Übungsbuch

für Anfängerinnen und Anfänger

CoderDojo LINZ

CoderDojo LINZ

Übungsbuch

für Anfängerinnen und Anfänger

2. Auflage

Herausgeber: Coding Club Linz e.V., https://linz.coderdojo.net
Umschlagfoto: Rainer Stropek

August 2021

Inhalt

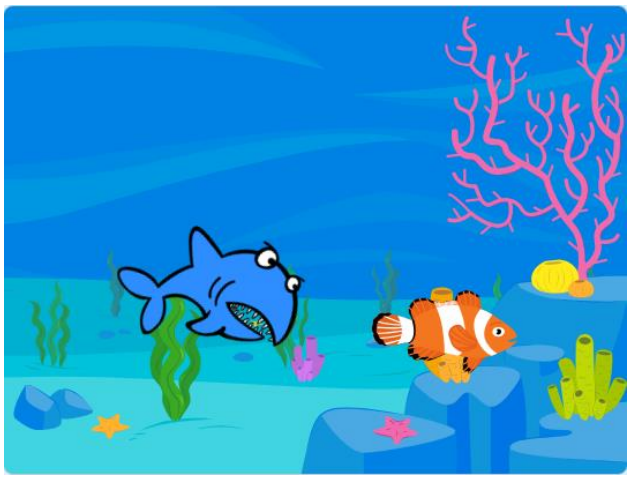
Scratch

Scratch (https://scratch.mit.edu) ist eine Blockprogrammiersprache, die ideal für Einsteigerinnen und Einsteiger ins Coding geeignet ist. Du musst noch nicht gut tippen können, um deine ersten Computerspiele zu bauen. Dieses Buch enthält vier Übungen, bei denen du von Spiel zu Spiel neue Dinge über Scratch lernen wirst. Viel Spaß beim Programmieren!

TypeScript

Im zweiten Teil dieses Buches lernst du, wie man textuell programmiert, indem du kleine Spiele mit einer Programmiersprache namens *TypeScript* baust. Im Gegensatz zu Scratch musst du den Code auf der Tastatur eintippen, genauso wie es die Programmier-Profis auch tun. Am Anfang ist es vielleicht eine Herausforderung, die richtigen Tasten zu finden. Du wirst aber sehen, dass du schon nach kurzer Zeit Code flott tippen kannst.

Fang mich - mein erstes Spiel mit Scratch



In diesem Spiel bist du ein kleiner Fisch, der dem großen Haifisch entkommen muss.

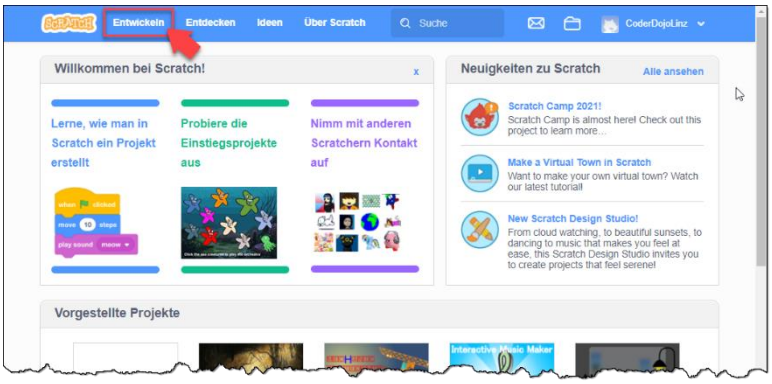
Schaffst du es?

Scratch starten

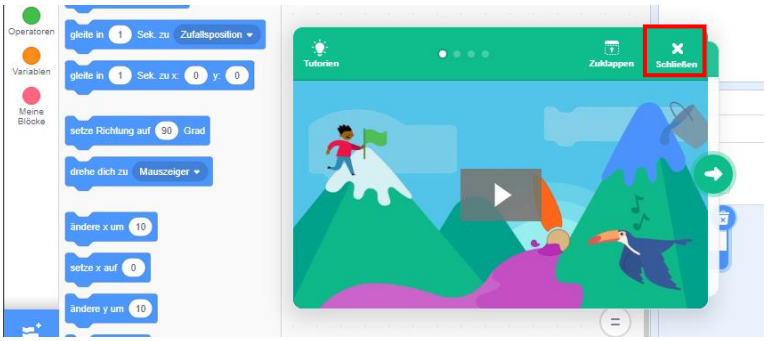
Öffnen einen Browser wie Chrome oder Firefox und öffne die Seite https://scratch.mit.edu.

Falls die Seite englisch angezeigt wird, scrolle ganz nach unten und ändere die Sprache auf Deutsch.

Klicke dann auf den ersten Menüpunkt *Entwickeln*, um mit dem Programmieren zu beginnen.

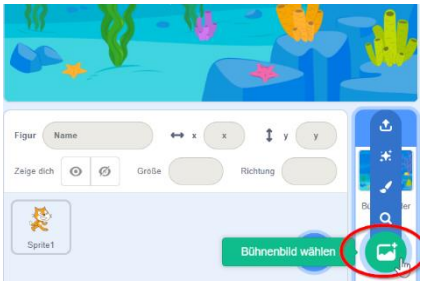


Das grüne Kärtchen mit den Tutorien kannst du schließen.



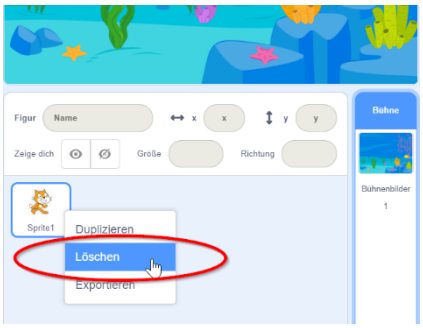
Bühne und Figuren anlegen

Spielfeld



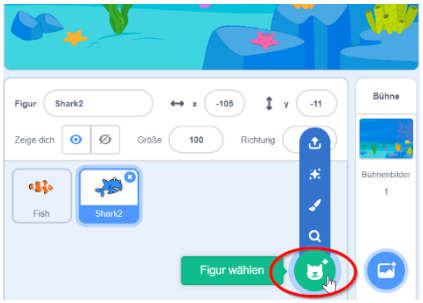
Als erstes legst du fest, wie dein Spielfeld aussehen soll. Wir brauchen zuerst das Aquarium, in dem die Fische schwimmen. Wähle als links unten unter *Bühnenbild aus der Bibliothek wählen* ein Bühnenbild aus - zum Beispiel ein Aquarium.

Scratchy löschen



Als nächstes lösche die Figur Scratchy mit dem Namen *Sprite1* oder *Figur1*, indem du mit der rechten Maustaste daraufklickst. Im angezeigten Menü kannst du Scratchy löschen.

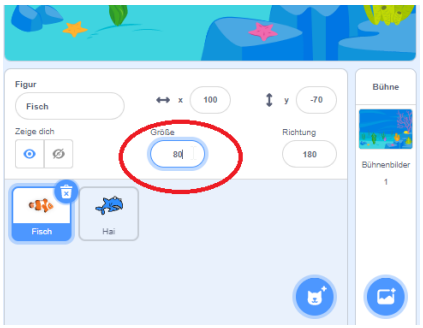
Figuren anlegen



Jetzt brauchen wir einen Haifisch und einen Fisch, mit dem wir dem Haifisch entkommen wollen. Klicke dazu auf *Figur aus der Bibliothek wählen* und füge einen Fisch und einen Haifisch hinzu.

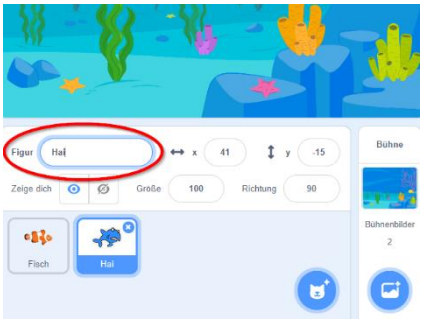
Natürlich können es auch andere Figuren sein, zum Beispiel ein Käfer, der einem Vogel davonläuft.

Fisch verkleinern



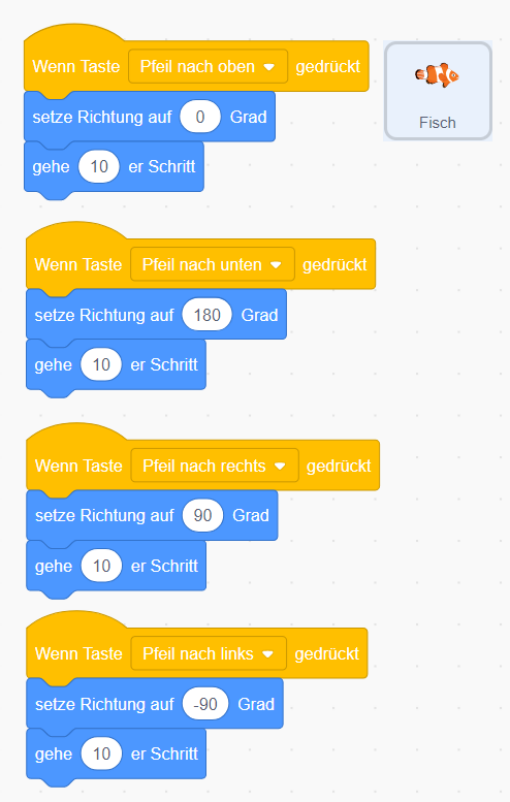
Damit der kleine Fisch auch kleiner ist als der große Haifisch, müssen wir den Fisch verkleinern. Wähle dazu die Figur aus und ändere die Größe, so dass die Figur gut in dein Bühnenbild passt.

Namen vergeben



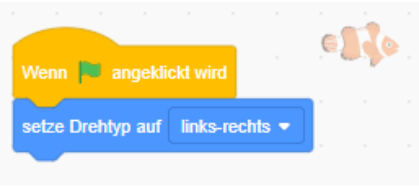
Damit du später die Figuren leichter verwenden kannst, gib ihnen Namen wie *Haifisch* und *Fisch*. Du kannst die Eigenschaften von Figuren ändern, indem du auf das blaue *i* links über der Figur klickst.

Fisch bewegen

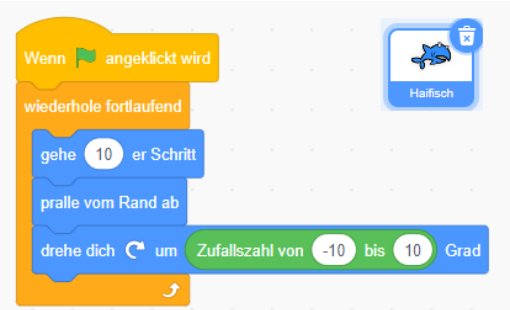


Damit du den Fisch bewegen kannst, musst er nach links und rechts sowie oben und unten bewegt werden können.

- Wähle zuerst den Fisch aus, damit er blau umrandet ist.
- Im Tab *Skripte* kannst du deinen Fisch nun bewegen. Verwende das Ereignis *Wenn Taste … gedrückt* unter *Ereignisse*.
- Verknüpfe es jeweils mit einer Drehung *setze Richtung auf …* unter *Bewegung*, damit der Fisch in die richtige Richtung schaut.
- Außerdem brauchen wir *gehe …er Schritt*, um den Fisch zu bewegen.
- Achte auf die Einstellungen der Kommandos.
  Für *Pfeil nach oben* gedrückt: Richtung 0 Grad, gehe 10er-Schritte.
  Für *Pfeil nach unten* gedrückt: Richtung 180 Grad, gehe 10er-Schritte.
  Für *Pfeil nach rechts* gedrückt: Richtung 90 Grad, gehe 10er-Schritte.
  Für *Pfeil nach links* gedrückt: Richtung -90 Grad, gehe 10er-Schritte.
- Experimentiere mit der Schrittanzahl. Je größer die Schrittanzahl, desto schneller ist dein Fisch.
- Damit der Fisch nicht auf dem Kopf steht, wenn es sich nach links bewegt, kannst du den Drehmodus ändern



Haifisch bewegen



Jetzt soll der Haifisch im Aquarium herumschwimmen.

- Wähle dazu den Haifisch aus, damit er blau umrandet ist.
- Im Tab *Skripte* kannst du den Haifisch nun bewegen.
- Unter *Ereignisse* wähle *Wenn … angeklickt*.
- Anschließend wähle *wiederhole fortlaufend* bei *Steuerung* aus.
- Bewege den Haifisch mit *gehe 10er-Schritt, pralle vom Rand ab* und *drehe dich um … Grad*
- Um etwas mehr Zufall reinzubringen, nimm im Menü *Operatoren* den Block *Zufallszahl von -10 bis 10* und ziehe ihn an die Stelle der *15 Grad*.

