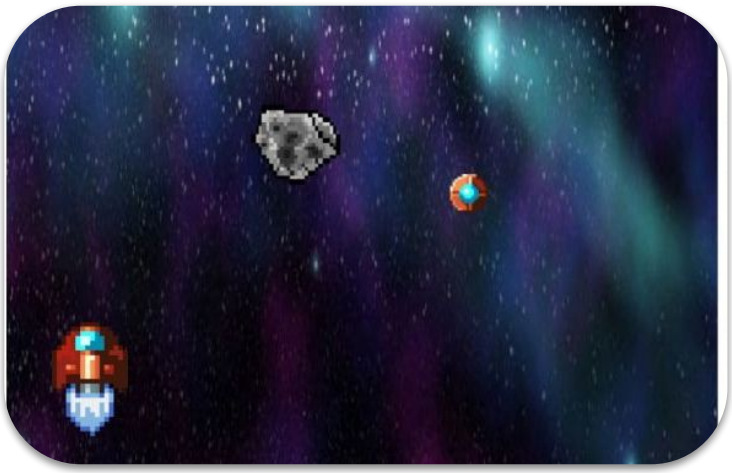
Übersicht: Game Lernkarten

Die Story

Bei einem Transport im Weltall ist leider die Fracht verschüttet worden. Eine mutige Raumfahrerin und ein mutiger Raumfahrer sind bereit, die Fracht zurückzuholen!

Die Aufgabe ist nicht einfach - im Weltall lauern Asteoriden, magische Figuren und die Gefahr, dass der Treibstoff ausgeht. Schaffst du es, so viel Fracht wie möglich zurückzubringen?



Zu diesem Spiel gehören:

- Die Assets (Bilddateien) im Ordner **Space Traveller > Assets**
- Alle Lernkarten, die mit **2020.Thunkable.Cheatsheet.AppDesign.Game.*** benannt sind
- Das fertige Spiel kann unter https://space.youthhackathon.com geöffnet und gespielt werden - auch ein Re-mix des Spiels ist möglich.

Zusätzlich gibt es auch noch die folgenden Unterlagen:

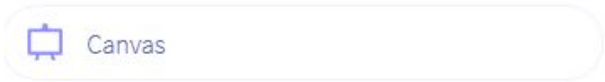
- Lernkarten mit allgemeinen Erklärungen
- Lernkarten zum Layouting
- Lernkarten zu Standard Apps (z.B. mit Maps, Übersetzer-Apps etc.)

thunkable

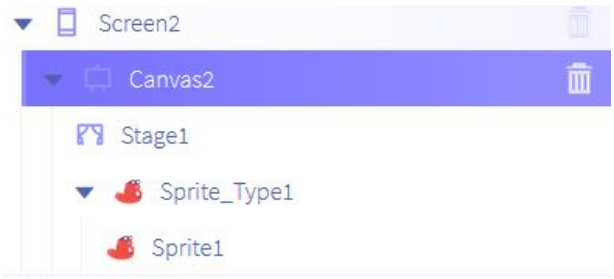
Eine Spielfigur erstellen 1/2

Ein Rahmen für die Spielfigur (Canvas)

Bevor wir überhaupt eine Spielfigur erstellen können, benötigen wir einen Rahmen. Zieh dazu die Komponente “**Canvas**” auf den Screen.



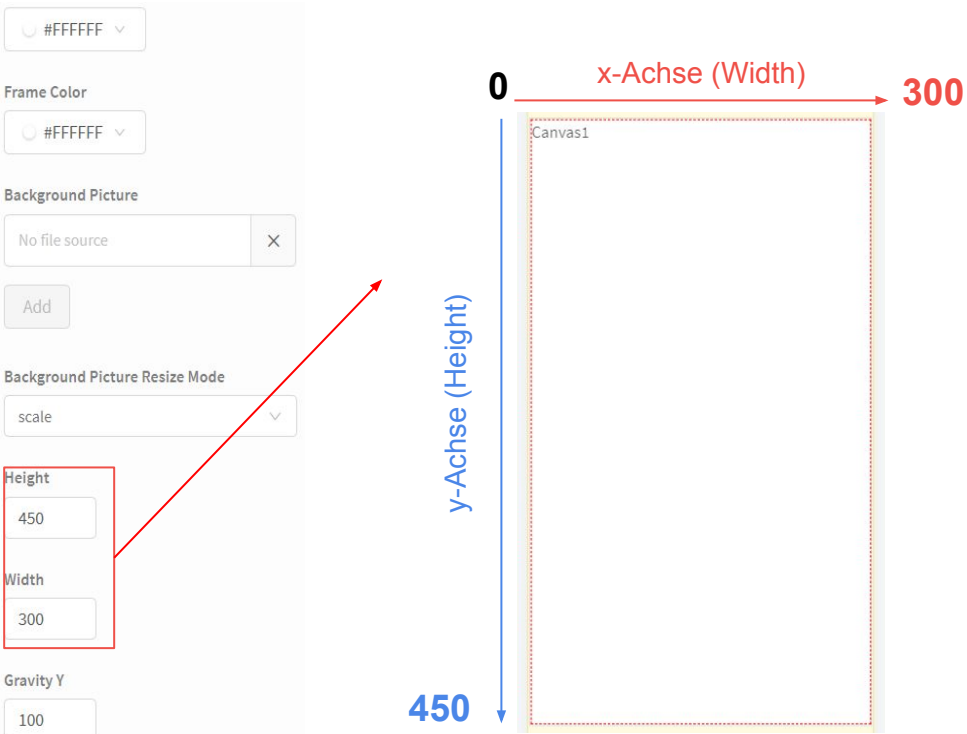
Ein **Canvas** kann man sich wie eine Leinwand vorstellen, auf der dein Spiel stattfindet. Ein **Canvas** ist touch-sensitiv, das heißt er erkennt sämtliche Berührungen durch die Finger. Wenn man einen Canvas auf einen Screen zieht, werden auch eine **Stage** (Bühne) und ein **Sprite Type** (die **Spielfigur**) erstellt.



Stage & Koordinatensystem für deine Spielfigur>

Die **Stage** ist die Bühne in deinem Spiel. In einer Stage kann man eine **Hintergrundfarbe** oder ein **Hintergrundbild** einstellen und auch die **Breite und Höhe** im Screen einstellen.

Height (Höhe) und **Width (Breite)** sind wichtige Einstellungen im Spiel, da sie das Koordinatensystem im Spiel bestimmen. Die Standardwerte sind eine **Höhe** von **450** und eine **Breite** von **300** Pixel.

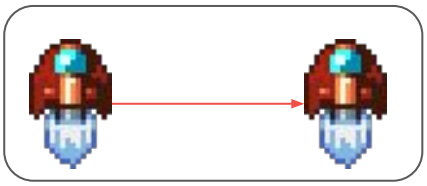


thunkable

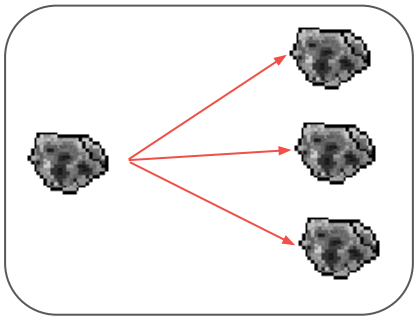
Eine Spielfigur erstellen 2/2

Jede Szene hat verschiedene “Figuren” und das sind die **Sprites.** Diese sind Bilder, die auf der Szene interagieren können, zum Beispiel eine Spielfigur, Gegner oder sammelbare Objekte.

Sprite Types repräsentieren die **Eigenschaften** von Sprites - sie sind nicht direkt auf der Stage zu sehen. Zum Beispiel kann es einen **Sprite Type** “Gegner” oder “Spieler” geben. Aus diesem Sprite Type lassen sich dann **konkrete** Sprites erstellen, die in der Szene agieren. Ein Sprite Type “Gegner” kann dann zum Beispiel immer wieder Gegner-Sprites erstellen. Du kannst sie dir wie einen **Stempel** vorstellen.



