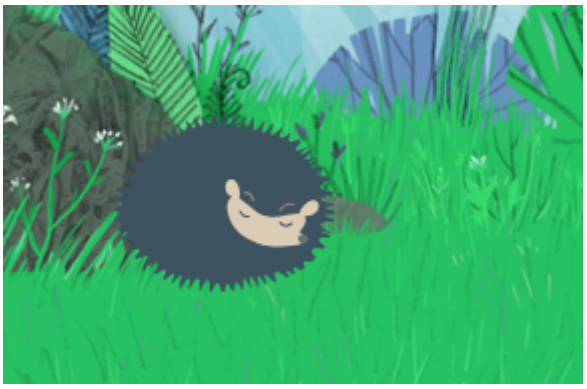
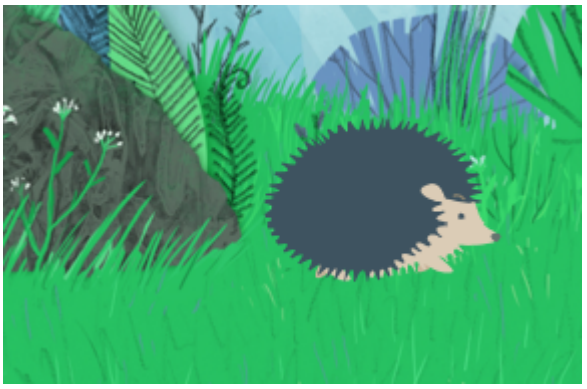
Scratch Pattern III

State Machine

Manche Figuren machen unterschiedliche Dinge, je nach Situation.
Der Igel schläft manchmal und manchmal läuft er umher.

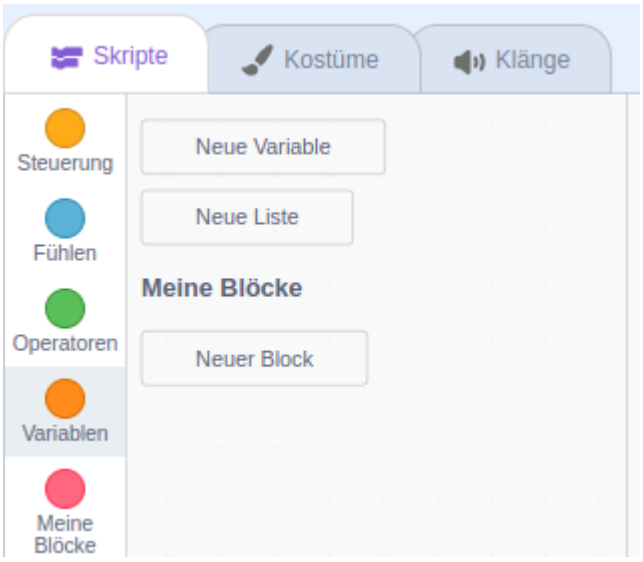


Igel schläft



Igel läuft

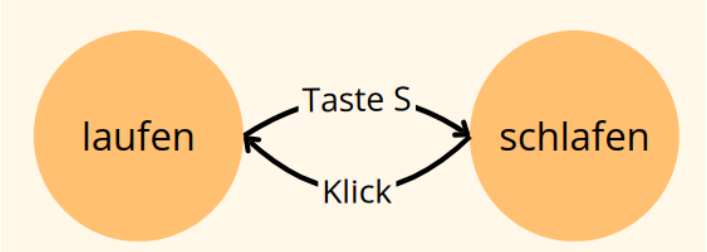
Immer wenn man so eine Figur hat, die sich manchmal so und manchmal anders verhält, kann man eine **StateMachine** verwenden.
Für eine StateMachine brauchen wir eine Variable, die sich merkt, in welchem Zustand wir gerade sind (zum Beispiel schlafen oder laufen). Das kann man auf dem Menü auf der linken Seite machen:



Im orangenen Bereich “Variablen” auf “Neue Variable” klicken. Dann kann man den Namen der Variablen eingeben (zum Beispiel “zustand”).
In dieser Variable speichern wir, ob wir gerade schlafen oder laufen.