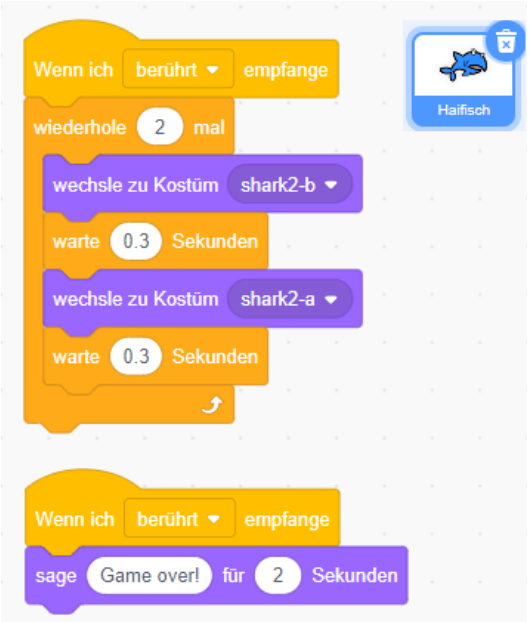
Fisch fangen

Haifisch berührt Fisch

Wenn der Haifisch den Fisch berührt, soll der Fisch ausgeblendet und wieder ins linke obere Eck gesetzt werden.

- Wähle den Fisch aus, damit er blau umrandet ist.
- Im Tab *Skripte* kannst du den Fisch verschwinden lassen, sobald er den Haifisch berührt.
- Unter *Ereignisse* wähle *Wenn … angeklickt*.
- Setze den Fisch an Position -230 und 170 mittels *gehe zu x: -230, y: 170*, um den Fisch ins linke obere Eck zu setzen, und *zeige dich*.
- Falls jetzt der Hai berührt wird (*Steuerung falls … dann*), dann *sende "berührt" an alle*, *verstecke dich*, *warte 5 Sekunden*, *zeige dich*, und gehe wieder ins linke obere Eck mit *gehe zu x: -230, y: 170*. Anschließend sage *Willkommen zurück* für 2 Sekunden.

Hai schnappt nach Fisch



Wenn der Haifisch den Fisch berührt, soll er zweimal schnappen und das Spiel "Game Over" sein.

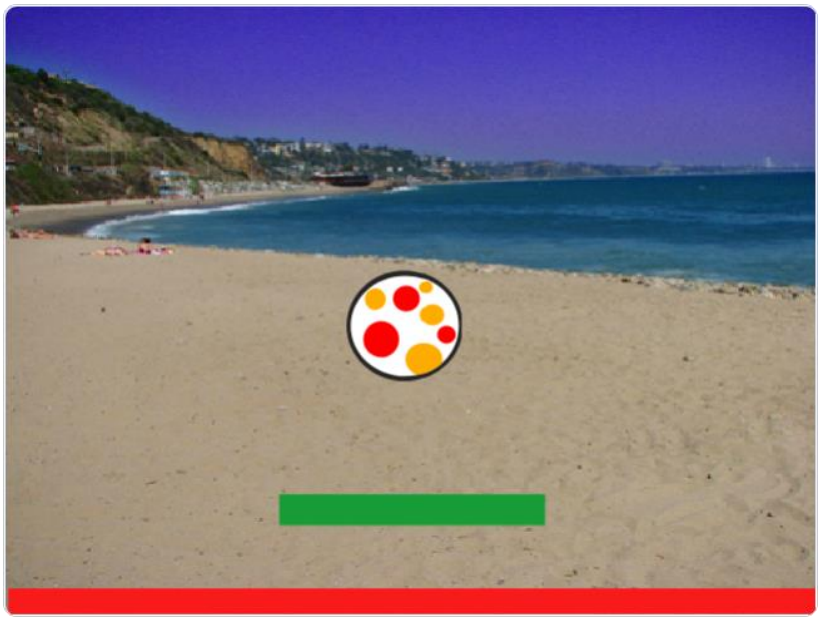
- Wähle dazu den Haifisch aus, damit er blau umrandet ist.
- Im Tab *Skripte* kannst du den Haifisch "Game Over" sagen lassen.
- Unter *Ereignisse* wähle, *Wenn ich … empfange*, der Hai reagiert somit auf die vom Fisch ausgelöste Nachricht. Anschließend wähle *wiederhole 2-mal* bei *Steuerung* aus.
- Um den Haifisch schnappen zu lassen, gibt es unter *Aussehen* verschiedene Varianten des Hais. Füge folgende Blöcke in den Wiederhol-Block: *wechsle zu Kostüm b*, *warte 0.3 Sek.*, *wechsle zu Kostüm a*, *warte 0.3 Sek.*
- Und um den Haifisch "Game over" sagen zu lassen, füge einen neuen *Wenn ich … empfange* Block hinzu und *sage "Game Over!" für 4.5 Sekunden*.

Du kannst das fertige Projekt unter https://meet.coderdojo.net/scratch-fang-mich ausprobieren.

Weitere Ideen

- Mach das Spiel schwieriger, indem du einen zweiten, langsameren Haifisch dazu gibst.
- Baue eine Uhr ein, um zu sehen, wie lange du dem Haifisch entkommen kannst.
- Steuere den Fisch mit der Maus anstatt der mit der Tastatur.

Paddle Game

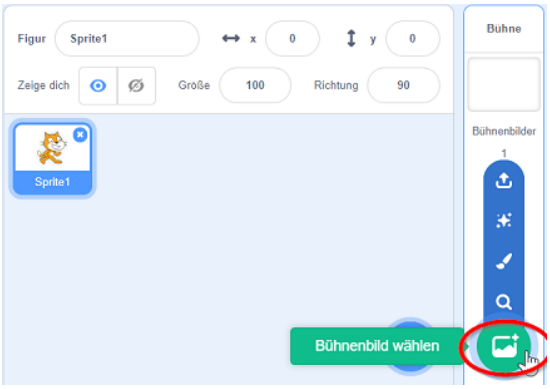


In dieser Übung baust du ein kleines Spiel, indem du versuchst, einen Ball mit einem Schläger in der Luft zu halten.

Bühne und Figuren anlegen

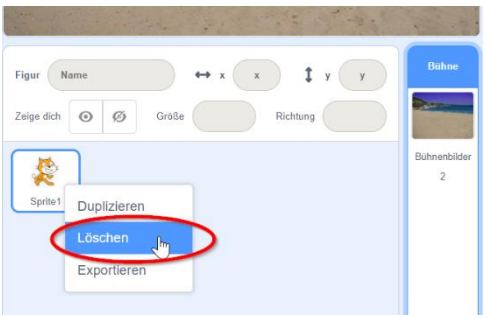
Spielfeld

Als erstes legst du fest, wie dein Spielfeld aussehen soll. Wir brauchen die Bühne, hier der Strand, einen Ball, einen Schläger und einen Bereich, der markiert, wo der Ball im Out ist. Wenn du ein neues Projekt startest, siehst du eine weiße Bühne mit Scratchy, der Katze. Wähle als erstes rechts unten ein Bühnenbild aus.



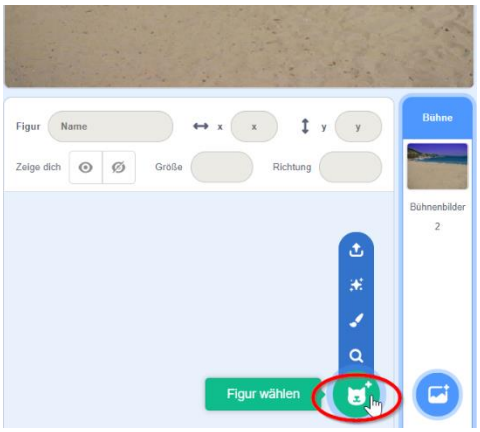
Scratchy löschen

Als nächstes lösche die Figur Scratchy mit dem Namen Sprite 1 indem du mit der rechten Maustaste daraufklickst. Im angezeigten Menü kannst du Scratchy löschen.



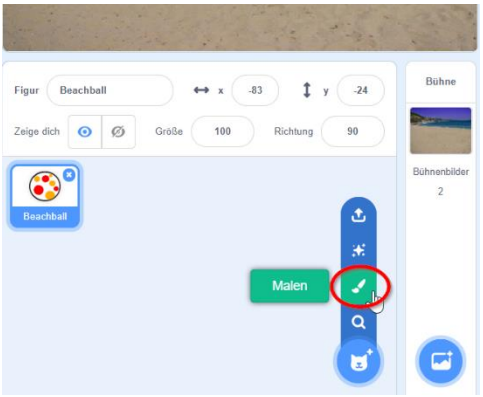
Ball

Jetzt brauchen wir einen Ball und einen Schläger. Links neben dem Bühnenbild findest du einen Menüpunkt, mit dem du neue Figuren hinzufügen kannst. Bei Thema Sport findest du verschiedene Bälle. Wähle einen davon aus.

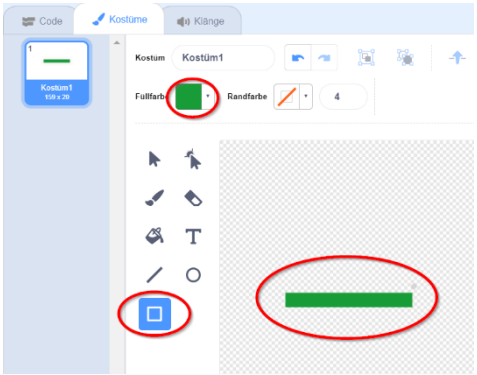


Schläger

Als nächstes ist der Schläger dran. Diesen malen wir selbst. Dafür führst du deine Maus zuerst auf das Feld Figur wählen, darüber siehst du nun neue Felder. Klicke auf das Feld Malen und du bekommst links eine Zeichenfläche.



Male ein einfaches Rechteck, dass den Ball daran hindern wird, am Boden aufzuschlagen. Du brauchst dazu das Werkzeug, um Rechtecke zu malen. Wenn du das Rechteck gemalt hast, wähle noch eine schöne Füllfarbe aus. Danach kannst du es im rechten Bereich auf der Bühne verschieben. Verschiebe es in den unteren Bereich der Bühne, aber nicht ganz nach unten. Hier brauchen wir noch ein wenig Platz für den Bereich, indem der Ball im Out ist.



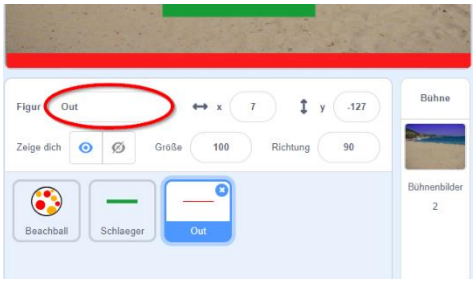
Achte darauf, dass der „Schläger“ genau in der Mitte der Figur ist. Du kannst dich beim Ausrichten am angezeigten Fadenkreuz orientieren.

Out-Bereich

Jetzt brauchst du noch den Out Bereich. Dafür malst du eine weitere Figur - ein Rechteck in einer anderen Farbe. Das Rechteck muss so breit wie der ganze Zeichenbereich sein. Das Rechteck schiebst du dann auf der Bühne ganz nach unten.

Figurennamen

Damit du später die Figuren leichter verwenden kannst, gib ihnen Namen wie Schläger und Out anstelle von Sprite 1 und Sprite 2. Du kannst den Namen unter der Bühne neben dem Wort Figur im Namensfeld ändern.



Schläger nach links und rechts bewegen

Schläger bewegen

Damit du den Ball in der Luft halten kannst, musst du den Schläger nach links und rechts bewegen können. Verwende dazu das Ereignis *Wenn Taste … gedrückt*. Du kannst einmal *Pfeil nach links* und einmal *Pfeil nach rechts* auswählen.

Wird der Pfeil nach links gedrückt, setze die Richtung auf - 90 Grad. Das bedeutet der Schläger bewegt sich nach links. Dann bewege ihn 15 Schritte. Für den Pfeil nach rechts setze die Richtung stattdessen auf 90 Grad.

Damit der Schläger nicht am linken oder rechten Rand verschwindet, kannst du prüfen, ob die x-Position des Schlägers noch nicht zu klein oder groß ist, bevor du den Schläger bewegst.

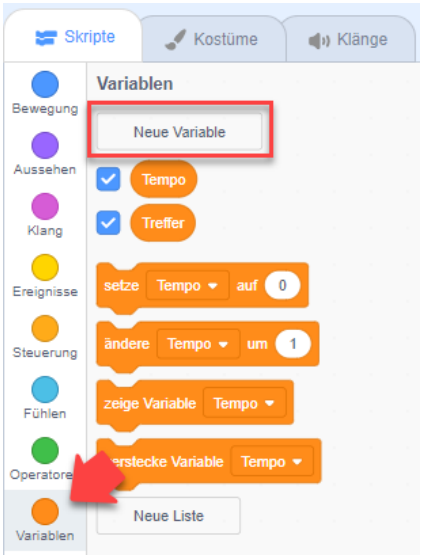
Wenn du jetzt die Pfeiltasten nach links oder rechts drückt, bewegt sich der Schläger. Beim Start des Spiels soll der Schläger in der Mitte sein. Verwende dafür den Block *Wenn Fahne angeklickt* und setze die x-Position des Schlägers auf 0.