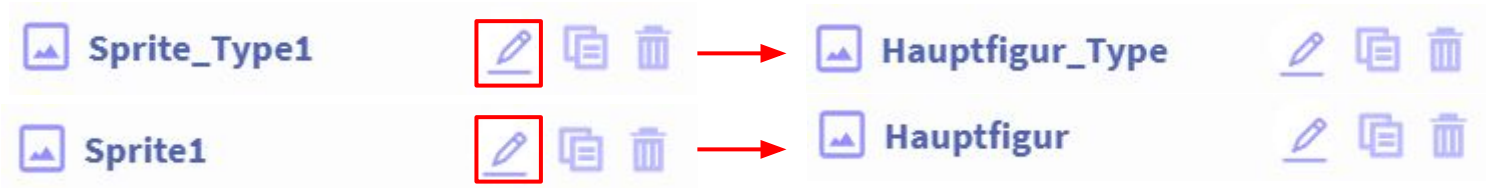
Sprite Type mit **einem** Sprite (z.B. Hauptfigur)



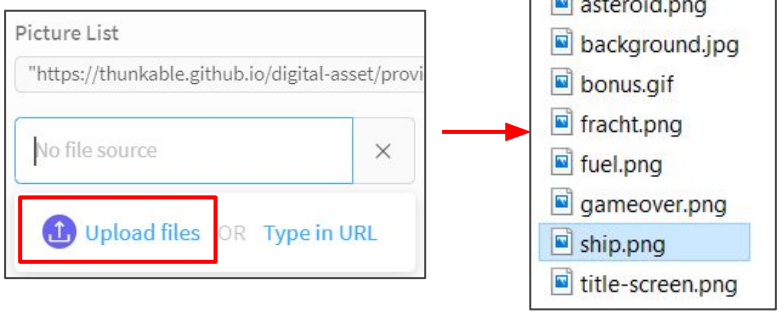
Sprite Type mit **mehreren** Sprites (z.B. Gegner oder Sachen zum Einsammeln)

Sprite_Type1 und Sprite1 umbenennen

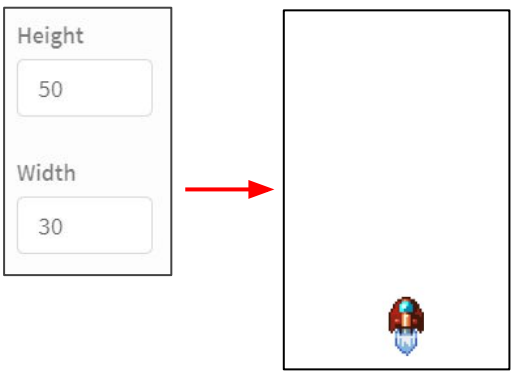
Bennenne jetzt Sprite_Type1 in “**Hauptfigur_Type**” und Sprite1 in “**Hauptfigur**” um.



Wähle jetzt den “**Hauptfigur_Type**” in der Hierarchie links aus. In den Properties rechts kannst du jetzt eine **Bilddatei** für deine Spielfigur hochladen:



Wähle jetzt die “**Hauptfigur**” (die ist am Screen sichtbar) in der Hierarchie links aus. In den Properties rechts kannst du jetzt auch noch die **Größe** angeben:

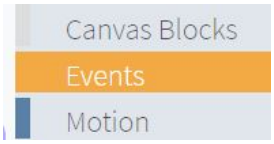


Tipp: Mit der Maus kannst du die Figur noch an die passende Stelle ziehen.

thunkable

Steuern der Spielfigur nach links und rechts

Wenn ein **Canvas** erzeugt wurde werden unter Blocks die Canvas Blocks zum Programmieren des Spiels angezeigt:



In der Kategorie Events unter Canvas Blocks findest du das Event **“when pointer is down on Canvas1”** - eskann Berührungen am Touchscreen erkennen:

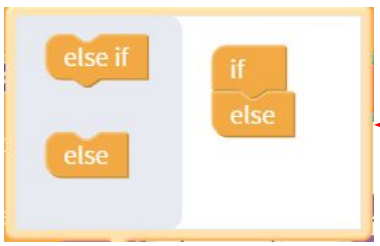
Als Output hat es einen **pointer.** Dieser speichert den Punkt, der am Touchscreen gedrückt wurde. Diesen kann man nutzen, um eine Steuerung für das Spiel zu programmieren:

Überlegungen zur Steuerung

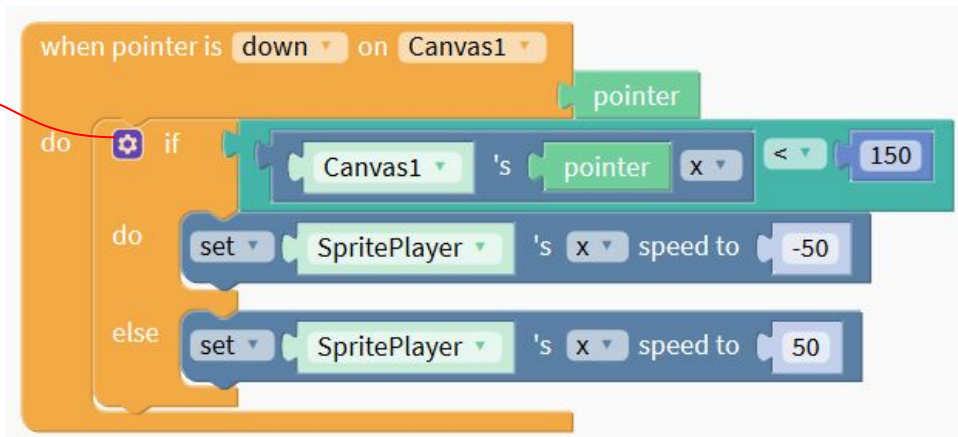
Für unser Beispiel haben wir uns entschieden, dass ein Touch auf die linke Bildschirmhälfte das Raumschiff nach links bewegt und ein Klick auf die rechte Hälfte nach rechts.

Das Spiel ist 300 Pixel breit, das heißt jeder pointer, dessen x-Position kleiner ist als 150 Pixel wird als “links” gewertet.

Diese Information kann man nutzen, um den Spieler zu steuern.



Mit dem Zahnrad kannst du dir aus einem **if-** einen **if-else-Block** zusammenstellen.