Auf der nächsten Seite zeigen wir, wie man zwischen den Zuständen wechseln kann.

Um sich besser zu verdeutlichen, wann welcher Zustand wechselt, können wir ein Diagramm zeichnen:



Dieses Diagramm zeigt die beiden Zustände “laufen” und “schlafen” und wie man zwischen diesen Zuständen wechselt.

Wenn der Igel schläft, können wir ihn anklicken, damit er aufwacht.
Wenn der Igel läuft, brauchen wir nur die Taste “S” zu drücken, damit er wieder einschläft.

Wenn wir unser Spiel starten, indem wir auf die grüne Flagge klicken, setzen wir unsere Variable “zustand” auf “laufen”.
Das heißt, wir beginnen damit, dass unser Igel läuft.
Damit der Igel wirklich laufen kann, müssen wir diese Fähigkeit erstmal programmieren.

Das heißt: Falls wir gerade laufen sollen, drehen wir uns zum Mauszeiger und gehen einen kleinen Schritt. Außerdem wechseln wir noch zu einem passenden Kostüm.

Schlafen ist Gott sei Dank ganz einfach. Wir müssen nur zum Schlafen-Kostüm wechseln.

Damit können wir schon laufen und schlafen. Aber wie können wir abwechselnd das eine oder das andere machen? Dafür müssen wir noch zwischen Zuständen wechseln.

Das ist glücklicherweise nicht schwer:
Wenn wir die Figur anklicken, dann wechseln wir zum Laufen. Wenn man extra vorsichtig sein möchte, kann man noch abfragen, ob wir aktuell schlafen. Es könnte ja sein, dass wir später weitere Zustände haben, bei denen wir nicht zum Laufen wechseln wollen.

Wenn wir die Taste S drücken, wechseln wir zum Schlafen. Probier es einfach mal aus und schau, ob es funktioniert. Auf der nächsten Seite findest du noch ein paar Aufgaben zur Inspiration.

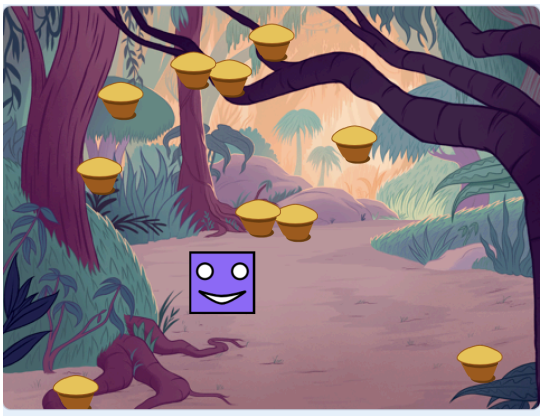
Aufgaben - State Machine

- [ ] Mein Igel sieht müde aus. Er würde viel lieber mit Schlafen anfangen. Wie geht das?
- [ ] Das Laufen sieht noch doof aus. Wie schafft man es, dass eine Laufanimation entsteht?
- [ ] Im Projekt findest du noch einen Apfel, der herunterfällt, wenn man auf ihn klickt. Was soll dann passieren? Können wir einen neuen Zustand erfinden? Er könnte sich zum Beispiel freuen, wenn er den Apfel einsammelt.
- [ ] Der Apfel fällt viel zu langsam. Schau im Kapitel Sprungtechnik nach, wie der Apfel realistischer fallen kann.
- [ ] Soll etwas anderes passieren, wenn der Igel im Schlaf von einem Apfel getroffen wird? Vlt ärgert er sich dann?
- [ ] Kennst du schon die Abteilung “Meine Blöcke”? Manchmal wird eine StateMachine zu unübersichtlich. Dann macht es Sinn jeden Zustand in einen eigenen Block zu packen!

Klone

Willst du ein Spiel entwickeln, wo du muffins sammelst und damit auflevelst? Dafür kannst du Klone verwenden.