Ball herumhüpfen lassen

Variablen

Für unser Spiel brauchen wir zwei *Variablen*: *Tempo* und *Treffer*. *Tempo* legt fest, wie schnell der Ball ist. *Treffer* zählt, wie oft wir den Ball schon nach oben geschossen haben.

Wähle auf der linken Bildschirmseite die Kategorie *Variablen* und klicke auf *Neue Variable*. Tippe den Variablennamen ein (erst *Tempo*, dann *Treffer*) und lege fest, dass die Variable *für alle Figuren* gelten soll.

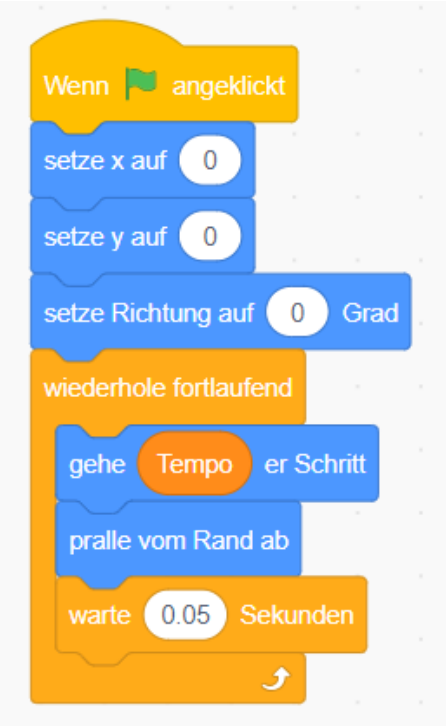


Ball steuern

Als nächstes kannst du den Ball bewegen. Wenn das Spiel gestartet wird, soll der Ball in die Mitte des Spielfelds gesetzt werden. Dazu setzt du die x- und y-Position auf 0. Setze außerdem die Richtung auf 0 - das heißt der Ball bewegt sich nach oben.

Dann kannst du die Bewegung starten. Wiederhole dazu drei Blöcke: gehe 10er-Schritt, pralle vom Rand ab - damit wechselt der Ball die Richtung, wenn er den Rand erreicht und warte 0.05 Sek. - damit bestimmst du die Geschwindigkeit des Balls.

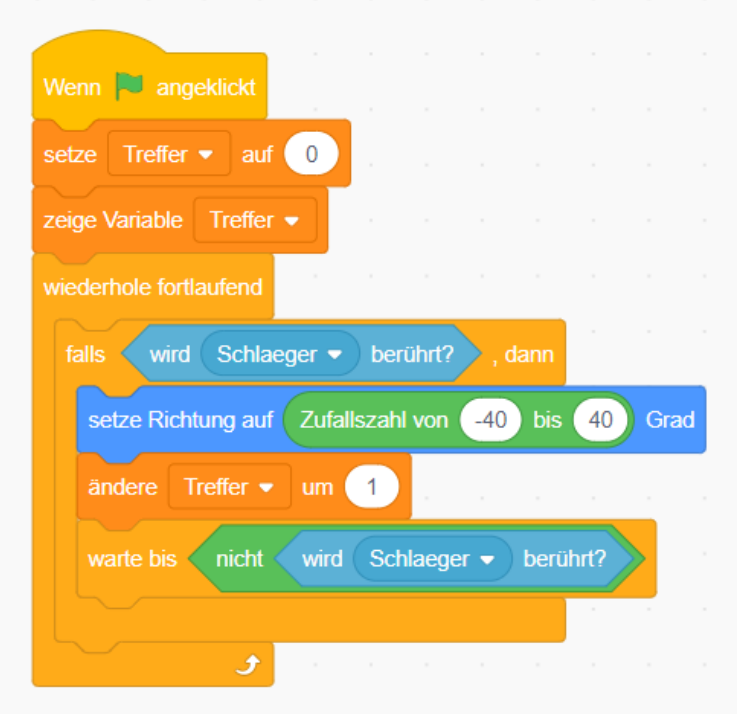
Wenn du diesen Block ausführst, pendelt der Ball zwischen dem oberen und unteren Rand der Bühne. Den Schläger ignoriert er aber noch.



Um die Richtung des Balls zu ändern, wenn er den Schläger berührt, brauchst du einen weiteren Block, der beim Start des Spiels ausgeführt wird.

Hier wird fortlaufend geprüft, ob der Ball gerade den Schläger berührt. Wenn ja, wird die Richtung auf einen zufälligen Wert zwischen -40 und 40 geändert. Die Richtung 0 Grad bedeutet gerade nach oben. Ein Wert zwischen -40 und 40 sagt aus, dass der Ball gerade nach oben oder aber auch etwas links oder rechts davonfliegt.

Dann warte, bis der Ball den Schläger nicht mehr berührt, bevor der Block weiter ausgeführt wird.



Als letztes musst du noch erkennen, wann der Ball den Out-Bereich berührt. Dann ist das Spiel vorbei.

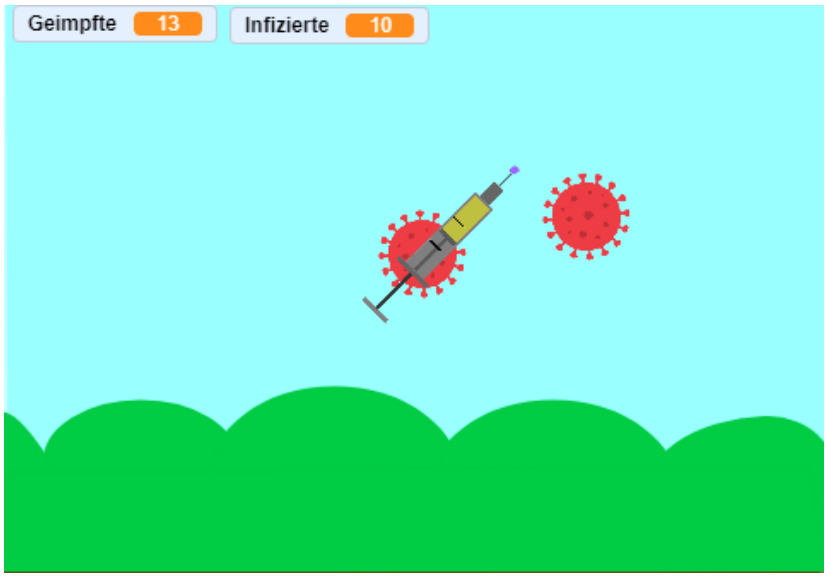
Fertiges Spiel

Gratuliere! Dein Spiel ist fertig. Probiere es gleich aus! Du kannst das fertige Projekt unter http://meet.coderdojo.net/scratch-paddle-game ausprobieren.

Weitere Ideen

- Steuere den Schläger mit der Maus statt mit der Tastatur.
- Bringe nach einiger Zeit einen weiteren Ball ins Spiel.

Virus-Buster

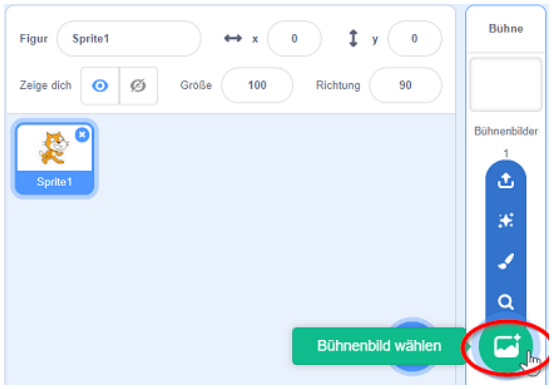


Durch dein schnelles Handeln kannst du in diesem Spiel bösartige Viren unschädlich machen.

Bühne und Figuren anlegen

Bühne

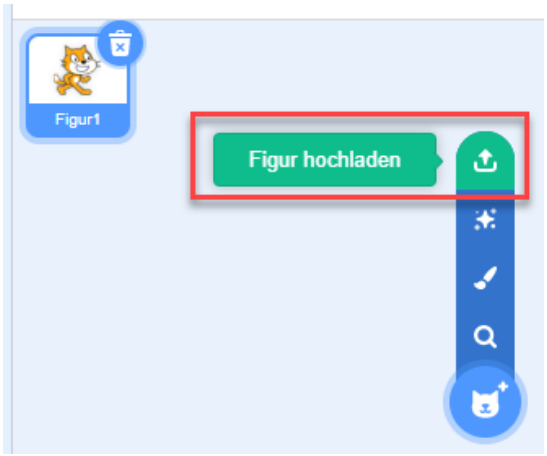
Als erstes legst du fest, wie dein Spielfeld aussehen soll. Wir brauchen einen Hintergrund, die Viren und die Impfspritze. Wenn du ein neues Projekt startest, siehst du eine weiße Bühne mit Scratchy, der Katze. Wähle als erstes rechts unten ein beliebiges Bühnenbild aus.



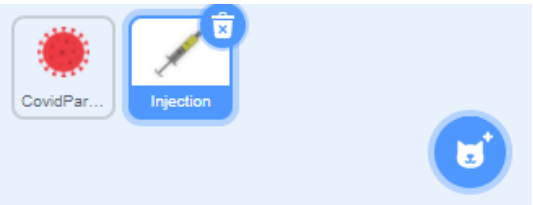
Figuren

Wir haben die Figuren für das Virus und die Impfspritze vor dich vorbereitet.

Lade die Virus-Figur von unserer Webseite herunter (https://meet.coderdojo.net/virus-buster-virus). Merke dir, wo du die Figur speicherst. Mit *Figur hochladen* kannst du anschließend die Figur in dein Scratch-Projekt übernehmen. Wiederhole den Vorgang mit der Figur für die Spritze (http://meet.coderdojo.net/virus-buster-injection). Abschließend kannst du die *Scratchy*-Figur (Katze) löschen.



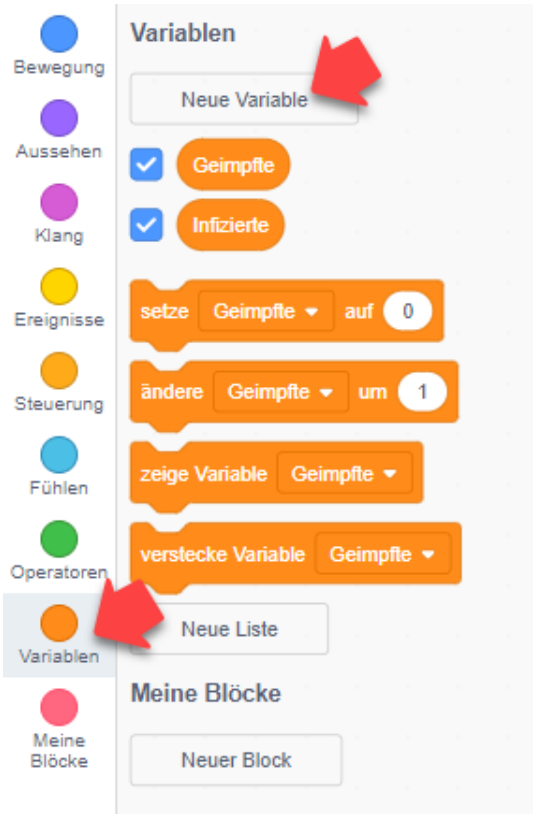
Jetzt solltest du zwei Figuren in deinem Projekt haben: Den Virus und die Spritze.



Variablen

In unserem Spiel möchten wir 2 Variable verwenden. Achte darauf, dass die Variablen mit einem Häkchen bei *Für alle Figuren* angelegt werden.

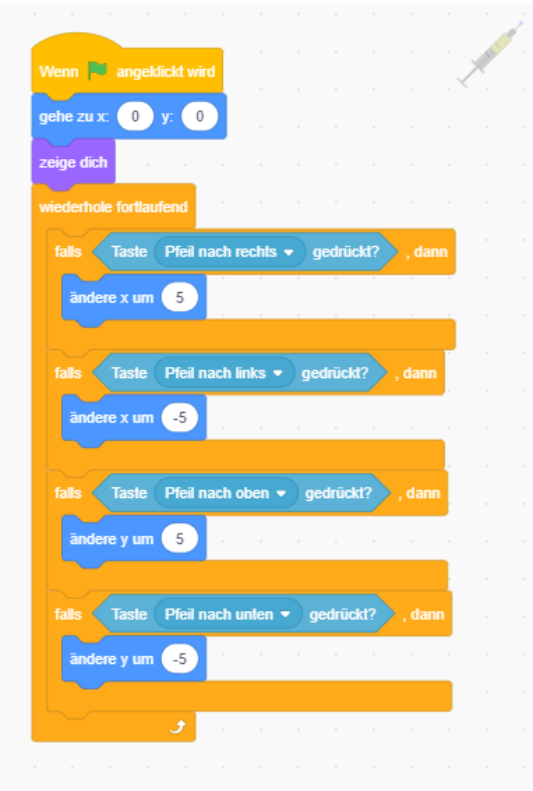
- *geimpfte*
- *infizierte*



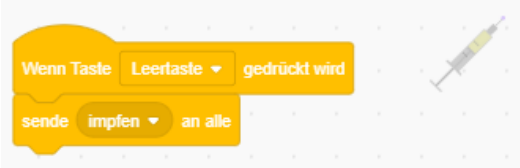
Spritze

Spritze steuern

Um die Spritze zu steuern, werden wir die Pfeil-Tasten verwenden. Da es bei diesem Spiel um schnelles Reagieren geht, können wir leider die Ereignisblöcke (z.B. *Wenn Pfeil nach unten gedrückt wird*) nicht verwenden. Wir werden stattdessen die Abfrage, ob eine Taste gerade nach unten gehalten wird, verwenden.