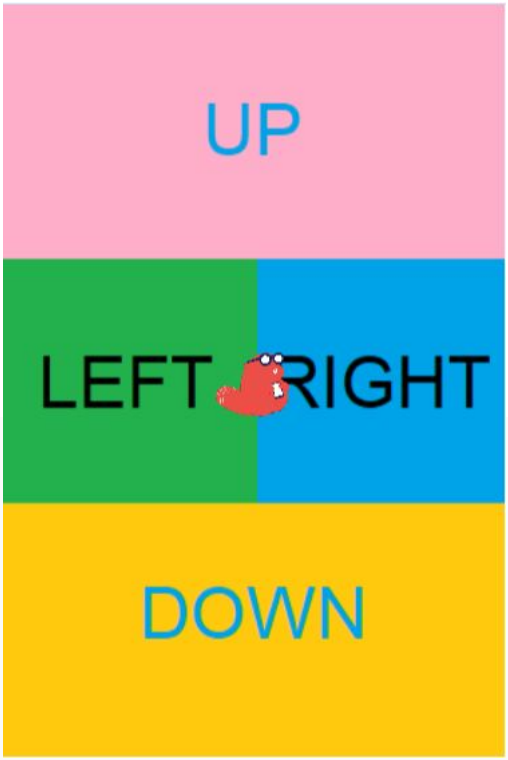


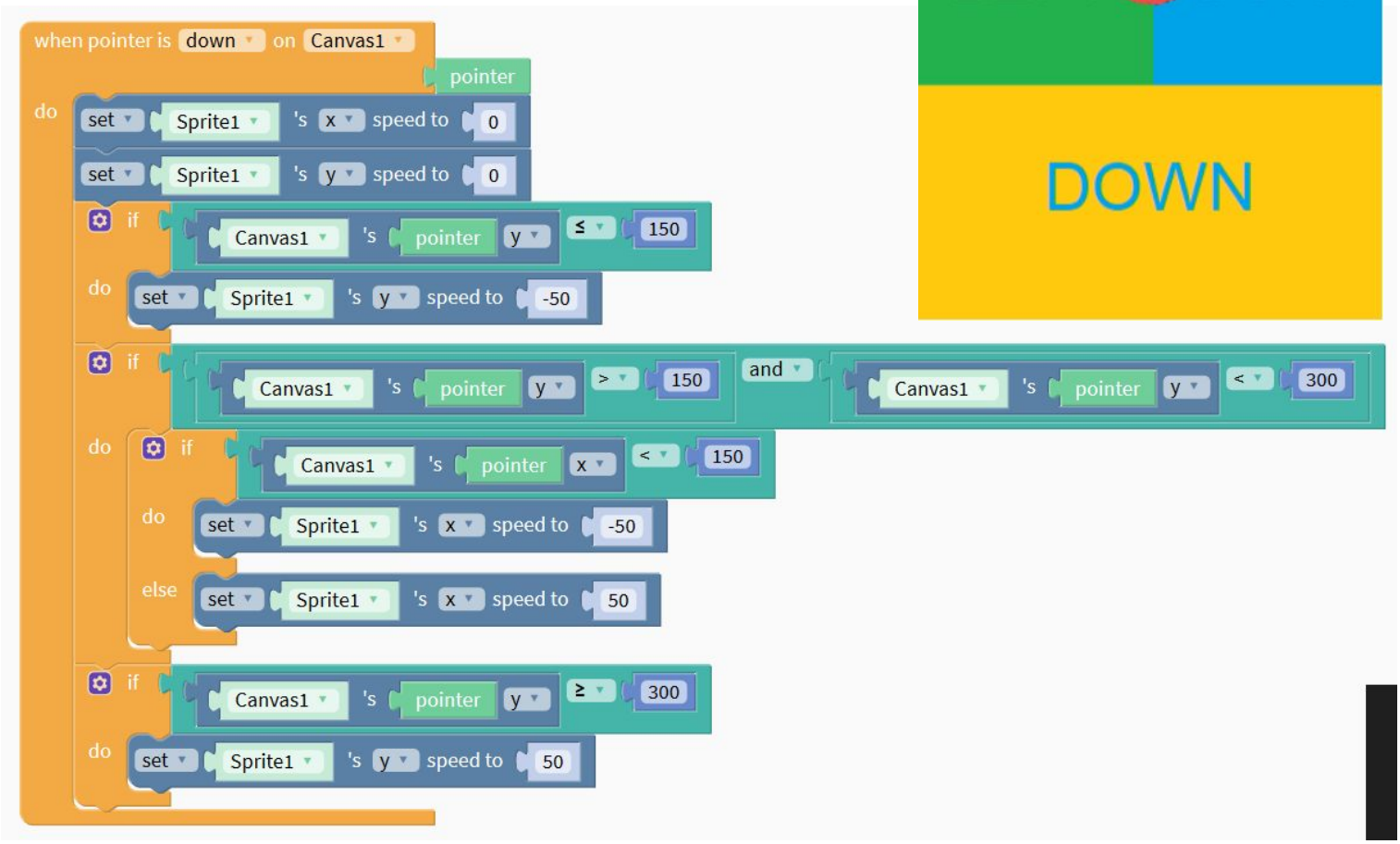
thunkable

Steuern der Spielfigur in alle Richtungen (↑→←↓)

Lies zunächst die Karte “Steuern der Spielfigur nach links und rechts”, damit du verstehst, was hier passiert.

Hier wird gezeigt, wie du deine Blöcke so erweitern kannst, dass deine Spielfigur sich durch einen Touch in den oberen oder unteren Bereich zusätzlich zu links und rechts auch noch nach oben und unten bewegen kann.

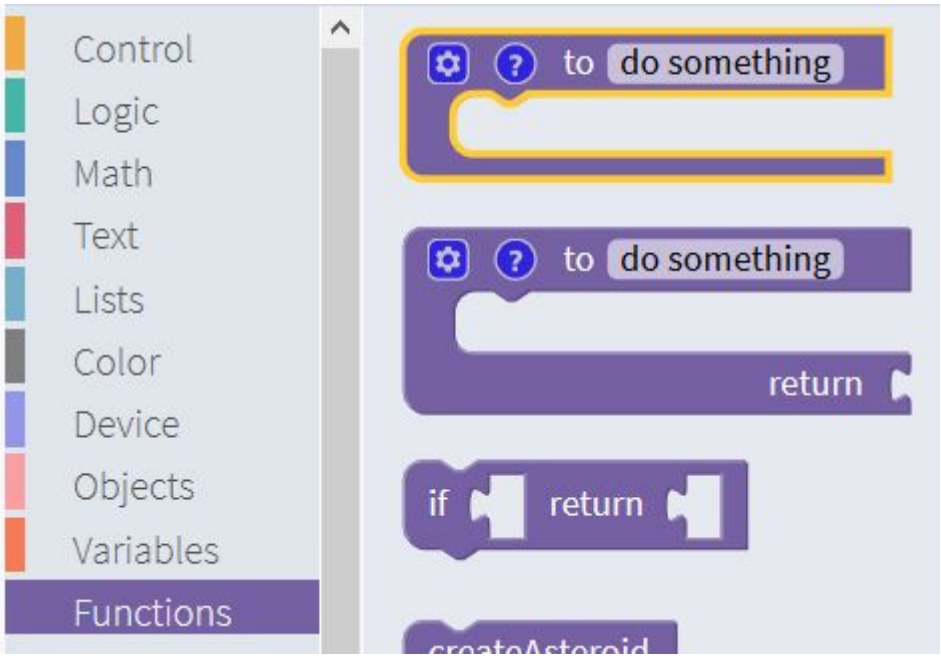




thunkable

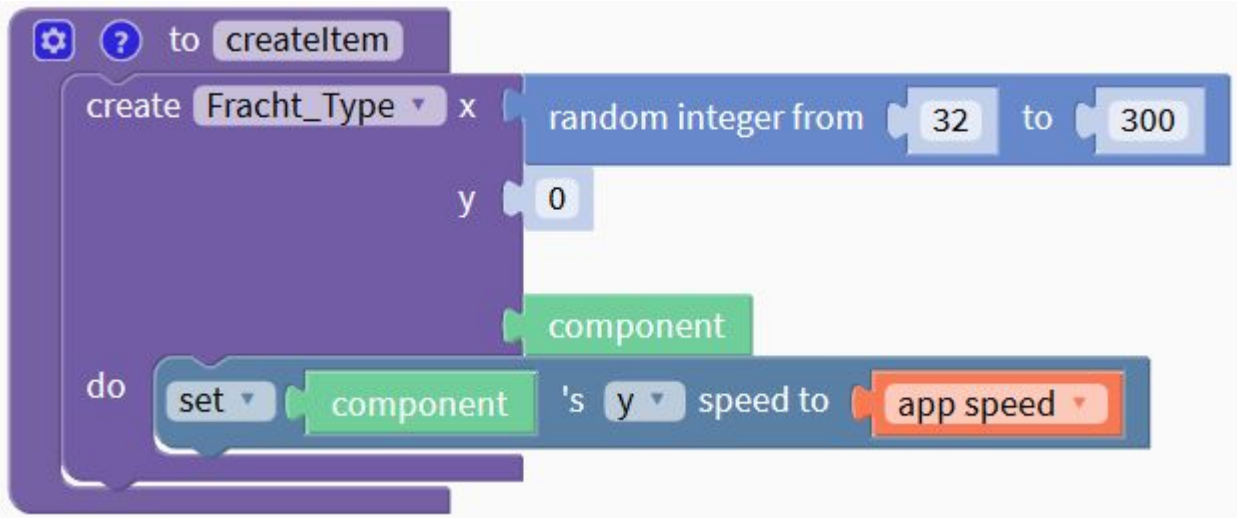
Funktionen

Funktionen sind ein Element der Programmierung welches dir erlauben, eine Abfolge an Befehlen an verschiedenen Stellen im Programm zu verwenden. Außerdem macht es deinen Code viel lesbarer. Du findest Funktionen in den Blöcken.



Eine Funktion schreiben

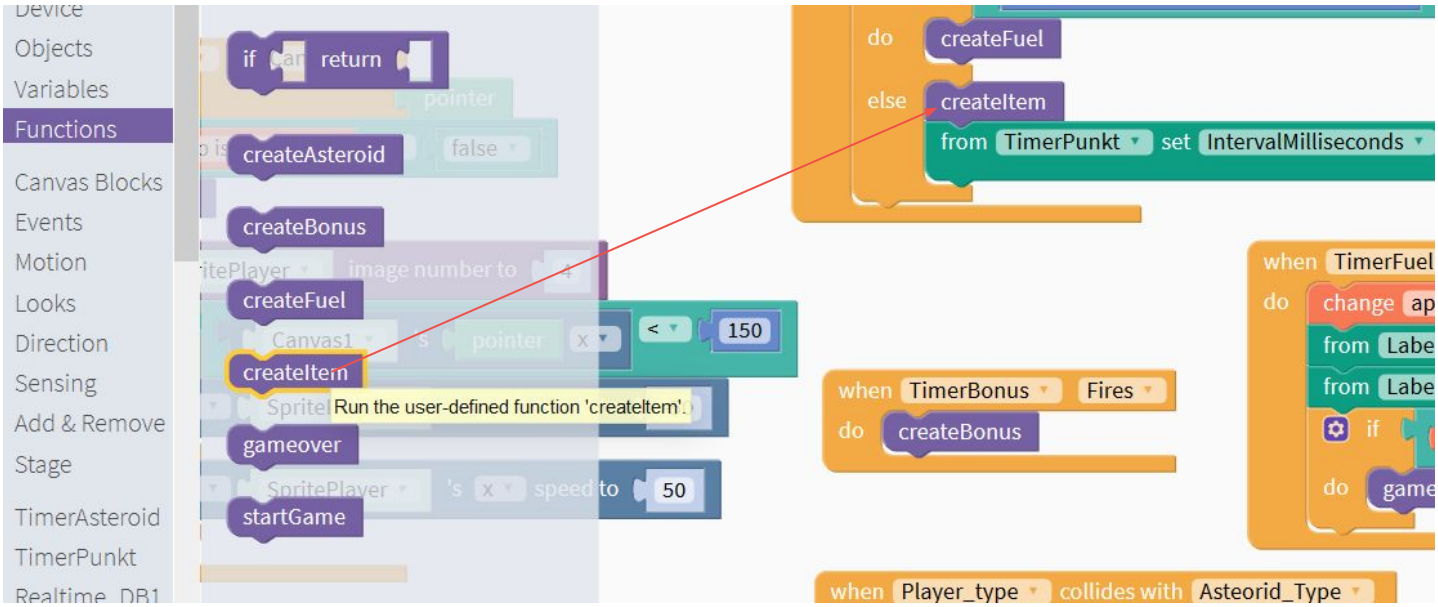
Als erstes musst du deiner Funktion einen **Namen** geben. Dieser sollte beschreiben, was die Funktion macht. Danach kannst du gewohnt deine Blöcke aneinanderreihen. Beispielsweise erzeugt die Funktion **createItem** in unserem Spiel eine Fracht und setzt deren Bewegung nach unten.