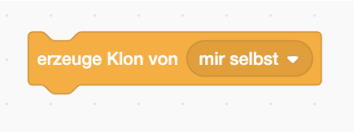
So sieht unser Spiel aus. Darin gibt es eine Figur, die Muffins sammelt. Es gibt 10 muffins und wenn die Figur einen isst, entsteht ein neuer.



Zuerst wählen wir uns eine Figur, die wir sammeln möchten. Für unser Spiel nehmen wir den Muffin.

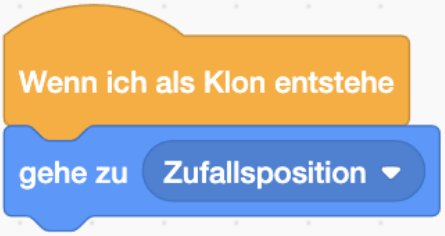
So sieht das Spiel nach dem ersten Schritt aus. Jetzt wollen wir zehn Klone von dem Muffin hinzufügen.

Aus dem Bereich “Steuerung” benutzen wir diese Operation, um einen Klon vom Muffin zu erzeugen. Es gibt aber ein Problem…



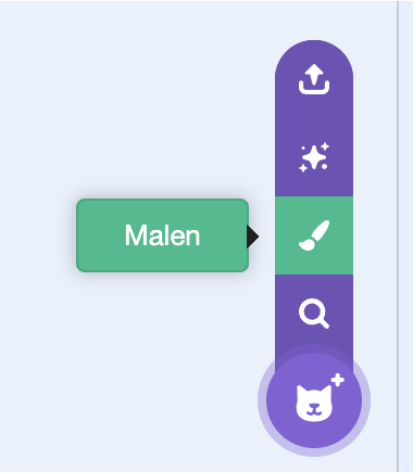
Wenn wir erstmal einen Klon erzeugen, können wir den Klon nicht sehen! Aber warum? Das liegt daran, weil der Klon am selben Platz erzeugt wurde. Beide Klone stehen aufeinander.

Um dieses Problem zu lösen, können wir diese zwei Operationen benutzen. Wenn der Klon entsteht, ziehen wir den Klon zu einer Zufallsposition.

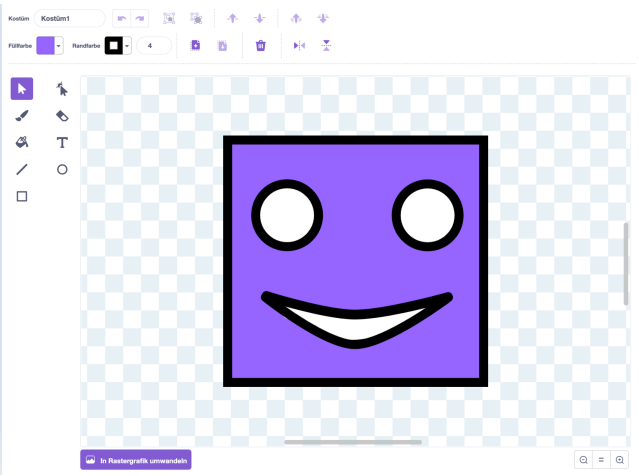


Damit ist das Problem gelöst.
Wir können den Muffin sowohl anzeigen als auch klonen und wir sind bereit, den nächsten Schritt zu machen. Nämlich die Muffins einzusammeln.

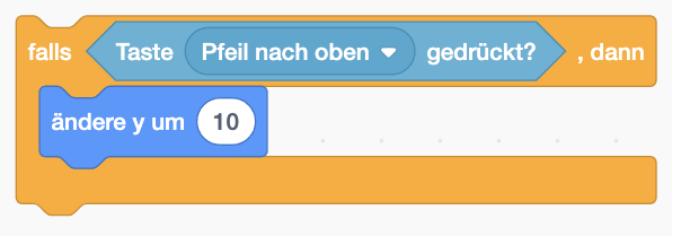
Diesmal malen wir uns eine Figur, statt eine zu wählen.