Impfung verabreichen

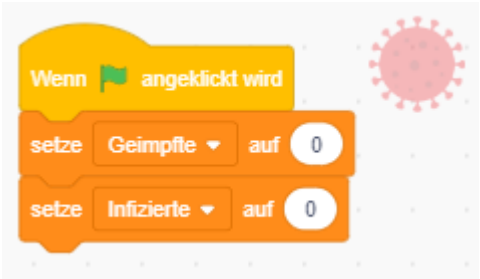


Für das Verabreichen der Spritze können wir das Ereignis *Wenn Taste Leertaste gedrückt wird* nehmen. Wir senden damit eine neue Nachricht (z.B. *impfen*) an alle.

Wir müssen noch einen kleinen Farbklecks an die Spitze der Spritze machen. Bitte verwende dazu eine Farbe, die sonst im Spiel nicht vorkommt (nicht im Hintergrund und nicht auf anderen Figuren, die das Virus berühren könnten). Dieser wird als unser *Trefferpunkt* dienen.

Virus

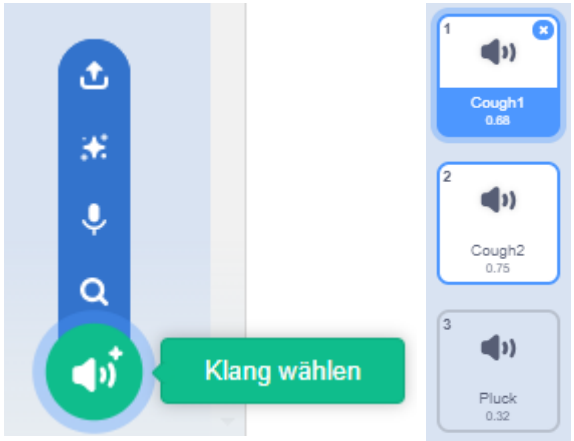
Spielstand



Unsere Viren sollen nach dem Freisetzen von rechts nach links über den Bildschirm fliegen. Wir müssen sie unschädlich machen, bevor sie den rechten Rand erreichen und weitere Menschen infizieren.

Beim Spielstart sollten wir die Punktestände auf 0 stellen.

Töne



Als nächstes holen wir uns noch Töne für das Freisetzen des Virus und deren Bekämpfung. Du kannst diese Klänge ganz einfach aus der Scratch-Bibliothek importieren.

Du kannst natürlich auch eigene Klänge verwenden, wenn dir diese besser gefallen.

Viren erzeugen

Da wir mehrere Viren auf dem Bildschirm haben möchten, werden wir zuerst die Figur unsichtbar machen. Nach einer zufälligen Zeit erzeugen wir dann einen Klon.

Virus bewegen

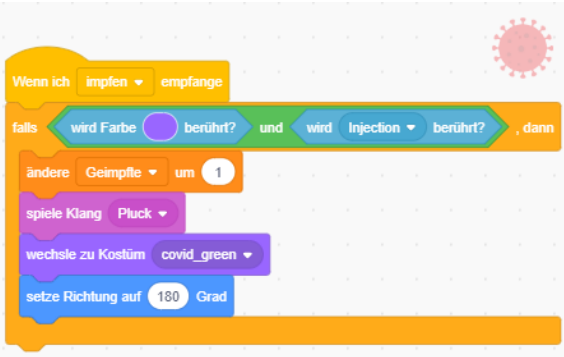
Den Klon bringen wir anschließend an eine zufällige Position an der rechten Seite und machen ihn sichtbar. Von dort aus tritt das Virus seine Reise über den Bildschirm an.

Das verwendete Kostüm soll uns Auskunft darüber geben, ob das Virus noch gefährlich oder schon unschädlich ist. Daher stellen wir es am Anfang auf das erste Kostüm (*covid_red*).

Gelangt das Virus zum Rand können wir so feststellen, ob noch jemand infiziert wurde oder nicht.

Als Abschluss zerstören wir den Klon.

Impfen



Nun fehlt uns nur noch das eigentliche Impfen. Wir haben bei der Spritze die Nachricht *impfen* abgeschickt und wollen nun wissen, ob wir einen Virus erwischt haben oder nicht.

Dabei wird uns unser Trefferpunkt (siehe Spritze) helfen. Bitte achte darauf, dass du genau die Farbe des Trefferpunktes beim *Wird Farbe .. berührt* Block einstellst. Du kannst dazu das *Pipetten*-Werkzeug verwenden.

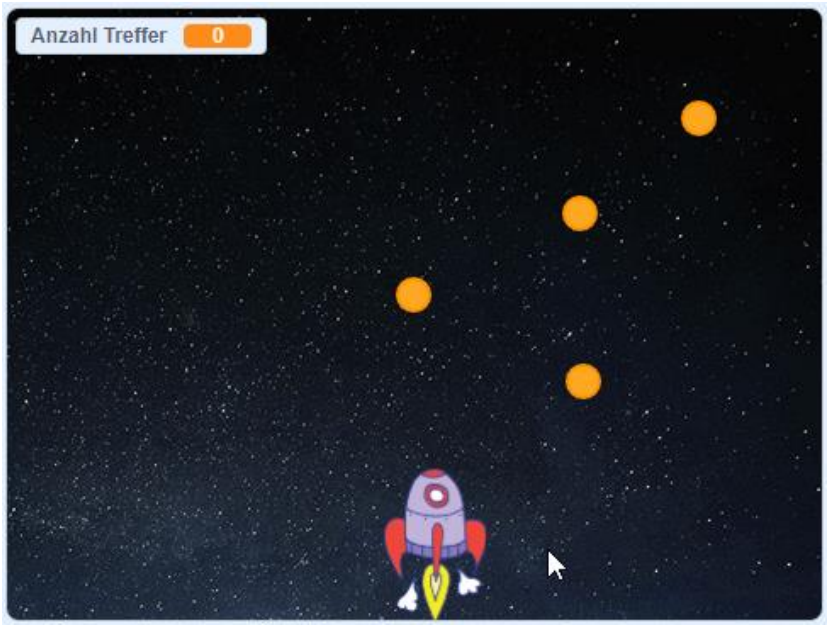
Fertig!

Gratuliere! Dein Spiel ist fertig. Probiere es gleich aus! Du kannst das fertige Projekt unter https://meet.coderdojo.net/virus-buster ausprobieren.

Weitere Ideen

- Bei den Hustentönen muss nicht jedes Mal der gleiche Ton verwendet werden. Du kannst zufällig entweder *Cough1* oder *Cough2* abspielen.
- Kannst du dein Skript so erweitern, dass nur 3 Spritzen verabreicht werden können bevor du z. B. 2 Sekunden warten musst? Wenn du magst, kannst du dafür die Figur *Impfung* verwenden (herunterzuladen unter http://meet.coderdojo.net/virus-buster-vaccine).

Space Shooter

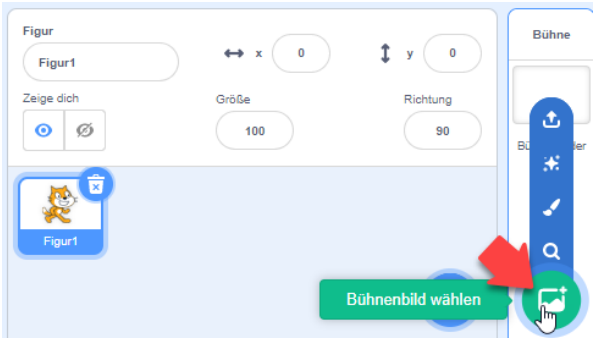


In dieser Übung schießt du mit deinem Raumschiff herabfallende Meteoriten ab bevor sie dein Raumschiff zerstören.

Bühne und Figuren anlegen

Spielfeld

Als erstes legst du fest, wie deine Bühne aussehen soll. Für dieses Spiel brauchst du das Weltall als Hintergrund. Wähle ein passendes Bild aus oder male selbst eines.

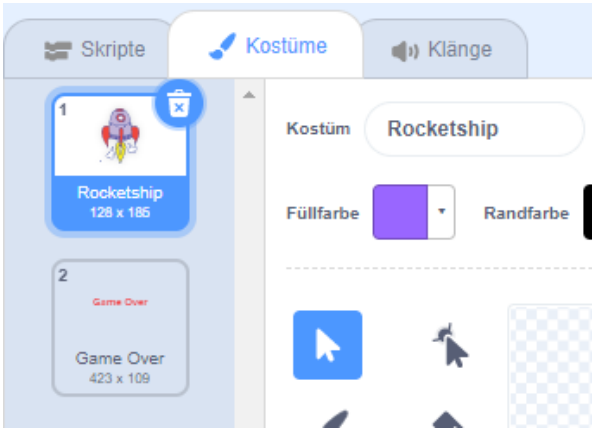


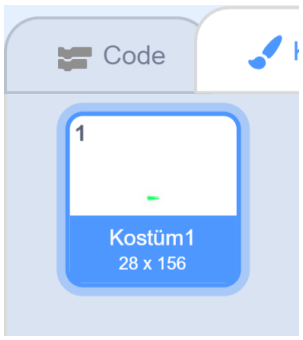
Figuren

Als erste Figur brauchst du das Raumschiff. Du fügst es mit *Figur wählen* ein.

Die Figur muss aus zwei Kostümen bestehen: dem Raumschiff selbst und einem weiteren Kostüm, das angezeigt wird, wenn das Spiel vorbei ist.

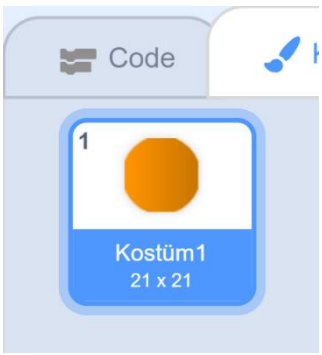
Erstelle dazu ein eigenes Kostüm mit dem Text *Game Over* oder male ein passendes Bild.

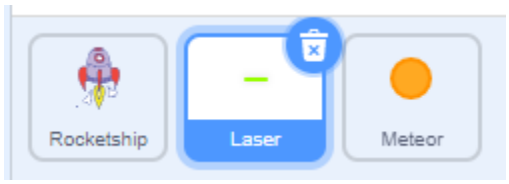




Die nächste Figur ist der Laserstrahl, der von der Rakete abgefeuert werden kann. Diesen kannst du dir selbst malen.

Und als letzte Figur brauchst du noch einen Meteoriten. Diesen kannst du dir selbst malen.

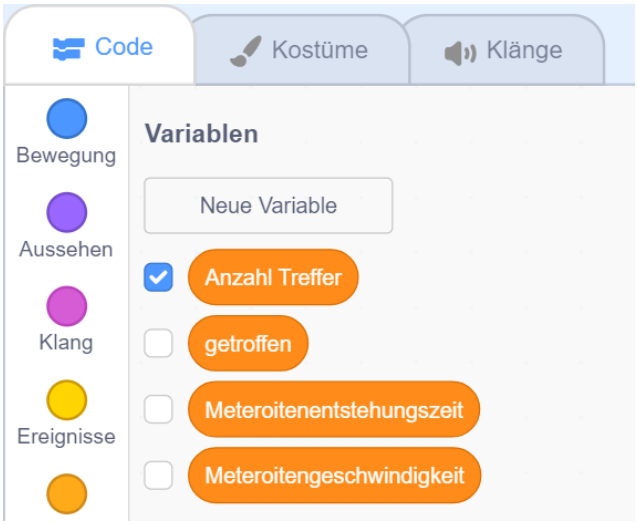




Am Ende solltest du drei Figuren haben: Raumschiff, Laser und Meteor.

Variablen

Wir brauchen für den Space Shooter einige *Variablen*: die Anzahl der Treffer, die Entstehungszeit von Meteoriten, deren Geschwindigkeit und einen Indikator, ob ein Meteorit getroffen wurde.