Funktionen

Eine Funktion anwenden

Deine erzeugte Funktion findest du in den Blöcken unter **Functions.** Du brauchst diese nur noch in eine geeignete Stelle ziehen!



Tipps & Anwendungen für Funktionen

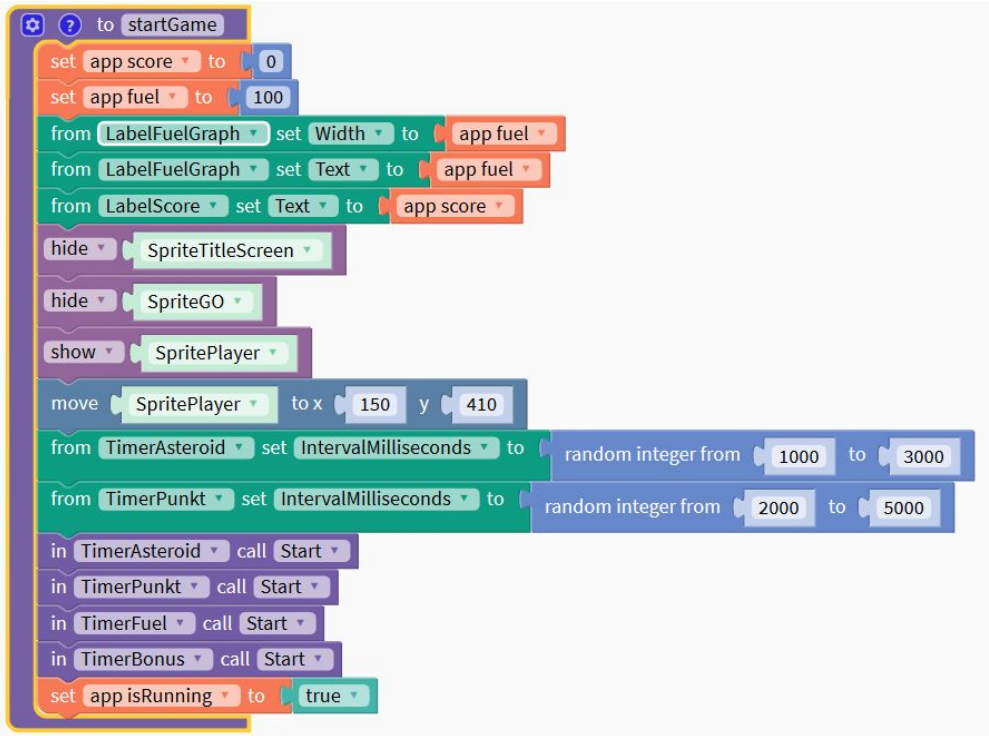
Funktionen erleichtern den Umgang mit deinem Code, da er nur an einer definierten Stelle vorkommt. Wenn du also etwas ändern musst, musst du nicht an vielen Stellen etwas bearbeiten sondern nur an einer!

Auf der nächsten Seite findest du einige Anwendungen für Funktionen, die wir im Spiel “Space Traveller” verwendet haben.

Funktionen - Beispiele

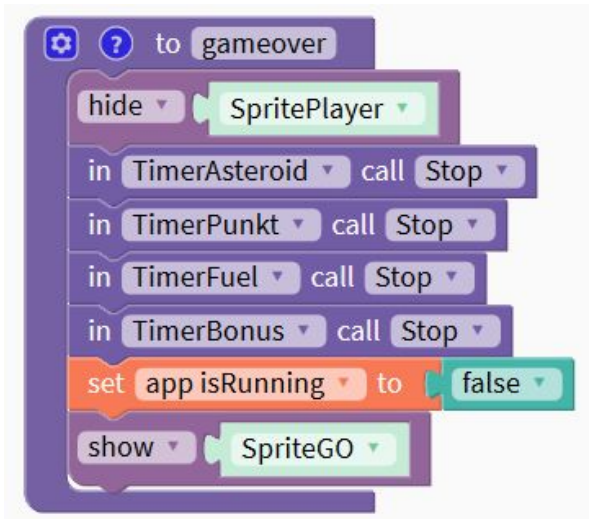
Spiel starten

Eine gute Anwendung ist zum Beispiel das Initialisieren (“Startkonfiguration”) des Spiels. Hier kannst du eine Funktion “startGame” nutzen, um alle Variablen zurückzusetzen und die Spieler richtig zu platzieren.



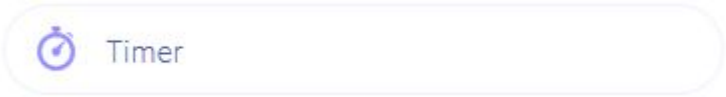
Spiel beenden (game over)

Space Traveller verwendet zum Beispiel eine Funktion “gameover” für das Beenden des Spiels. Das Spiel kann auf mehrere Arten Enden (Kollision mit Asteoriden, Tank leer), also ist es gut, dafür eine Funktion zu verwenden.



Sachen zum Einsammeln 1/2

Die Komponente **Timer** ermöglicht es, gewisse Ereignisse nach einer abgelaufenen Zeit auszulösen. Zum Beispiel ist dies nützlich, wenn du alle paar Sekunden ein Objekte (Sachen, die Punkte bringen oder Gegner) zum Einsammeln oder Ausweichen erstellen willst:



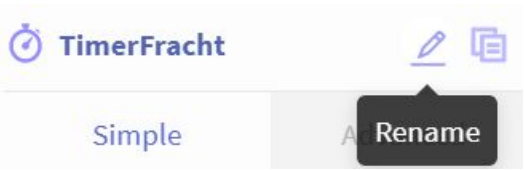
Den Timer kannst du rechts mit den Eigenschaften konfigurieren.

Setze das Intervall z.B. auf **5 Sekunden**.

Schalte **Loops** auf **true**, damit der Timer nach 5 Sekunden wieder von vorn startet.

Wichtig: beim Spielstart musst du den Timer **starten**. Das kannst du so machen:

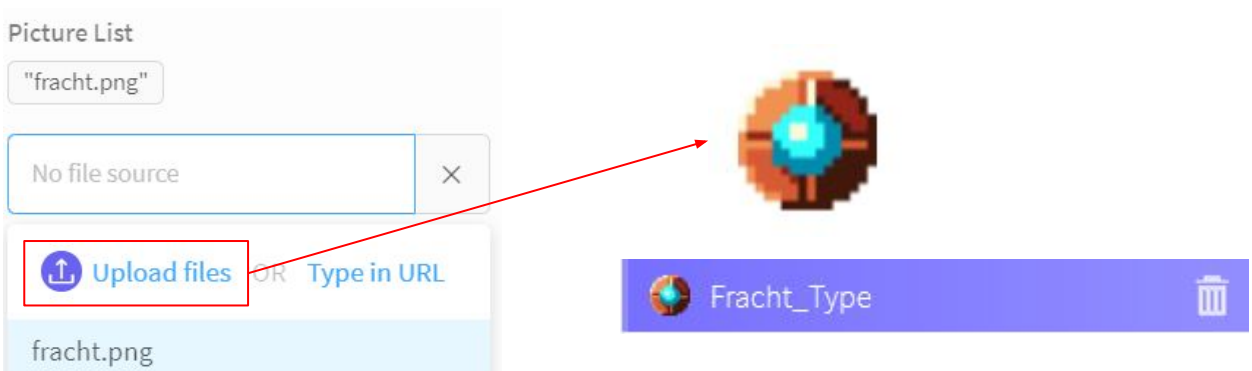
Benenne den Timer jetzt (rechts in den Eigenschaften) um, so dass man an seinem Namen erkennt, welches Objekt er steuert, z.B. **TimerFracht** für die Fracht, die vom Raumschiff eingesammelt wird:



Jetzt müssen wir einen neuen Sprite Type hinzufügen. Das kannst du machen, indem zu zuerst auf die Stage klickst und dann unten auf “**Add Sprite Type**”:

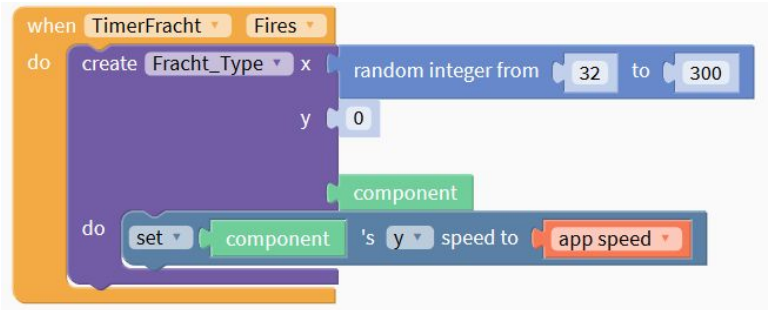
Den Sprite Type solltest du so umbenennen, wie die Sache heißt, die hinunterfällt. Bei Fracht z.B. “**Fracht Type**”.