Ich habe einen SmileySquare gemalt, du kannst deine eigene Figur malen. Der nächste Schritt ist, die Figur mit den Pfeilen zu bewegen.

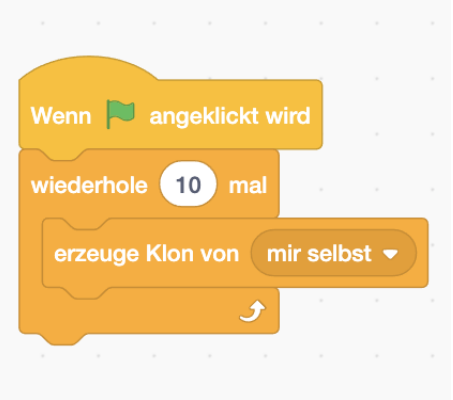


Um die Position der Figur zu ändern, müssen wir die X- und die Y-Achse ändern. Hier schauen wir, falls der Pfeil nach oben gedrückt wird, addieren wir 10 zur Y-Achse.



Dasselbe machen wir auch mit den anderen Pfeilen in einem “wiederhole fortlaufend” Block. Dort senden wir auch eine Nachricht “ Game loop”, die den anderen Figuren bescheid gibt, dass das Spiel noch läuft.
Jetzt gehen wir wieder in die Muffin Figur.

Du kannst es entscheiden, wie viele Muffins du zeigen willst. Hier erzeugen wir nur 10 Klonen.



Als letzten Schritt checken wir, ob der Sammler berührt wird. Dann löschen wir den Klon, den der Sammler gegessen hat. Dann erzeugen wir einen neuen Klon, um den Klon zu ersetzen, der gerade gegessen wurde.

Aufgaben - Level up

- [ ] Kannst du einen Score anzeigen? Benutz eine Variable und zeige, wie viele Muffins schon gegessen wurden.
- [ ] Willst du aufleveln? Du kannst auch ein Levelsystem entwickeln.
- [ ] Du kannst immer alles einfach sammeln. Füg schlechte Figuren hinzu, die Minuspunkte geben, wenn der Sammler sie isst.

Parallax Scrolling + Sprungmechanik

Parallax Scrolling

Du sitzt im Auto und schaust aus dem Fenster.
Siehst du, dass Sachen, die **sehr nah** sind, **sehr schnell** vorbeiziehen?

- Bäume am Straßenrand
- Menschen in der Stadt
- Andere Autos

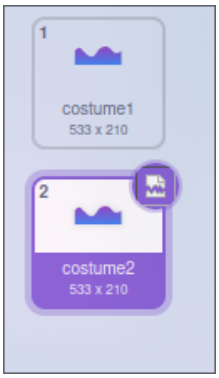
Sachen, die **weit weg** sind, ziehen **langsamer** vorbei.

- Wolken
- Berge

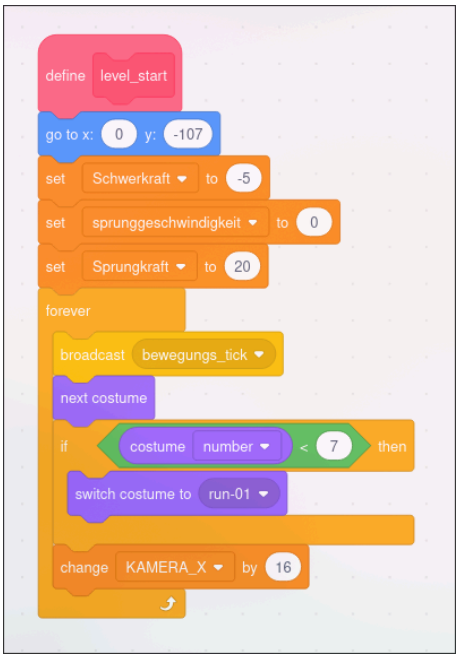
Damit unser Spiel cooler aussieht, haben wir das in Scratch auch so gemacht!