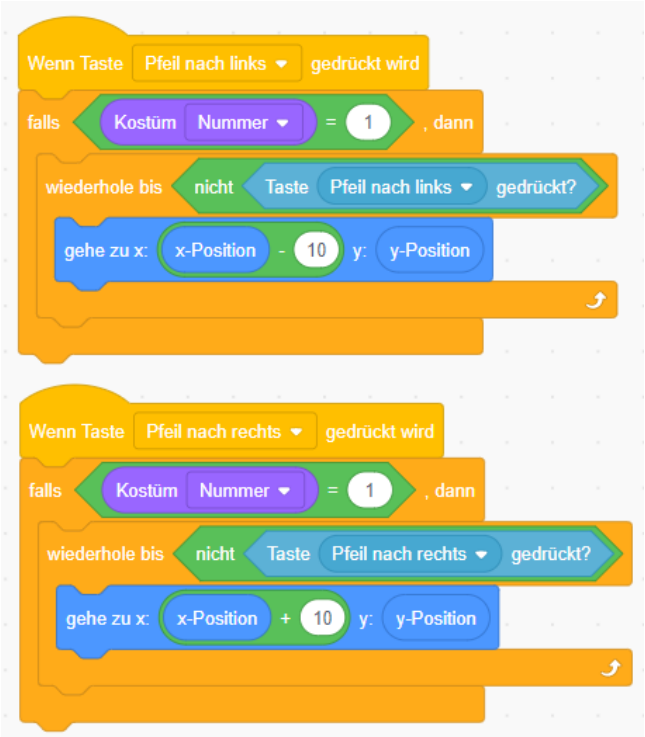


Achtung: **Die Variable getroffen gilt nur für die Figur "Meteorit", alle anderen Variablen gelten *für alle Figuren*.**

Skripte für das Raumschiff

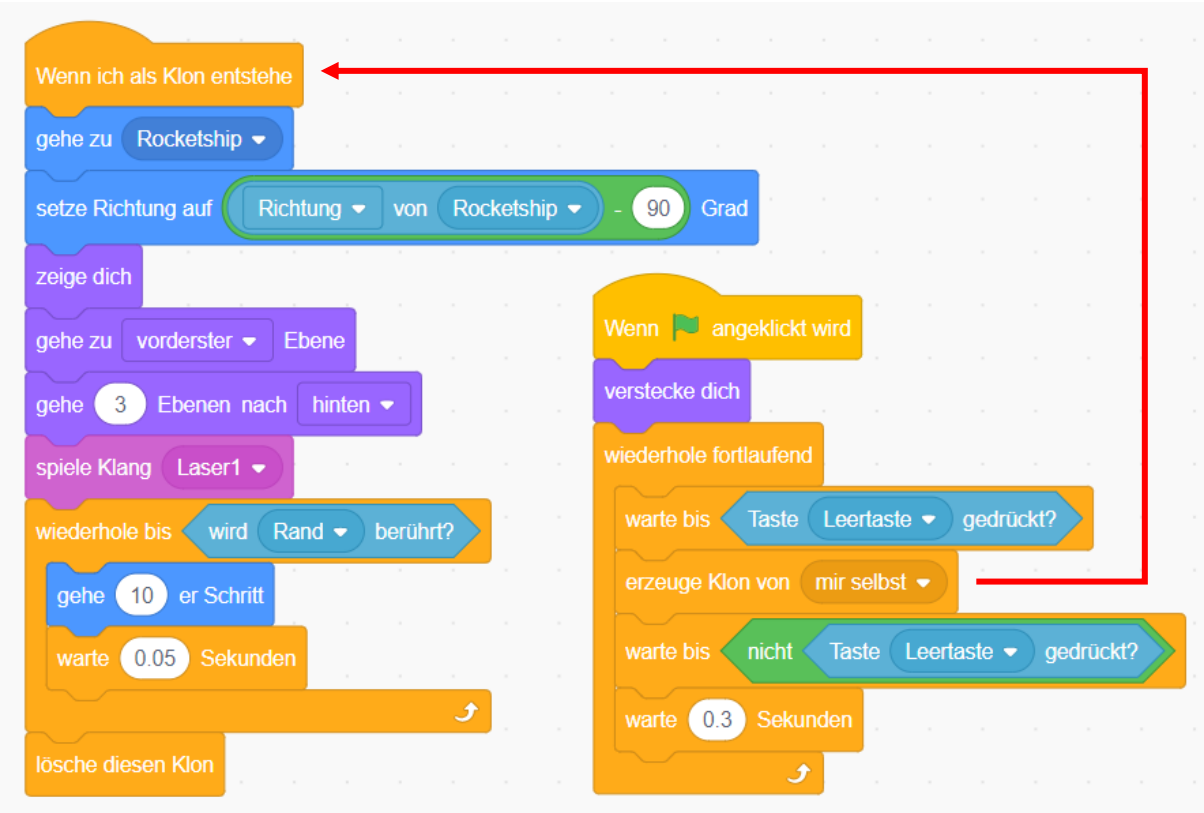
Das Raumschiff hat drei Aufgaben:

- Es muss erkennen, wann es von einem Meteoriten getroffen wurde und dann das Spiel beenden.
- Mit den Pfeiltasten kann es nach links und rechts bewegt werden.
- Mit den Tasten *a* und *d* kann es nach links und rechts gedreht werden.



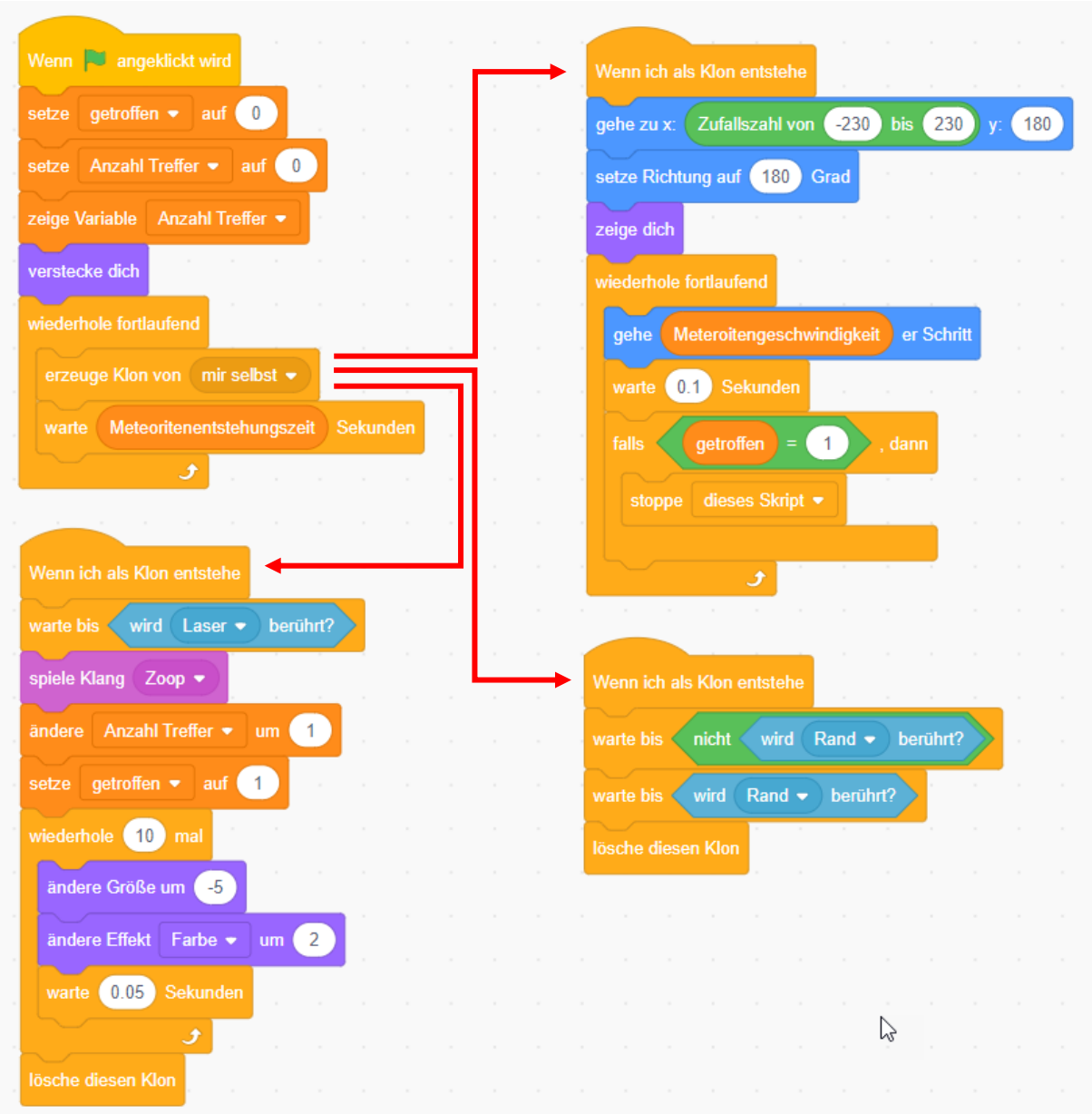
Skripte für den Laser

Jedes Mal, wenn die Leertaste gedrückt wird, muss ein neuer Laserstrahl erzeugt und abgefeuert werden.



Skripte für den Meteoriten

Für die Meteoriten brauchen wir eine Schleife, in der alle paar Sekunden ein neuer Meteorit erzeugt wird. Die Meteoriten müssen langsam nach unten fallen. Sie müssen verschwinden, wenn sie am unteren Bildschirmrand angekommen sind oder von einem Laser getroffen wurden.



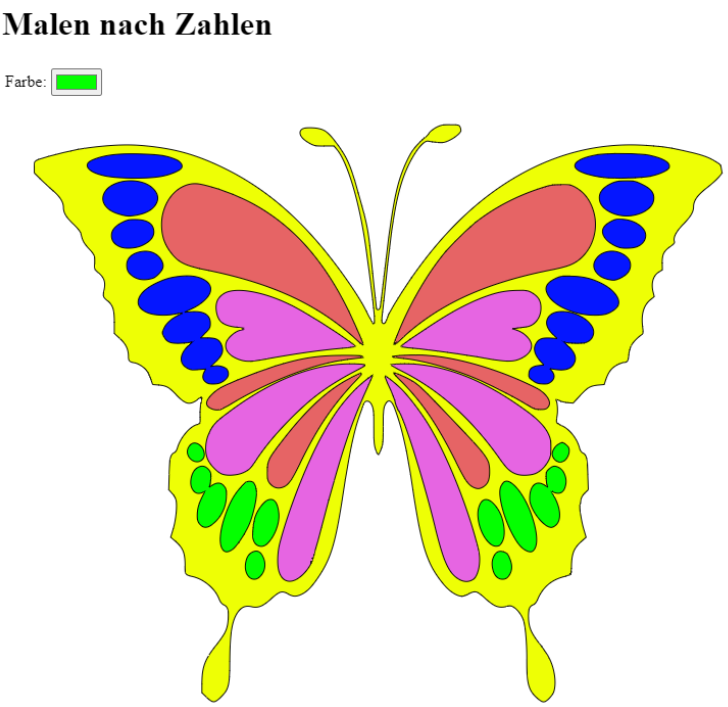
Fertig!

Gratuliere! Dein Spiel ist fertig. Probiere es gleich aus! Du kannst das fertige Projekt unter https://meet.coderdojo.net/scratch-spaceshooter ausprobieren.

Weitere Ideen

- Füge verschiedene Arten von Meteoriten hinzu.
- Ändere den Hintergrund, je länger das Spiel läuft.
- Zeige den Feuerstrahl des Raumschiffs nur an, wenn es sich bewegt.

Malen nach Zahlen



Wir lernen, wie man im Browser programmiert und bauen dabei ein kleines Mal-programm.

Ziel der Übung

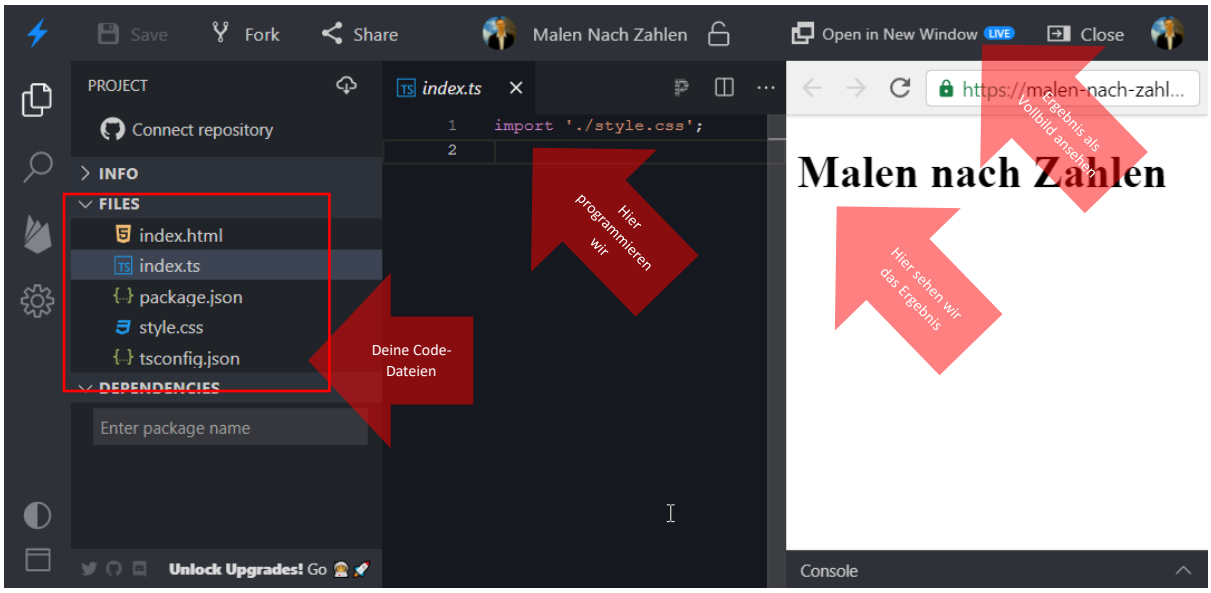
Wenn du noch nie textuell programmiert hast, ist das die richtige Übung zum Einsteigen für dich. Du kannst das Tippen von Code auf der Tastatur üben und lernst Grundlagen der Browser-Programmierung wie HTML, Variablen und Schleifen kennen.