Jetzt kannst du ein **Bild** für den Sprite Type **hochladen**. Klicke auf Fracht Type und lade rechts bei den Eigenschaften ein Bild hoch (z.B. fracht.png aus den Assets zum Spiel).

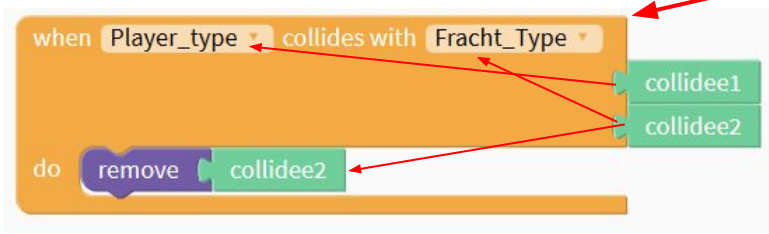


Sachen zum Einsammeln 2/2

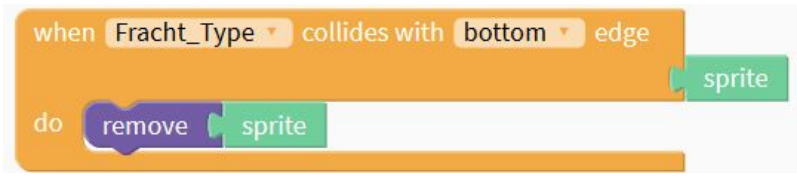
In unserem Spiel wird das Objekt an einer Zufälligen x-Position ganz oben erzeugt, nachdem ein Timer abgelaufen ist.



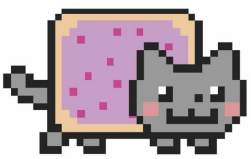
Als nächstes muss noch programmiert werden, was bei einem Zusammenstoß (Kollision) des sammelbaren Objekts und dem Spieler passiert. Das passiert durch den **"when … colides with …"** Block.



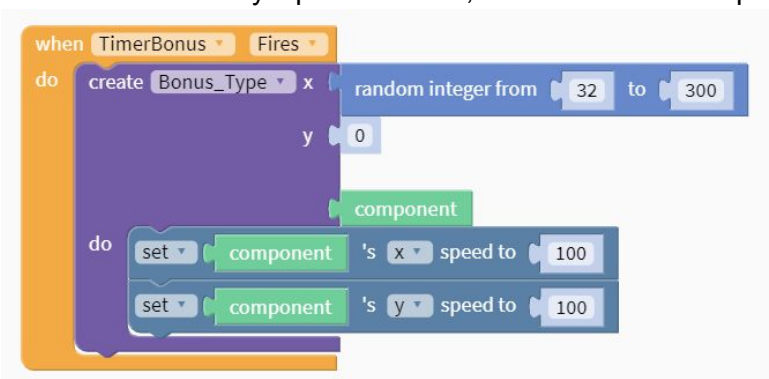
Der Block hat zwei Ergebnisse: **colidee1** und **colidee2**. Diese referenzieren die jeweiligen im Block gewählten Sprites. Man kann nun diese Referenzen nutzen, um einen Colidee zu **entfernen** oder etwas anderes mit ihm zu machen (Geschwindigkeit ändern, Richtung ändern etc.)

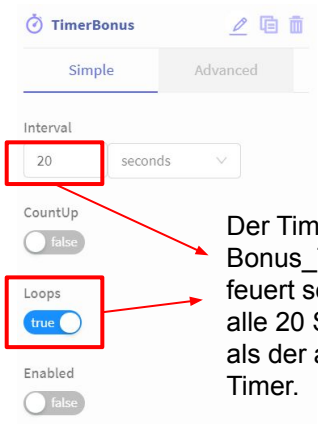


Wenn das Objekt den **unteren Rand** berührt soll es ebenfalls **entfernt** werden.

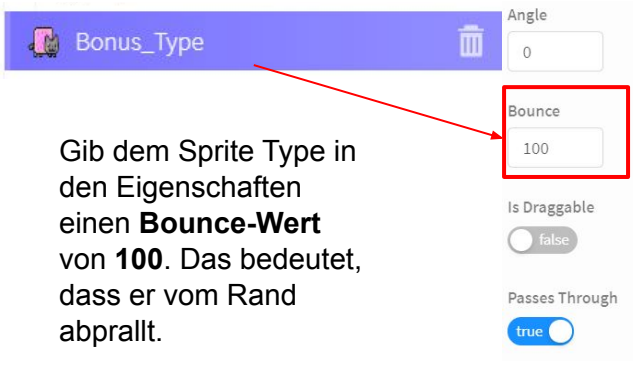


Weitere Sprite Types (z.B. für **Bonuspunkte**) kannst du z.B. auch schräg herunterfallen lassen. Dafür musst wieder einen neuen Timer erzeugen, den Timer starten und sowohl die x- als auch die y-Speed setzen, wenn der zum Sprite gehörende Timer feuert:





Der Timer für die Bonus_Type-Figuren feuert seltener (nur alle 20 Sekunden) als der andere Timer.

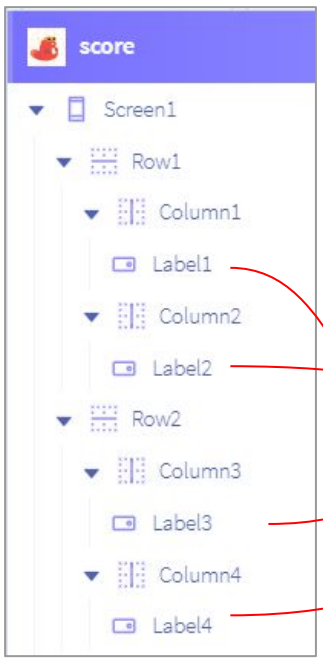


Gib dem Sprite Type in den Eigenschaften einen **Bounce-Wert** von **100**. Das bedeutet, dass er vom Rand abprallt.

***Tipp:** Wie du Punkte und Highscores zählen kannst, findest du auf der Lernkarte **Score und Highscore**.*

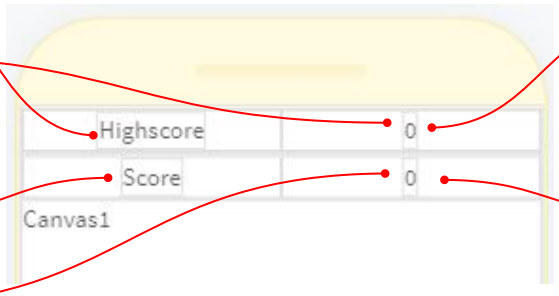
Score & High Score

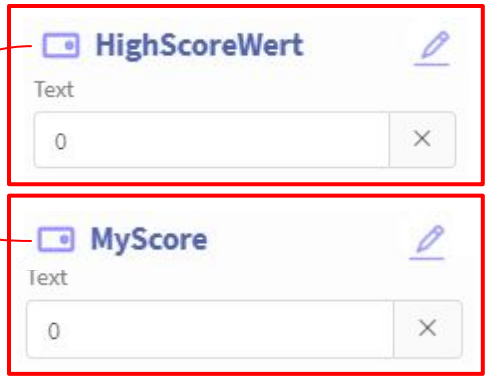
Du möchtest in deinem Game Punkt zählen (Score) und einen Highscore ausgeben, dann findest du hier die Anleitung dazu.



Score und Highscore anzeigen

Für die Anzeige der Punkte benötigst du zunächst auf dem Screen einen Bereich mit Reihen (*Row*) und Spalten (*Column*). In jeder Zelle werden Beschriftungen (*Labels*) eingefügt und wie folgt benannt:
Im Label ***HighScoreWert*** sollen die High Score Werte angezeigt werden und im Label ***MyScore*** wird die eigene Punktzahl angezeigt.

