stück geschafft! Jetzt geht's zum Endspurt.

Den Punktestand/Sekundenzähler erhöhen

Um die Zeit bzw. die Punkte hochzuzählen, müsst ihr in der Funktion **Uhr.Zeitgeber** dafür sorgen, dass die **Bezeichnung** für den **Punktestand** geändert wird. Zum Hochzählen der Zeit erhöht ihr jedes Mal den **Zahlenwert** der Bezeichnung um 1.



Verwendet die folgenden Blöcke, um zu erreichen, dass die Sekunden richtig hochgezählt werden. Testet dann eure App.

[17]

[18]

[19]

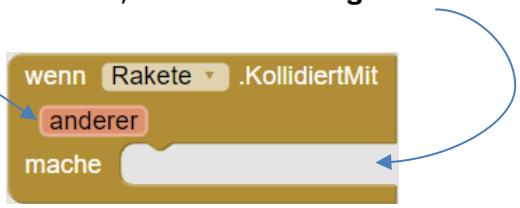
[20]

Die Kollision mit einem Asteroiden

Als Nächstes kümmert ihr euch um die Kollision mit einem Asteroiden. Dazu müsst ihr die folgenden Dinge umsetzen:

1. Es muss abgefragt werden, **ob** eine **Kollision** stattgefunden hat.
2. Wenn eine Kollision stattgefunden hat, soll die Rakete **explodieren**.
3. Außerdem muss das **Spiel gestoppt** werden.

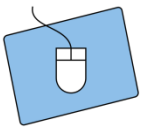
Jede ZeichenAnimation kann abfragen, ob sie mit einer anderen ZeichenAnimation kollidiert.
Wenn die **Rakete** mit **etwas** (anderer) **kollidiert**, dann kann etwas **gemacht** werden.



[21]

Sobald dieser Fall eintritt, soll das Bild der Rakete in eine Explosion umgeändert werden, und die Geschwindigkeit aller ZeichenAnimationen soll auf 0 gesetzt werden.

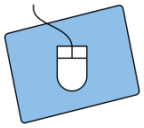
Weiter geht es auf der nächsten Seite…



1. Ladet aus dem bekannten Ordner das **Bild einer Explosion** hoch.
2. Sucht im Blöcke-Editor bei der **Rakete** eine Funktion, mit der ihr das **Bild ändern** könnt.
3. Hängt an diese Funktion einen **Text** an. Dieser Text soll den Namen der Explosions-Datei haben, also z. B. explosion1.png.
4. Setzt die **Geschwindigkeit** aller ZeichenAnimationen auf 0.
5. Testet eure App.

Die Reset-Taste

Als Letztes müsst ihr dafür sorgen, dass das Spiel zurückgesetzt werden kann. Dazu braucht ihr die Reset-Taste.

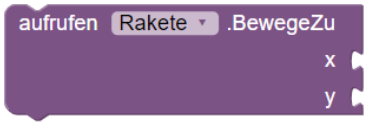


1. **Wenn** die **Reset-Taste** gedrückt wird, **dann** sollen die **Grundeinstellungen** wiederhergestellt werden. Verwendet deshalb zunächst folgende Funktion:

[22]

2. Überlegt euch nun, was die **Grundeinstellungen** sind und dementsprechend in den **mache**-Teil gehört. Hierbei können euch die folgenden **Hinweise** helfen:
   - Der **Punktestand** hat eine Grundeinstellung.
   - Die **Rakete** hat ein **Startbild**, eine **Startposition** und eine **Startgeschwindigkeit**.
   - Die **Asteroiden** haben **Startpositionen**.
3. Setzt eure Ideen aus Aufgabe 2 in der **Wenn-ResetTaste.Klick-Funktion** um.

Um die Position einer ZeichenAnimation zu setzen, gibt es die Funktion



[23].

*Gratulation ☺. Damit habt ihr ein voll funktionsfähiges Asteroids-Spiel programmiert. Im Folgenden findet ihr noch ein paar Tipps, Hinweise und Anregungen, um das Spiel zu erweitern. Wenn ihr lieber ein neues Spiel programmieren möchtet, dann meldet euch bei den Betreuer*innen.*

Erweiterungen

Hier findet ihr einige Ideen, um das Spiel noch zu erweitern.

Mehrere Raketen und Hintergründe zur Auswahl

Um diesen Effekt zu erzielen, könnt ihr für jeden Hintergrund und jede Rakete eine eigene Taste (mit dem jeweiligen Bild darauf) anlegen. Wenn ihr diese Taste anklickt, soll sich die ZeichenAnimation in das neuausgewählte Bild ändern.

Höhere Schwierigkeitsgrade

Auch um mehrere Schwierigkeitsgrade zur Auswahl anzubieten, könnt ihr mit Tasten arbeiten. Je höher der Schwierigkeitsgrad, umso schneller werden die Asteroiden und umso langsam eure Rakete.

Ein dritter Asteroid

Ihr könnt nach einer bestimmten Zeit einen dritten Asteroiden auftauchen lassen. Dazu müsst ihr euch die aktuelle Zeit anschauen (also den Wert der Bezeichnung). Sobald diese einen bestimmten Wert erreicht hat, macht ihr einen dritten Asteroiden sichtbar. Das bedeutet, dass der dritte Asteroid bereits zu Beginn des Spiels existiert, dann allerdings noch unsichtbar ist.

Ein Spielende

... wäre doch sehr schön. Dazu könnt ihr eine weitere Bezeichnung hinzufügen, die beim Start 100 anzeigt. Jedes Mal, wenn der Uhr.Zeitgeber auslöst, reduziert ihr den Wert um 1. Wenn der Wert 0 erreicht wird, könnt ihr das Spiel resetten oder einen Text ausgeben, der sagt, wie viele Punkte der Spieler oder die Spielerin erhalten hat.

Quellenverzeichnis:
Abb. 1 – *Quelle: ClipSafari, Autor: purzen (https://www.clipsafari.com/clips/o28806-cartoon-rocket), CC0 1.0 (https://creativecommons.org/publicdomain/zero/1.0/), abgerufen am: 07.03.2023.*
Abb. 2 – *Quelle: ClipSafari, Autor: presquesage (https://www.clipsafari.com/clips/o191215-colorful-galaxy,), CC0 1.0 (https://creativecommons.org/publicdomain/zero/1.0/), abgerufen am: 07.03.2023.*
Abb. 3 *Quelle: ClipSafari, Autor: roboxman (https://www.clipsafari.com/clips/o193397-black-asteroid), CC0 1.0 (https://creativecommons.org/publicdomain/zero/1.0/), abgerufen am: 07.03.2023.*
Abb. 4 bis 23 – *Quelle: Screenshot des MIT App Inventor 2 (http://ai2.appinventor.mit.edu/?locale=de), CC BY-SA 4.0 (https://creativecommons.org/licenses/by-sa/4.0/), erstellt am: 07.03.2023.*
 – *Quelle: InfoSphere, CC BY-SA 4.0 Attribution-ShareAlike 4.0 International (https://creativecommons.org/licenses/by-sa/4.0/).*