Du solltest für diese Übung schon ein wenig Programmiererfahrung mit einer Blockprogrammiersprache wie *Scratch* oder *Snap!* haben. Falls du noch nie Programmcode eingetippt hast, solltest du auch etwas Geduld mitbringen. Es ist am Anfang nicht immer leicht, die vielen Sonderzeichen im Code auf der Tastatur zu finden. Halte durch, Übung macht den Meister!

Start

Du brauchst für diese Übung keine spezielle Software auf deinem Computer zu installieren. Es ist nur ein aktueller Web-Browser (vorzugsweise Chrome) notwendig.

Um loszulegen, öffne die URL https://stackblitz.com/edit/malen-nach-zahlen-starter. Du wirst dadurch im Web-Browser ein Fenster sehen, das in etwa so aussieht:



- Links siehst du eine Liste der Dateien. Wir programmieren heute in *index.html* und *index.ts*.
- Unseren Programmcode schreiben wir im Textfenster in der Mitte.
- Rechts sehen wir nach jeder Codeänderung gleich das Ergebnis.
- Mit *Open in New Window* kannst du das Ergebnis deines Programms im Vollbildmodus ansehen.

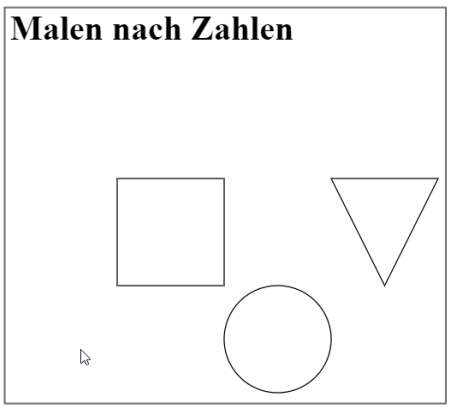
In dieser Übung programmierst du mit HTML und der Programmiersprache *TypeScript*.

Programmieren mit HTML

Worum geht es?

In unserem Malprogramm starten Benutzer mit einem Schwarzweißbild. Wir werden mit einfachen Formen wie Rechtecken und kreisen beginnen, wollen am Ende aber dann einen schönen Schmetterling sehen. Anschließend wählt man eine Farbe und klickt auf eine weiße Fläche im Bild. Dadurch färbt man die Fläche entsprechend ein. Es entsteht ein schönes, buntes Bild.

Erste Formen



Als erstes zeichnen wir einfache Formen auf den Bildschirm. Dafür verwenden wir sogenannte *Vektorgrafiken*. In Englisch heißen sie *Scalable Vector Graphics*, kurz *SVG*.

Öffne *index.html* und füge die in Rot dargestellten Codezeilen hinzu. Das Ergebnis sollte so aussehen wie rechts angezeigt.

```
<h1>Malen nach Zahlen</h1>

<svg id="image" width="1000" height="600">
   <rect x="100" y="100" width="100" height="100" stroke="black" />
   <circle cx="250" cy="250" r="50"  stroke="black" />
   <path d="m300,100 h100 l-50,100 Z" stroke="black" />
</svg>
```

Hier ein paar Tipps zum Code, den du gerade geschrieben hast:

- *Width* heißt auf Deutsch *Breite*
- *Height* heißt auf Deutsch *Höhe*
- *Rect* ist eine Abkürzung für *Rectangle* und heißt auf Deutsch *Rechteck*. Mit *rect* zeichnen wir also das Rechteck.
- *Circle* heißt auf Deutsch *Kreis*. Mit *circle* zeichnen wir also den Kreis.
- *Path* heißt auf Deutsch *Pfad*. Damit können wir beliebige Linienpfade zeichnen. Bei uns malen wir ein Dreieck.
- *Stroke* heißt auf Deutsch *Pinselstrich*. Damit legst du die Farbe fest. Im Moment steht überall *black*, auf Deutsch *Schwarz*. Probiere zum Beispiel einmal, *black* durch *red* (Rot) zu ersetzen. Siehst du den Unterschied?

Auf die Details der Koordinaten (*x*, *y*, *cx*, *cy* etc.) wollen wir hier im Moment noch nicht eingehen. Du kannst aber mit den Zahlen experimentieren. Was passiert, wenn du größere oder kleinere Werte verwendest?

Farbauswahl

Als nächstes möchten wir, dass man sich eine Malfarbe aussuchen kann. Dazu müssen wir wieder *index.html* erweitern.

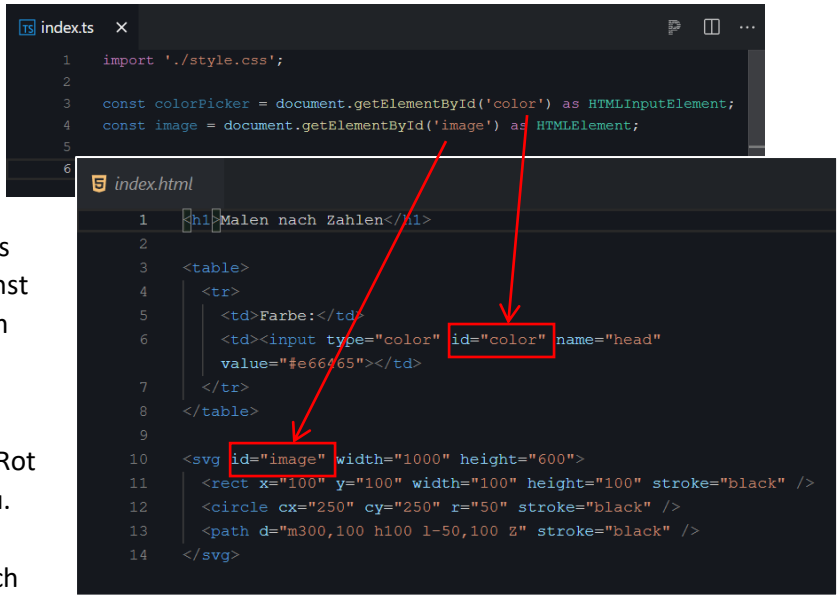
Öffne *index.html* und füge die in Rot dargestellten Codezeilen hinzu. Das Ergebnis sollte so aussehen wie rechts angezeigt.

```
<h1>Malen nach Zahlen</h1>

<table>
  <tr>
    <td>Farbe:</td>
    <td><input type="color" id="color" name="head" value="#e66465"></td>
  </tr>
</table>

<svg id="image" width="1000" height="600">
  <rect x="100" y="100" width="100" height="100" stroke="black" />
  <circle cx="250" cy="250" r="50" stroke="black" />
  <path d="m300,100 h100 l-50,100 Z" stroke="black" />
</svg>
```

Super gemacht! Wir haben Formen und eine Farbauswahl. Jetzt können wir mit TypeScript den Code schreiben, der es uns ermöglicht, die Formen einzufärben.

Programmieren mit TypeScript

Elemente suchen

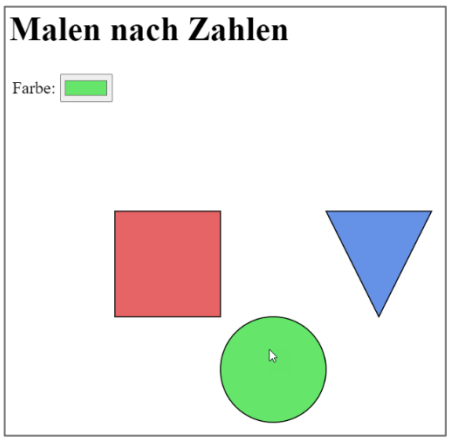


Als erstes müssen wir in TypeScript die HTML-Elemente, die du zuvor in HTML programmiert hast, heraussuchen. Wir machen das über ihre *id*. Im Bild rechts siehst du, wie die *id* in HTML mit dem Code zusammenhängt, den du gleich schreiben wirst.

Öffne *index.ts* und füge die in Rot dargestellten Codezeilen hinzu. An der Ausgabe deines Programms ändert sich dadurch noch nichts.

```
import './style.css';

const colorPicker = document.getElementById('color') as HTMLInputElement;
const image = document.getElementById('image') as HTMLElement;
```

Auf Mausklick reagieren



Jetzt können wir die Formen ausmalen, wenn man auf sie mit der Maus klickt.

Öffne *index.ts* und füge die in Rot dargestellten Codezeilen hinzu. Danach wähle eine Farbe aus und klicke auf eine der Formen. Ihre Füllfarbe sollte sich verändern, wenn du alles richtig gemacht hast.

```
import './style.css';

const colorPicker = document.getElementById('color') as HTMLInputElement;
const image = document.getElementById('image') as HTMLElement;

for(let i = 0; i < image.children.length; i++) {
  const child = image.children[i] as SVGElement;

  child.onclick = () => {
    child.style.fill = colorPicker.value;
  }
}
```

Schmetterling

Jetzt wollen wir die einfachen Formen durch den Schmetterling ersetzen. Der Code der Vektorgrafik mit dem Schmetterling ist natürlich viel länger als der Code für die drei Formen. Damit du nicht so viel tippen musst, haben wir den Schmetterling vorbereitet. Deine Aufgabe ist es, den Code zu kopieren und an der richtigen Stelle in deinem Programm einzufügen.

Code für Schmetterling kopieren

Öffne https://meet.coderdojo.net/malen-nach-zahlen-schmetterling in einem eigenen Browser-Fenster. Markiere mit der Maus den gesamten SVG-Code. Im Bild oben siehst du, was damit gemeint ist. **Achtung:** Der Code ist lang! Du musst weit nach unten scrollen, um alles zu erwischen.

Anschließend klicke mit der rechten Maustaste auf den markierten Code und wähle *Kopieren*.

Schmetterling einfügen

Öffne *index.html* und ersetze den bestehenden SVG-Codeblock durch den Code des Schmetterlings. Du musst dafür den unten durchgestrichen dargestellten Code löschen und stattdessen den kopierten Code für den Schmetterling einfügen.

```
<h1>Malen nach Zahlen</h1>

<table>
  <tr>
    <td>Farbe:</td>
    <td><input type="color" id="color" name="head" value="#e66465"></td>
  </tr>
</table>

<svg id="image" width="1000" height="600">
  <rect x="100" y="100" width="100" height="100" stroke="black" />
  <circle cx="250" cy="250" r="50" stroke="black" />
  <path d="m300,100 h100 l-50,100 Z" stroke="black" />
</svg>

<svg id="image" width="1000" height="600"  viewBox="0 10 204 172"
xmlns:svg="http://www.w3.org/2000/svg" xmlns="http://www.w3.org/2000/svg">
  <path style="stroke:#000000;stroke-width:0.26458332"
        d="m 57.689061,177.30516 ..." id="path3782" />
  ...
</svg>
```

Diesen Code löschen

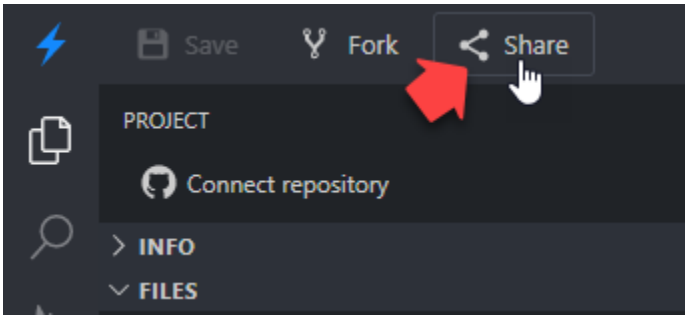
Kopierten Code einfügen

Fertig!