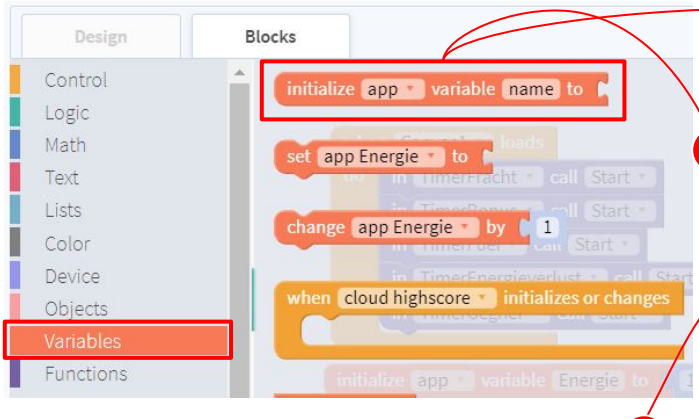


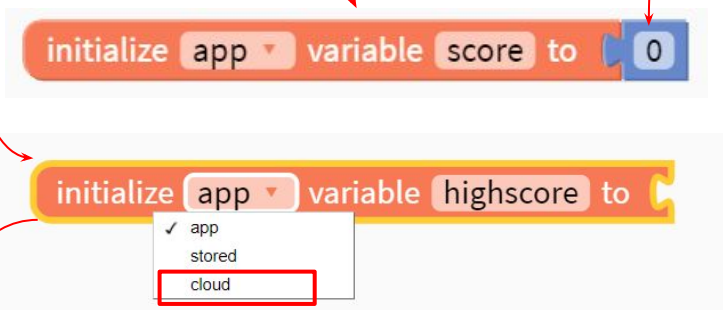


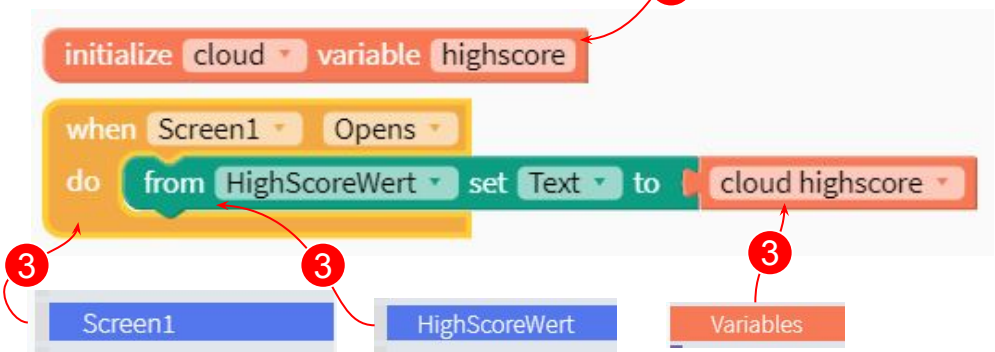
Punktezahl in Variablen Speichern

Für das Zählen und Speichern der Punktzahl (Score und Highscore) müssen zwei Variablen ***Score*** und ***highscore*** angelegt werden.

1. Die Variable ***Score*** wird mit 0 initialisiert (= belegt)
2. Die Variable ***HighScore*** wird als *cloud* Variable angelegt.







3. Das Label ***HighScoreWert*** wird zu Spielbeginn mit dem Wert der Cloud Variable ***HighScore*** belegt.

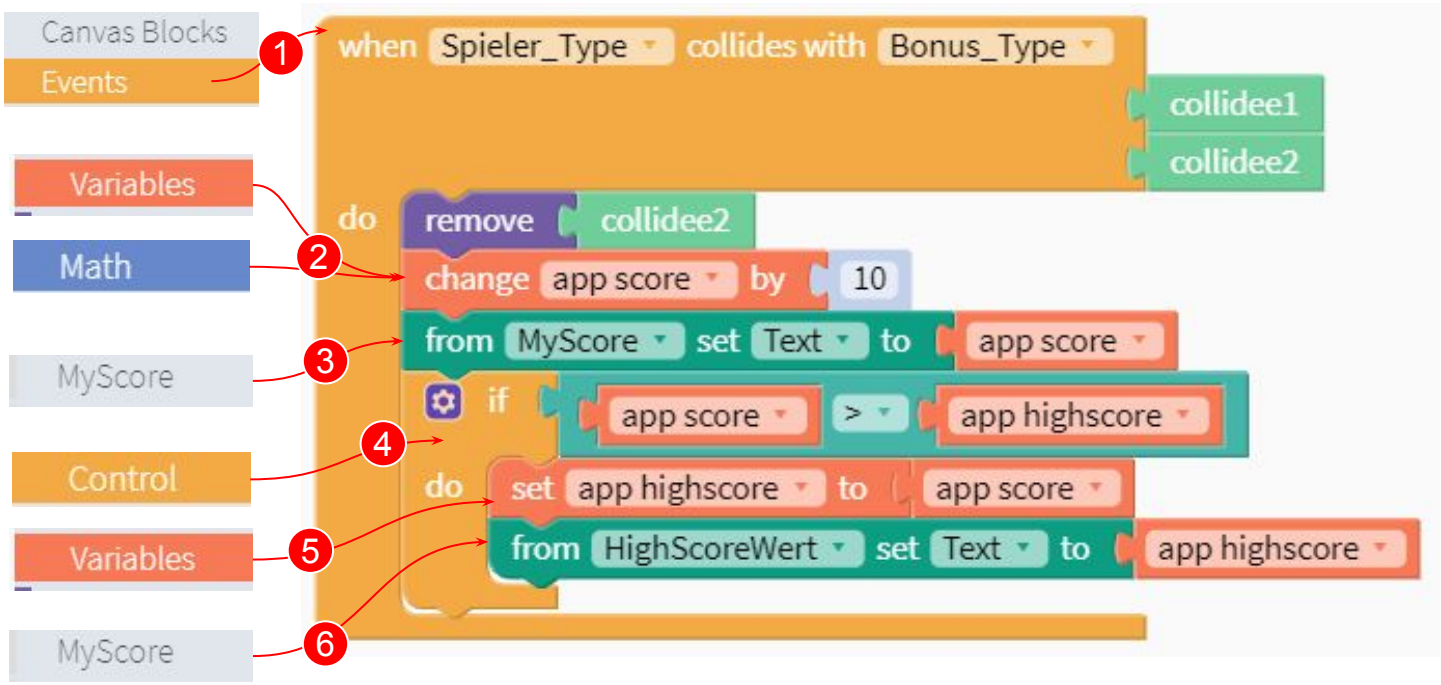
thunkable

Score & High Score

Erhöhen der Punktezahl

Sollen während des Spiels Punkte gesammelt werden

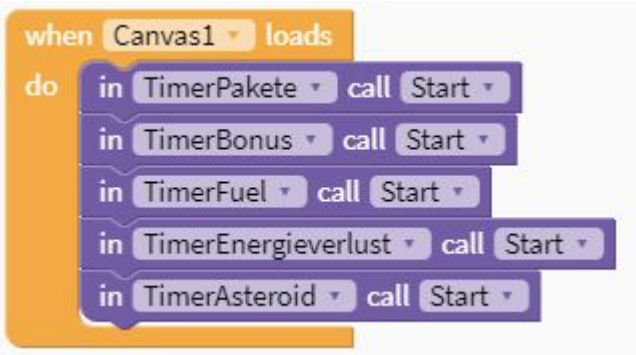
1. z.B. wenn SpielerInnen Elemente vom Typ ***Bonus_Type*** einsammeln
2. so muss die Variable ***Score*** (z.B. um 10 Punkte) erhöht werden.
3. Für die Aktualisierung der Punkteanzeige muss dem Label ***MyScore*** die Variable ***Score*** zugewiesen werden.
4. Sollte die aktuelle Punktzahl den Highscore übersteigen,
5. so werden der Variable ***HighScore*** und
6. dem Label ***HighScoreWert*** die aktuelle Punktzahl zugewiesen.



Gegner

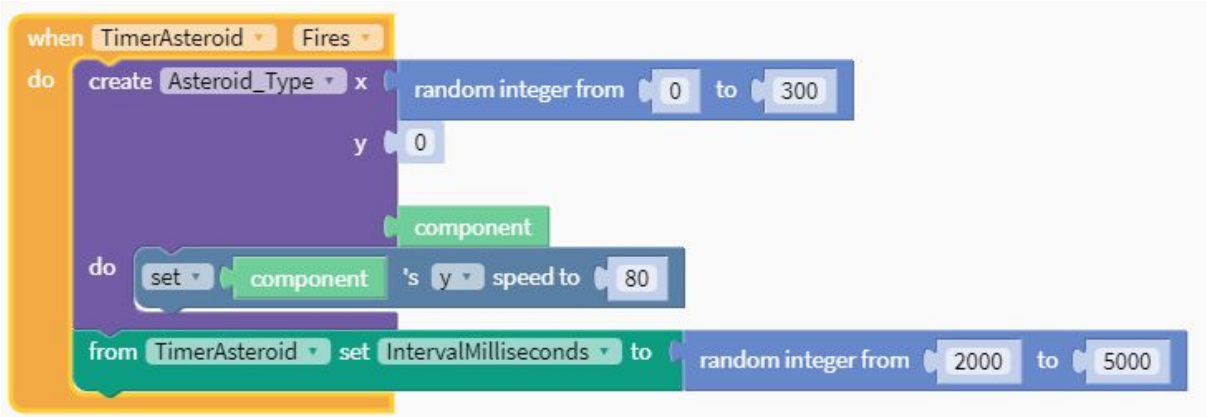
Gegner kannst du genau so wie sammelbare Objekte programmieren. In unserem Spiel sind die Gegner Asteoriden. Sie werden nach einer zufälligen Zeit an einer zufälligen Position ganz oben erzeugt und bewegen sich in Richtung des Spielers.