- Die Berge im **Hintergrund** ziehen **langsam** vorbei
- Die Bäume im **Vordergrund** ziehen **schneller** vorbei
- Die Sonne ist so weit weg, dass sie sich gar nicht bewegt

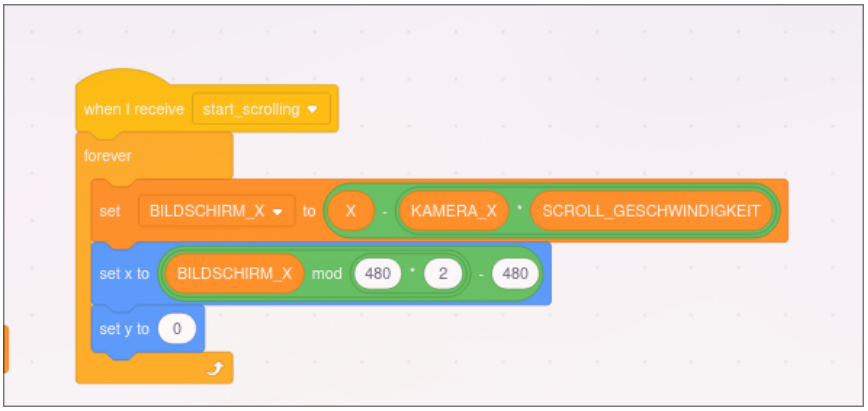
So sieht es in Scratch aus:



Hintergrund und Vordergrund existieren zwei Mal, und werden immer von rechts reingeschoben.

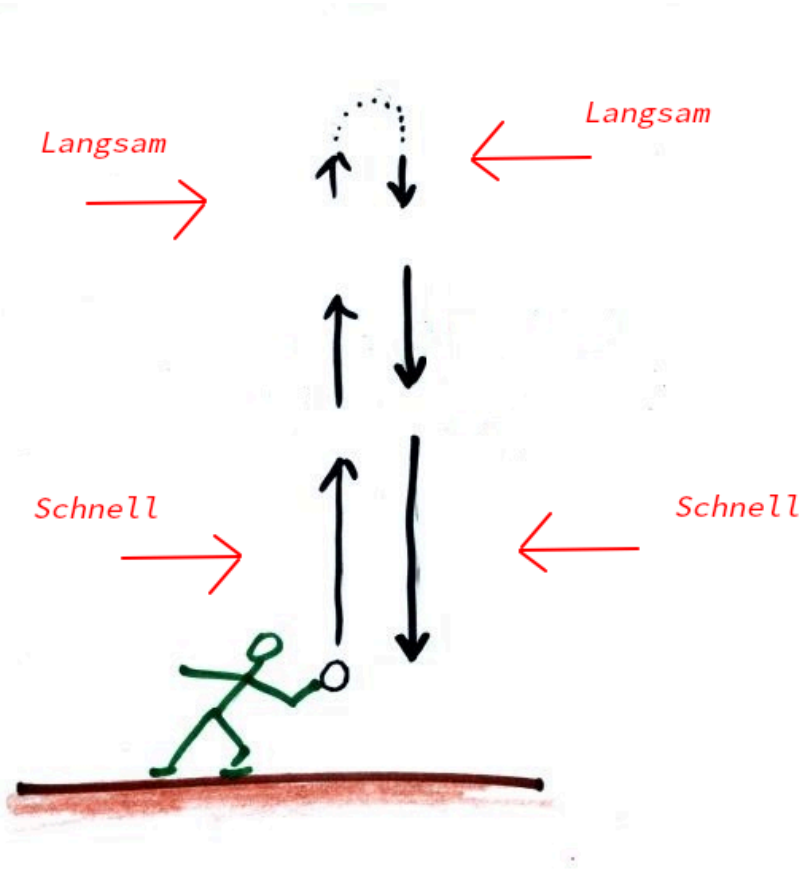


Scratchy läuft immer ein bisschen (Bewegungstick) und schiebt dann mit **Kamera_X** die Position von Hintergrund und Vordergrund ein wenig nach links.