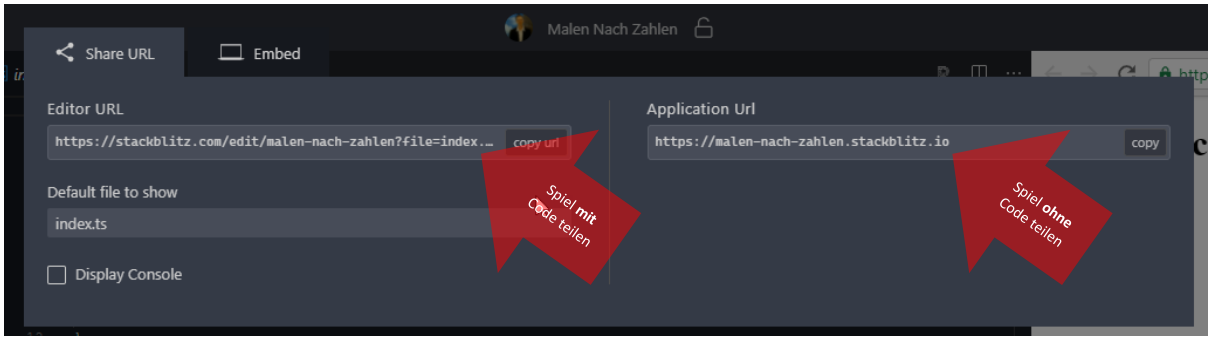
Gratuliere! Dein Spiel ist fertig. Probiere es gleich aus und male den Schmetterling mit schönen Farben aus.

Spiel teilen

Möchtest du das Spiel mit anderen teilen, damit sie es auch ausprobieren können? Wenn du etwas in Stackblitz programmierst, kannst du mit *Share* einen Link abrufen, den du weitergeben kannst.

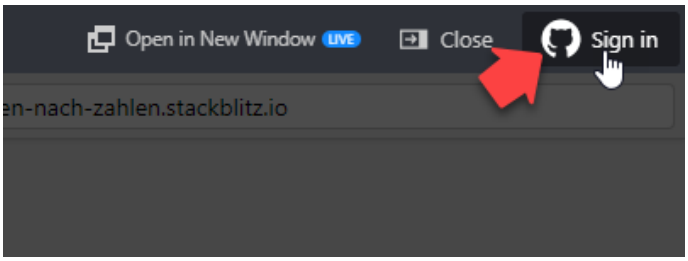


Du kannst wählen, ob du dein Spiel mit oder ohne Code teilen möchtest. Wähle mit Code, wenn die Empfängerin oder der Empfänger weiterprogrammieren möchte. Ohne Code passt besser wenn das Spiel nur gespielt werden soll und der Code für die Empfängerin oder der Empfänger weniger interessant ist.



Spiele speichern

Wenn du öfter in Stackblitz programmierst, ist es praktisch, wenn alle deine Spiele automatisch für dich gespeichert werden. Falls du das möchtest, lege dir ein kostenloses Konto bei GitHub an und melde dich damit bei Stackblitz an. Du kannst auch ein bestehendes Konto verwenden, falls du schon eines hast.



Feuerwerk mit TypeScript

Heute programmieren wir einen Feuerwerk-Simulator. Dabei kannst du das Tippen von Code auf der Tastatur üben und lernst Grundlagen der textuellen Programmierung kennen.

Ziel der Übung

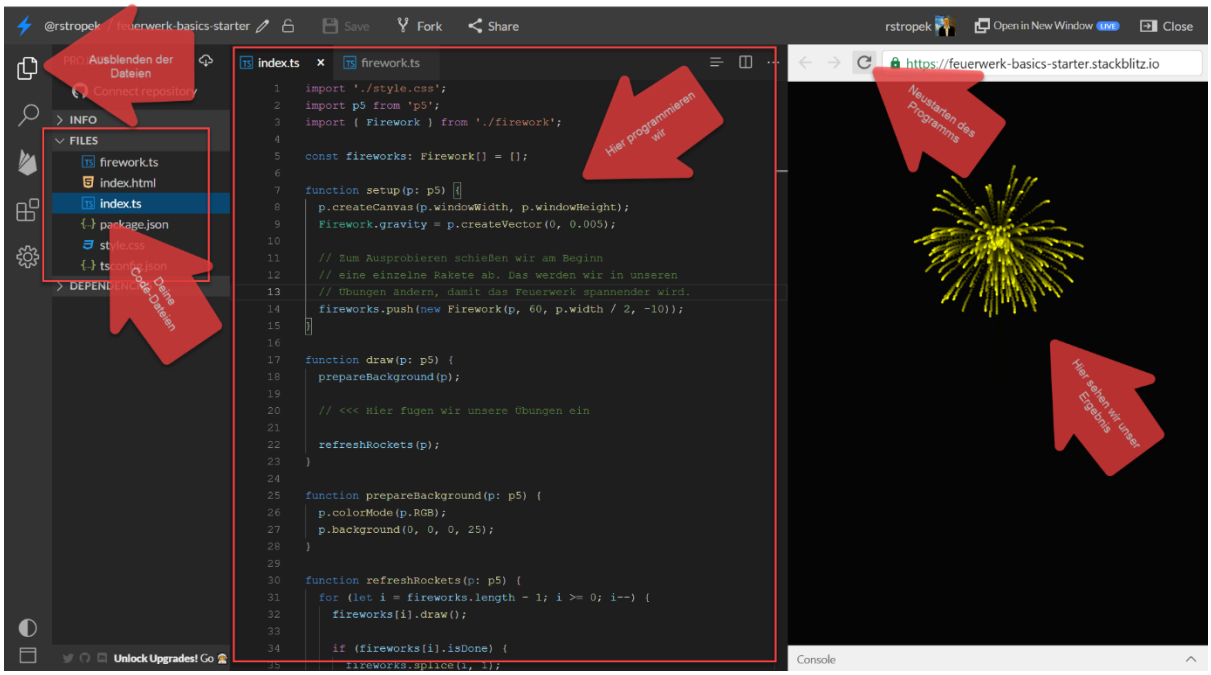
Heute wollen wir gemeinsam einen Feuerwerk-Simulator programmieren. Dabei kannst du das Tippen von Code auf der Tastatur üben und lernst Grundlagen der Programmierung wie Variablen, Schleifen und Funktionen kennen.

Du solltest für diese Übung schon ein wenig Programmiererfahrung mit einer Blockprogrammiersprache wie *Scratch* oder *Snap!* haben. Falls du noch nie Programmcode eingetippt hast, solltest du auch etwas Geduld mitbringen. Es ist am Anfang nicht immer leicht, die vielen Sonderzeichen im Code auf der Tastatur zu finden. Halte durch, Übung macht den Meister!

Start

Du brauchst für diese Übung keine spezielle Software auf deinem Computer zu installieren. Es ist nur ein aktueller Web-Browser (vorzugsweise Chrome) notwendig.

Um loszulegen, öffne die URL https://stackblitz.com/edit/feuerwerk-basics-starter?file=index.ts. Du wirst dadurch im Web-Browser ein Fenster sehen, das in etwa so aussieht:



- Links siehst du eine Liste der Dateien. Wir programmieren heute nur in *index.ts*, daher kannst du die Dateien mit dem Symbol links oben ausblenden (siehe Pfeil Ausblenden der Dateien im Bild oben).
- Unseren Programmcode schreiben wir im Textfenster in der Mitte.
- Rechts sehen wir nach jeder Codeänderung gleich das Ergebnis.

In dieser Übung programmierst du mit der Programmiersprache *TypeScript*. Außerdem verwenden wir die Game Engine *p5js*.

Experimente mit der ersten Rakete

Wir haben dir unter der oben genannten URL schon ein wenig Code zum Starten vorbereitet. Ein Programm wie den Feuerwerksimulator ganz von vorne zu programmieren, ist nichts für AnfängerInnen. Dafür braucht man etwas Übung.

Der von uns vorbereitete Code ist aber ziemlich dumm. Er schießt nach dem Laden der Ergebnisseite nur eine einzelne Rakete ab. Die dafür verantwortliche Zeile in unserem Code ist:

fireworks.push(new Firework(p, 60, p.width / 2, 75, 15));

Deine Aufgabe ist, diese Codezeile zu suchen. Tipp: Sie ist in der Funktion *setup*.

Kommentare

Die Zeilen, die mit // beginnen, sind nur Kommentare. Kommentare schreibt man in den Code, damit man später noch versteht, was man sich gedacht hat, als man den Code geschrieben hat. In diesem Fall verwenden wir Kommentare, um dir zu erklären, was die Codezeilen machen.

Abschießen der ersten Rakete

Hier ein paar Erklärungen zu der Codezeile zum Abschießen der Rakete:

- *fireworks.push* bedeutet, dass wir eine Rakete zu einer Liste von Raketen hinzufügen, die wir auf den Bildschirm zeichnen wollen. Push ist Englisch und bedeutet *schieben* oder *schubsen*. Wir werden später *fireworks.push* oft aufrufen, um mehr als eine Rakete zu zeichnen.
- *new Firework* legt eine Rakete (*Firework* heißt auf Deutsch *Feuerwerkskörper* und *new* heißt *neu*) an. In den Klammern gibt man die Parameter für die Rakete an. Sie sind durch Kommas getrennt. **Achtung:** Achte beim Experimentieren darauf, dass du die Kommas nicht löschst.

Deine Aufgabe ist, mit den Parametern zu experimentieren. Hier ein paar Hinweise dazu:

- Den ersten Parameter *p* ändere bitte nicht. Er ist aus technischen Gründen notwendig und wir gehen heute nicht näher darauf ein.
- Ändere den zweiten Parameter *60* auf einen anderen Wert zwischen *0* und *360*. Hier siehst du eine Darstellung, welcher Wert zu welcher Farbe wird:

- Beim dritten Parameter *p.width / 2* wird die Mitte des Bildschirms errechnet. Du kannst hier statt der Formel auch eine Zahl angeben. Je kleiner die Zahl, desto weiter links erscheint die Rakete, je größer desto weiter rechts. Probiere mal 20, 200 und 400 aus. Siehst du, wie sich die Position der Rakete ändert?
- Der vierte Parameter 75 steuert, wie hoch die Rakete fliegen wird. 0 bedeutet, dass sie kaum abhebt und 100 bedeutet, dass sie bis zum oberen Bildschirmrand fliegen wird. Probiere mal 0, 50, 75 und 100 aus. Siehst du, wie sich die Höhe der Rakete ändert?
- Der fünfte Parameter 15 bestimmt die Größe der Explosion. Probiere mal 10, 15 und 25 aus.

Mehrere Raketen

Code kopieren

Statt einer Rakete möchten wir jetzt mehrere Raketen gleichzeitig abschießen. Deshalb müssen wir die Codezeile zum Abschießen der Rakete, mit der wir gerade experimentiert haben, kopieren. Weißt du schon, wie das geht?

- Stelle deinen Cursor in die Codezeile, die du kopieren möchtest.
- Drücke Strg + C (erst Strg, Strg gedrückt halten, dann C, danach beide Tasten loslassen), um die Zeile zu kopieren.
- Drücke Strg + V, um die Zeile einzufügen.
- Jetzt hast du die Zeile zwei Mal im Code.
- Deine Aufgabe ist es, die Zeile so oft zu kopieren, dass sie fünf Mal im Code ist.

Es macht natürlich keinen Sinn, wenn alle fünf Raketen an der gleichen Stelle losfliegen. Deine nächste Aufgabe ist es, den Code so zu ändern, dass die Raketen über die ganze Bildschirmbreite verteilt sind. So würde das aussehen:

```
fireworks.push(new Firework(p, 60, p.width * 0 / 4, 75, 15));
fireworks.push(new Firework(p, 60, p.width * 1 / 4, 75, 15));
fireworks.push(new Firework(p, 60, p.width * 2 / 4, 75, 15));
fireworks.push(new Firework(p, 60, p.width * 3 / 4, 75, 15));
fireworks.push(new Firework(p, 60, p.width * 4 / 4, 75, 15));
```

Konstanten

Fällt dir auf, dass du die Farbe (60) in jeder Zeile drinnen hast? Wenn du die Farbe ändern willst, musst du fünf Mal die Zahl ändern. Das ist unpraktisch, oder? In einem solchen Fall verwendet man eine Konstante. Deine Aufgabe ist es, den Code so zu ändern, dass die Farbe nur noch einmal enthalten ist und dadurch das Ändern der Farbe einfacher wird. So muss der Code nach der Änderung aussehen:

```
const color = 180;
fireworks.push(new Firework(p, color, p.width * 0 / 4, 75, 15));
fireworks.push(new Firework(p, color, p.width * 1 / 4, 75, 15));
fireworks.push(new Firework(p, color, p.width * 2 / 4, 75, 15));
fireworks.push(new Firework(p, color, p.width * 3 / 4, 75, 15));
fireworks.push(new Firework(p, color, p.width * 4 / 4, 75, 15));
```

Deine nächste Aufgabe ist, nach dem gleichen Prinzip wie bei *color* auch für die Flughöhe der Raketen (vorletzter Parameter) und für die Größe der Explosion (letzter Parameter) Konstanten einzufügen und zu verwenden. So muss der Code nach der Änderung aussehen:

```
const color = 180;
const height = 75;
const size = 15;
fireworks.push(new Firework(p, color, p.width * 0 / 4, height, size));
fireworks.push(new Firework(p, color, p.width * 1 / 4, height, size));
fireworks.push(new Firework(p, color, p.width * 2 / 4, height, size));
fireworks.push(new Firework(p, color, p.width * 3 / 4, height, size));
fireworks.push(new Firework(p, color, p.width * 4 / 4, height, size));
```

Zufallszahlen

Immer die gleiche Farbe ist langweilig, oder? Beim Programmieren kannst du ganz einfach Zufallszahlen erzeugen. Das funktioniert ähnlich wie bei Spielen durch Würfeln. Beim Programmieren kann man aber nicht nur Zufallszahlen zwischen 1 und 6 “würfeln”, sondern beliebige Zahlen zufällig erzeugen lassen. Deine Aufgabe ist es, die Farbe zufällig vom Computer auswählen zu lassen. Du musst dafür nur den Wert der Konstanten *color* wie folgt anpassen:

```
const color = p.random(360);
```

Jedes Mal, wenn du das Programm neu startest, wird der Computer eine zufällige Farbe verwenden.