Dafür brauchen wir wieder einen **neuen Sprite_Type** in unserem Canvas, den wir **Asteroid_Type** nennen und ein Asteroiden-Bild aus dem Ressourcen-Ordner geben. Außerdem erzeugen wir eine neue **Timer-Komponente** in unserem Screen, nennen diese **TimerAsteroid**, setzen diese zunächst auf **3 Sekunden** und aktivieren die **Loop**-Funktion.

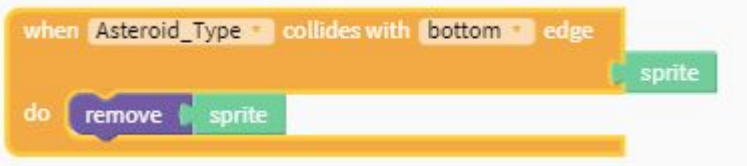


Nun starten wir wieder den TimerAsteroid unter Blocks im Event wenn Canvas1 geladen wird.

Wenn der Timer beendet wird, also nach 3 Sekunden, erzeugen wir einen Asteroiden. Für das zufällige Erscheinen des Nächsten, geben wir danach eine Zufallszahl zwischen 2000 und 5000 Millisekunden (entspricht 2-5 Sekunden) für den Timer an:



Vergiss außerdem nicht die neu erstellten Asteroiden zu löschen sobald sie den unteren Rand des Screens berühren:



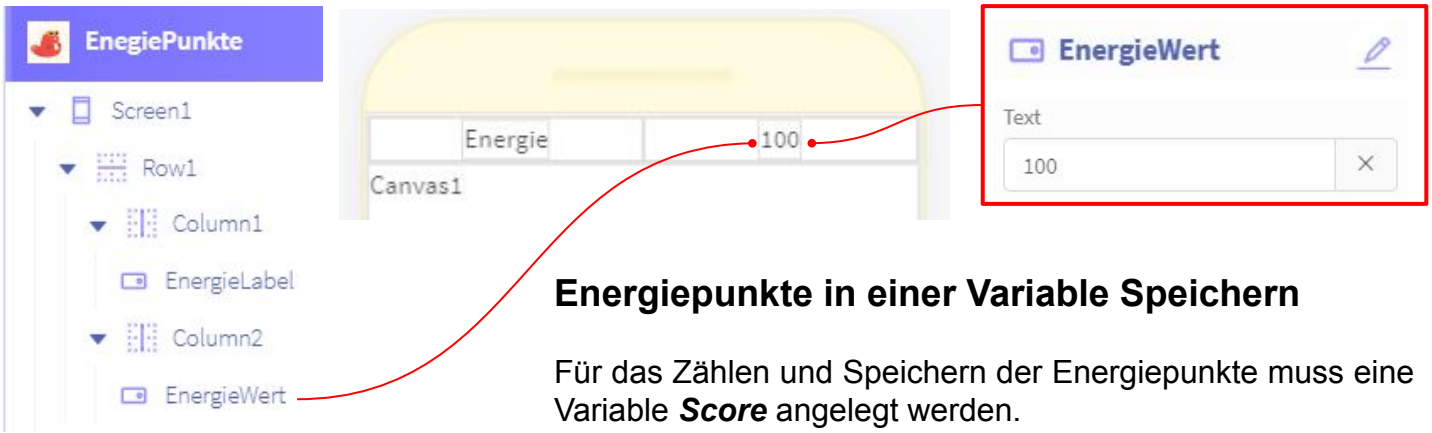
Wenn der Asteroid mit dem Spieler kollidiert ist das Spiel aus und diese Funktion findest du auf der Lernkarte **Game Over**.

Energiepunkte

Du möchtest, dass deine Spielfigur Energiepunkte (z.B. Treibstoff) im Laufe der Spielzeit verliert und dass diese durch Einsammeln von Bonuselementen wieder aufgebaut werden?

Energiepunkte anzeigen

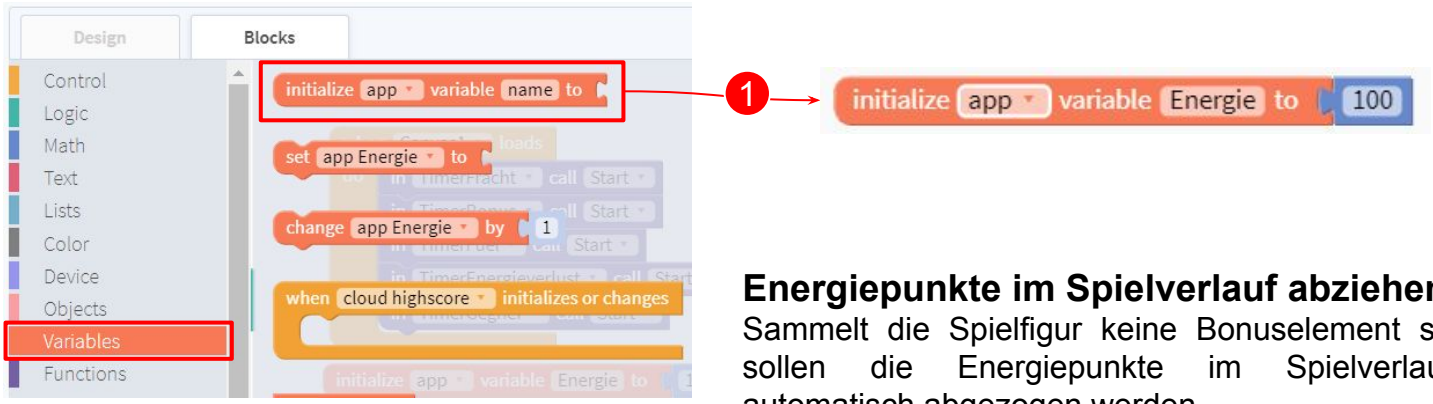
Für die Anzeige der Energiepunkte benötigst du zunächst auf dem Screen einen Bereich mit einer Reihe (*Row*) und 2 Spalten (*Column*). In jeder Zelle werden Beschriftungen (*Labels*) eingefügt und wie folgt benannt. Im Label ***EnergieWert*** sollen die Werte angezeigt werden.



Energiepunkte in einer Variable Speichern

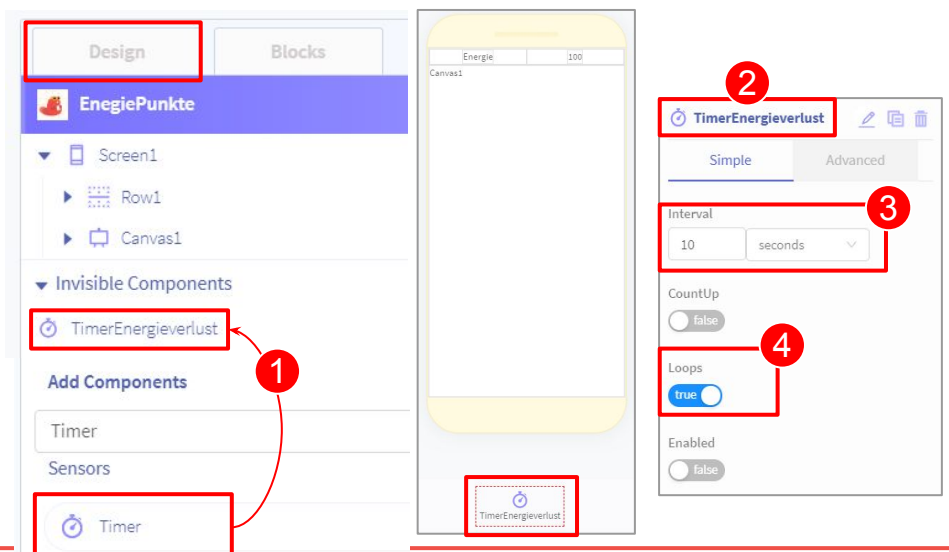
Für das Zählen und Speichern der Energiepunkte muss eine Variable ***Score*** angelegt werden.