Hintergrund und Vordergrund schieben sich dann mit ein bisschen Mathe basierend auf **Kamera_X** nach links.

Sprungmechanik



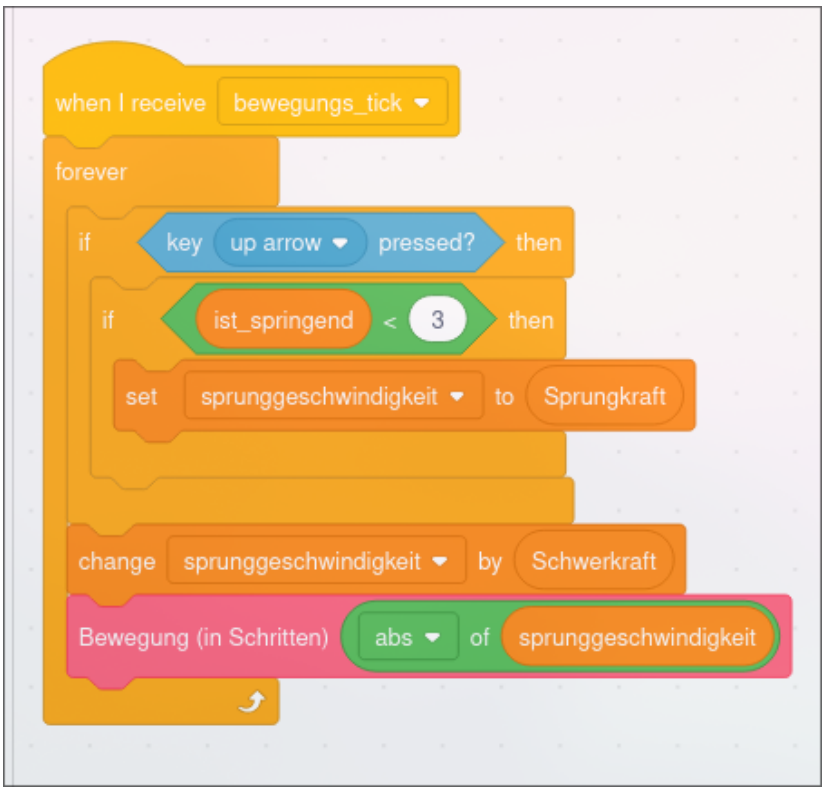
Du wirfst einen Ball senkrecht nach oben.
Siehst du, dass er **am Anfang schnell** ist, dann **immer langsamer** wird, und schließlich, wenn er oben ankommt und herunterfällt, wieder **schneller wird**?

Das liegt an der **Schwerkraft,** welche den Ball versucht, auf die Erde zurückzuziehen.

Beim Springen ist es genauso!

Damit Scratchy beim Springen realistischer aussieht, haben wir Schwerkraft eingebaut.
Probier es aus!

So sieht es in Scratch aus



Jeden Tick ziehen wir die Schwerkraft von der Sprunggeschwindigkeit ab. Damit wir Scratchy nicht unendlich hoch springen lassen können, checken wir mit ist_springend, ob Scratchy schon im Sprung ist oder nicht.

Ideen für Aufgaben

- [ ] Schau dir den Code an. Verstehst du, wie das Parallax Scrolling funktioniert?
- [ ] Verstehst du, wie Scratchys Springen funktioniert?
- [ ] Findest du heraus, wie man Scratchy höher springen lassen kann?
- [ ] Findest du heraus, wie Scratchy einen Doppelsprung machen kann?
- [ ] Wie kann Scratchy schneller rennen?
- [ ] Kannst du nur den Hintergrund schneller oder langsamer vorbeiziehen lassen?
- [ ] Schaffst du es, einen Vogel/Mond/Flugzeug/…. im Hintergrund vorbeifliegen zu lassen?