Raketen im Intervall abschießen

Endlosschleife

Wir möchten unser Programm jetzt so erweitern, dass jede Sekunde Raketen abgeschossen werden. Dafür brauchen wir eine Schleife. Bei unserem ersten Experiment möchten wir fortlaufend neue Raketen abschießen. Deshalb verwenden wir eine Endlosschleife, also eine Schleife, die für immer läuft. In Scratch wäre das die Wiederhole fortlaufend-Schleife.

Deine Aufgabe ist es, den Code so zu ändern, dass jede Sekunde Raketen starten. So muss der Code nach der Änderung aussehen:

```
while (true) {
  const color = p.random(360);
  const height = 75;
  const size = 15;
  fireworks.push(new Firework(p, color, p.width * 0 / 4, height, size));
  fireworks.push(new Firework(p, color, p.width * 1 / 4, height, size));
  fireworks.push(new Firework(p, color, p.width * 2 / 4, height, size));
  fireworks.push(new Firework(p, color, p.width * 3 / 4, height, size));
  fireworks.push(new Firework(p, color, p.width * 4 / 4, height, size));
  await delay(1000);
}
```

Die Zeile *await delay(1000);* wartet 1000 Millisekunden, also eine Sekunde. Deine Aufgabe ist es, mit dieser Zahl zu experimentieren. Wie sieht das Ergebnis aus, wenn du 2000 statt 1000 verwendest? Wie beim Wert von 500?

for-Schleife

Bereit für eine Herausforderung? Lass uns genau 10 Mal Raketen abschießen, dann soll das Feuerwerk zu Ende sein. In solchen Fällen verwendet man eine for-Schleife. Deine Aufgabe ist es, die *while*-Schleife in eine for-Schleife umzubauen. So muss der Code nach der Änderung aussehen:

```
for (let i = 0; i < 10; i++) {
  const color = p.random(360);
  const height = 75;
  const size = 15;
  fireworks.push(new Firework(p, color, p.width * 0 / 4, height, size));
  fireworks.push(new Firework(p, color, p.width * 1 / 4, height, size));
  fireworks.push(new Firework(p, color, p.width * 2 / 4, height, size));
  fireworks.push(new Firework(p, color, p.width * 3 / 4, height, size));
  fireworks.push(new Firework(p, color, p.width * 4 / 4, height, size));
  await delay(1000);
}
```

Hier ein paar Tipps zur *for*-Schleife:

- Sie beginnt bei Null zu zählen (*i = 0*).
- Sie läuft solange *i* kleiner als 10 ist, also zehn Mal.
- Bei jedem Schleifendurchlauf wird i um eins erhöht (*i++*).

Eine weitere *for*-Schleife

Hmmm, könnten wir die for-Schleife nicht auch verwenden, um die vielen, kopierten Aufrufe von *fireworks.push* zu vereinfachen. Deine Aufgabe ist es, eine *for*-Schleife zu verwenden, um aus den fünf *fireworks.push*-Aufrufen einen zu machen. So muss der Code nach der Änderung aussehen:

```
for (let i = 0; i < 10; i++) {
  const color = p.random(360);
  const height = 75;
  const size = 15;
```

```
  for (let j = 0; j < 5; j++) {
    fireworks.push(new Firework(p, color, (p.width * j) / 4, height,
        size));
  }
  await delay(1000);
}
```

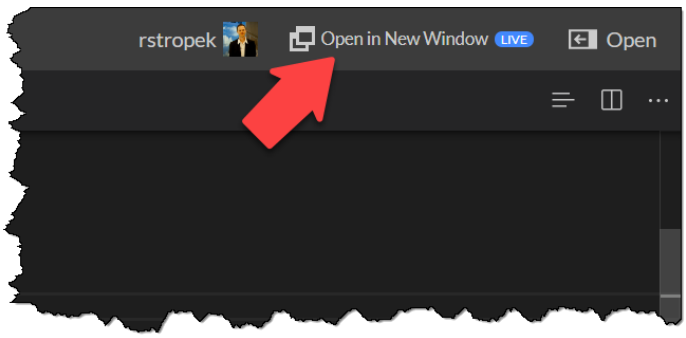
Noch mehr Raketen…

Was müssten wir ändern, damit wir nicht nur fünf Raketen parallel abschießen können, sondern beliebig viele? Wir müssen die Obergrenzen für j anpassen und die Berechnungsformel für die X-Koordinate (horizontale Abschussposition) ändern. Das ist deine nächste Aufgabe. So muss der Code nach der Änderung aussehen:

```
for (let i = 0; i < 10; i++) {
  const color = p.random(360);
  const height = 75;
  const size = 15;
  const numberOfRockets = 10;
  for (let j = 0; j < numberOfRockets; j++) {
    fireworks.push(new Firework(p, color,
        (p.width * j) / (numberOfRockets - 1), height, size));
  }
  await delay(1000);
}
```

Wow, da tut sich ordentlich was, oder?

Hier noch ein Tipp: Wenn du ein richtig großes Feuerwerk sehen willst, das über den ganzen Bildschirm geht, klicke auf Open in *New Window* (rechts oben im Bildschirmfenster), drücke danach *F11* (Vollbildschirm) und zum Schluss F5 zum neu Laden. Cool, oder? Um aus der Vollbildschirmanzeige rauszukommen, drücke einfach nochmals F11.



Raketen in Formation

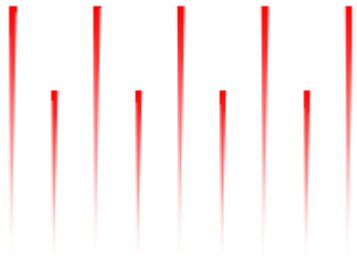
Abwechselnde Höhe

Jetzt wollen wir unsere Raketen in Formation fliegen lassen. Damit wir vom gleichen Startpunkt ausgehen, ist deine Aufgabe, den Code zu ändern, damit er so aussieht:

```
for (let i = 0; i < 100; i++) {
  const height = 75;
  const size = 15;
  const numberOfRockets = 5;
  for (let j = 0; j < numberOfRockets; j++) {
    fireworks.push(new Firework(p, p.random(360),
        (p.width * j) / (numberOfRockets - 1), height, size));
  }
```

```
  await delay(1000);
}
```

Als erstes möchten wir probieren, wie es aussieht, wenn man jede zweite Rakete nur 3⁄4 mal so hoch fliegen lässt. Hier eine Skizze, wie die Flughöhe aussehen soll:



Damit wir diese Herausforderung meistern können, brauchen wir ein wenig Mathematik. In der Schule hast du sicher schon Operationen wie Addition, Subtraktion oder Multiplikation gelernt. Es gibt beim Programmieren jedoch eine weitere Operation, die man häufig braucht: *Modulo*. Modulo gibt den Rest einer Division zurück.

Hier ein paar Beispiele:

Division	Ergebnis	Rest
9 / 5	1	4
10 / 5	2	0
5 / 2	2	1
6 / 2	3	0
0 / 3	0	0

Was hilft uns das bei unserer Herausforderung? Wenn man fortlaufende Zahlen (0, 1, 2, 3, 4, 5 etc.) modulo 2 rechnet, erhält man abwechselnd 0 und 1. Um genau zu sein erhalten wir bei geraden Zahlen die 0 als Ergebnis der Modulo-Operation und bei ungeraden 1. Hier als Tabelle dargestellt:

Zahl	Ergebnis der Zahl modulo 2
0	0
1	1
2	0
3	1
4	0
5	1

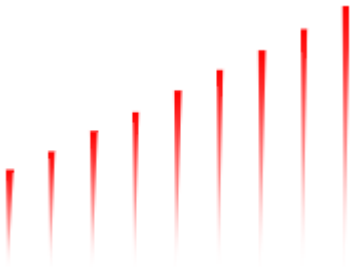
Modulo schreibt man im Code mit dem Zeichen *%*. Deine Aufgabe ist jetzt, mit diesem Wissen den geforderten Formationsflug unserer Raketen zu programmieren. So muss der Code nach der Änderung aussehen:

```
for (let i = 0; i < 100; i++) {
  const size = 15;
  const numberOfRockets = 8;
  for (let j = 0; j < numberOfRockets; j++) {
    let height: number;
    if (j % 2 == 0) height = 90;
    else height = 67.5;
    fireworks.push(new Firework(p, p.random(360),
        (p.width * j) / (numberOfRockets - 1), height, size));
  }
```

```
  await delay(1000);
}
```

Diagonale

Als zweite Formation möchten wir, dass die Raketen eine Diagonale bilden. Die erste Rakete soll auf Höhe 25 fliegen, die letzte auf Höhe 100. Hier eine Skizze, wie die Flughöhe aussehen soll:



Hier brauchen wir wieder etwas Mathematik. In diesem Fall hilft uns Prozentrechnung. Falls du das in der Schule noch nicht gelernt hast, ist das kein Problem. Wenn du magst, bitte eine ältere Person, eine CoderDojo-Mentorin oder einen CoderDojo-Mentor, dir das Prinzip der Prozentrechnung zu erklären. Du kannst aber auch den unten angeführten Code abtippen und Geduld haben, bis ihr Prozentrechnung in der Schule lernt.

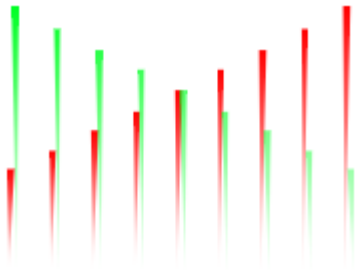
Die Höhe in unserem Fall berechnet sich wie folgt:

- Das Minimum ist 25.
- Zu diesem Minimum addieren wir den Abstand zur Maximalhöhe *(100 - 25 = 75)* multipliziert mit dem Index der Rakete *j* dividiert durch die Anzahl der Raketen *numberOfRockets minus 1*.
- Die Formel lautet also: *height = 25 + 75 * j / (numberOfRockets - 1)*

Deine Aufgabe ist es, diese Formel in unseren Code einzubauen. So muss der Code nach der Änderung aussehen:

```
for (let i = 0; i < 100; i++) {
  const size = 15;
  const numberOfRockets = 8;
  for (let j = 0; j < numberOfRockets; j++) {
    let height = 25 + 75 * j / (numberOfRockets - 1);
    fireworks.push(new Firework(p, p.random(360),
        (p.width * j) / (numberOfRockets - 1), height, size));
  }

  await delay(1000);
}
```

Einen besonders schönen Effekt bekommst du, wenn du zwei Diagonalen gegenläufig erzeugst. Eine steigt von links nach rechts und die andere von rechts nach links.



Das ist schwer theoretisch zu erklären, man muss es probieren. Deine Aufgabe ist es, den Code wie folgt zu ändern:

```
for (let i = 0; i < 100; i++) {
  const size = 15;
  const numberOfRockets = 8;
  for (let j = 0; j < numberOfRockets; j++) {
    let height = 25 + 75 * j / (numberOfRockets - 1);
    fireworks.push(new Firework(p, 60,
        (p.width * j) / (numberOfRockets - 1), height, size));
    fireworks.push(new Firework(p, 120,
        (p.width * j) / (numberOfRockets - 1), 25 + 100 - height, size));
  }

  await delay(3000);
}
```

Text hinzufügen

Zum Abschluss möchten wir noch einen Glückwunsch (z. B. Neujahrswünsche, Geburtstagswünsche) zu unserem Feuerwerk hinzufügen. Das kannst du mit wenigen Zeilen Code in der *draw*-Funktion erledigen. Deine letzte Aufgabe für heute ist es, die *draw*-Funktion zu suchen und die folgenden Zeilen für die Textausgabe hinzuzufügen:

```
function draw(p: p5) {
  p.colorMode(p.RGB);
  p.background(0, 0, 0, 25);

  p.fill('red');
  p.textSize(60);
  p.textStyle(p.BOLD);
  p.textAlign(p.CENTER, p.CENTER);
  p.text('Schönes,\nneues Jahr\n🍾🎊🎉', p.width / 2, p.height / 4);

  // Hier folgt die for-Schleife, die wir zuvor programmiert haben
  for (let i = fireworks.length - 1; i >= 0; i--) {
    ...
  }
}
```

Der Druck dieser Übungsanleitung wurde unterstützt durch die Digitale Meile Linz (https://techharbor.at/hafenlinz). Wir bedanken uns für die Unterstützung!


CODING
CLUB
LINZ