1. Die Variable ***Score*** wird mit 100 initialisiert (= belegt) damit startet die Spielfigur mit 100 Energiepunkten.



Energiepunkte im Spielverlauf abziehen

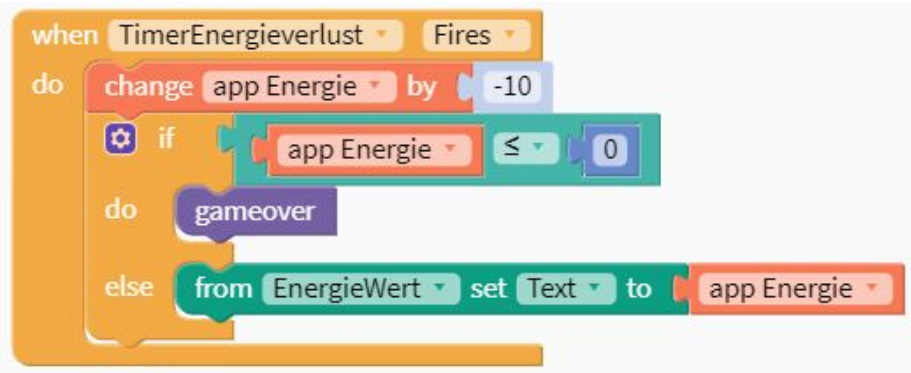
Sammelt die Spielfigur keine Bonuselement so sollen die Energiepunkte im Spielverlauf automatisch abgezogen werden.



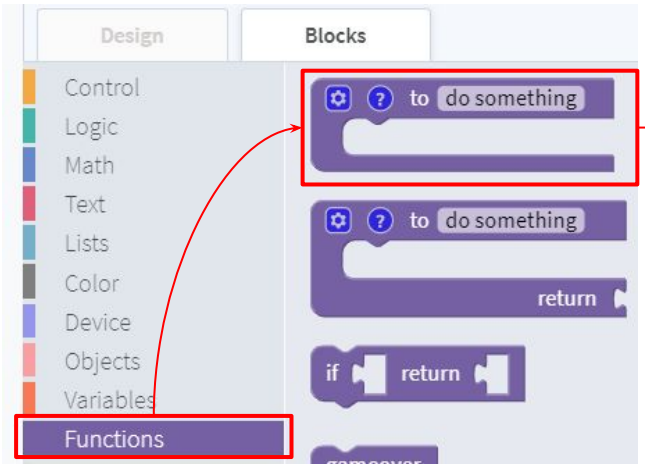
1. Hierzu muss auf der Design Seite die *unsichtbare Komponente* (=Invisible Component) Timer angelegt werden.
2. Dem Timer wird der Name ***TimerEnergieVerlust*** zugewiesen.
3. Es muss ein Interval (z.B. 10 Sekunden) angegeben werden.
4. Durch die Einstellung Loops=True wird der Timer nach Ablauf des Intervals neu gestartet.

Energiepunkte

Energiepunkte im Spielverlauf abziehen (Fortsetzung)



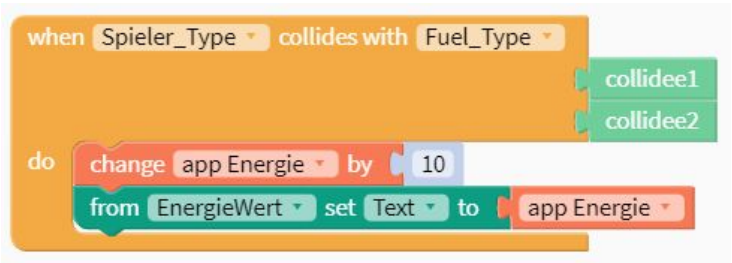
Energiepunkte sollen nun nach Ablauf des Timers ***TimerEnergieVerlust*** abgezogen werden. Wird der Wert der Variable Null oder ist dieser kleiner als Null so endet das Spiel - hierbei wird die Funktion “gameover” aufgerufen.



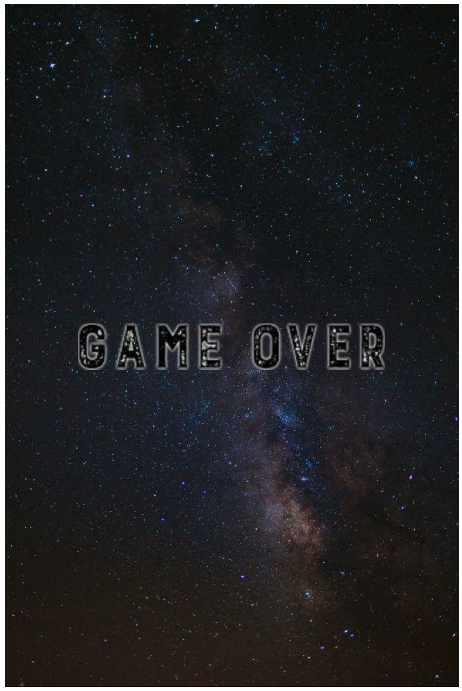
In der Funktion “gameover” können noch weitere Befehle aufgenommen werden um das Spiel zu beenden.

Energiepunkte erhöhen

Wird seitens der Spielfigur ein Bonuselement erfasst (z.B. “Kollision” mit einem Treibstoff Element (Fuel_Type)) so wird die Variable ***Energie*** um 10 erhöht und das Anzeigeelement Label ***EnergieWert*** aktualisiert.



Game Over



Wenn ein gewisses Ereignis eintritt soll es zum Ende des Spiels führen. Es können auch mehrere unterschiedliche Dinge geschehen die das Game beenden, in unserem Spiel gibt es da zwei Möglichkeiten:

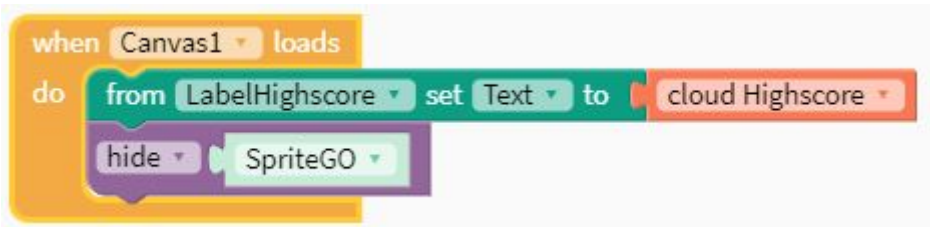
1. Der Tank geht aus
2. Das Raumschiff wird von einem Asteroiden getroffen

Damit wir nicht zweimal den gleichen Code schreiben müssen, erstellen wir uns dafür eine **Function** die zum gewünschten Ergebnis führt.

GAME OVER Screen

Als erstes brauchen wir etwas, das uns visuell zeigt, dass das Spiel vorbei ist sobald das dazugehörige Ereignis eintritt. Dafür erstellen wir einen einen neuen Sprite-Type und nennen ihn Sprite-UI (User Interface). Für den Schriftzug verwenden wir ein Bild aus dem Ressourcen-Ordner. Mit den Koordinaten rechts wird er zentriert. Die Optionen “Passes Through”, “Ignore Gravity” und “Fixed Rotation” sollten aktiviert sein.

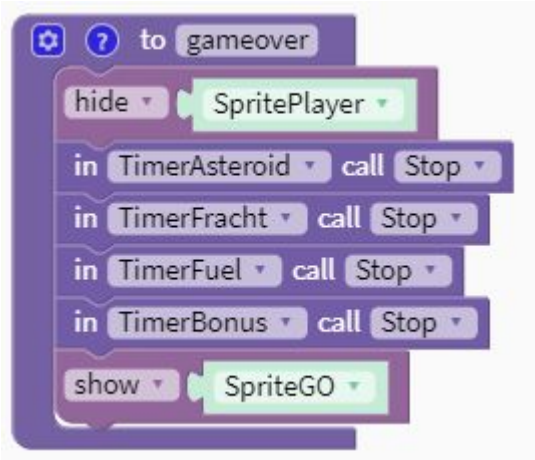
Der Schriftzug soll bei start des Spiels unsichtbar sein, deswegen müssen wir das in dieser Funktion anwenden:



Game Over (Fortsetzung)

Funktion für “Game Over” erstellen

Bei den Blocks unter Functions erstellen wir eine neue und nennen sie **gameover**. Dort kommen nun alle Blöcke hinein, die ausgeführt werden sollen, wenn das Spiel aus ist:
Wir stoppen alle Timer, machen das Raumschiff unsichtbar und wir machen den Schriftzug für Game Over sichtbar:



Die Funktion anwenden

Deine erzeugte Funktion findest du in den Blöcken unter **Functions.** Du brauchst diese nur noch in eine geeignete Stelle ziehen!

Wir benötigen die Funktion in diesen beiden Blöcken:

1. Wenn die Energie 0 erreicht hat:

2. Wenn der Spieler mit einem Asteroiden in Kontakt kommt: