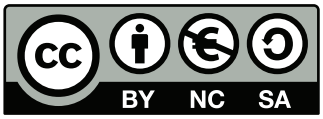
Programmieren für Kinder mit

Bernd Gärtner (ETH Zürich und Kinderlabor)
Katharina Daun (ETH Zürich)



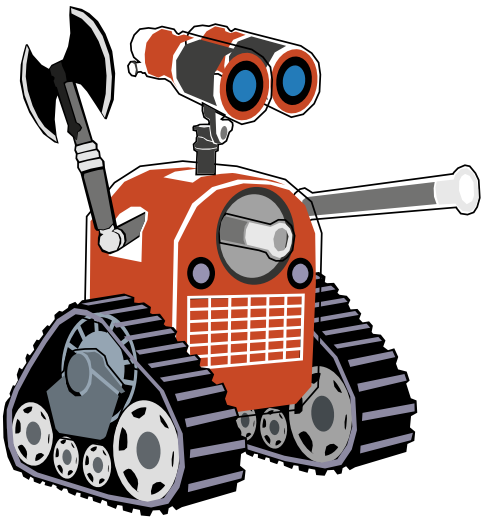
Inhaltsverzeichnis

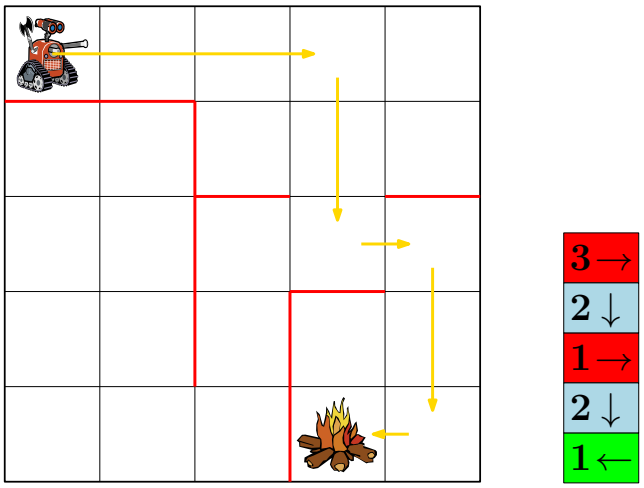
Kapitel 1

Programmieren

Hier erfährst du, was Programmieren bedeutet und was Computerprogramme sind. Du lernst, dir aus Legosteinen Programme zur Steuerung eines vorgestellten Feuerlöschroboters zu bauen. Du kannst die Steuerung des Löschroboters hier auch „in echt“ ausprobieren:

https://scratch.mit.edu/projects/16532746





Die **Begriffe**, die hier neu eingeführt werden:

- Programmieren
- Befehl
- Programmiersprache
- Programm

1.1 Zeichenerklärung

Hier wird ein Scratch-Programm vorgestellt, das du selbst am Computer ausprobieren sollst. Diese Programme findest du unter https://scratch.mit.edu/studios/26899250/.

Hier steht, was passiert, wenn du das korrekt zusammengebaute Programm ausführst.

Hier wird ein wichtiger Begriff oder ein neuer Scratch-Befehl eingeführt.

Hier bekommst du Tipps oder Hinweise.

Aufgabe 1.1

In jedem Kapitel gibt es viele Aufgaben, die dir beim Scratch-Lernen helfen! Es gibt verschiedene Aufgabentypen:

Bei diesen Aufgaben sollst du etwas in Scratch programmieren.

Diese Aufgaben sollst du auf Papier lösen, ohne zu programmieren. Du kannst am Schluss aber deine Lösung am Computer überprüfen.

Bei diesen Aufgaben sollst du mit deinen Klassenmitgliedern diskutieren.

Hier musst du den Fehler suchen.

1.2 Was ist Programmieren?

Programmieren heisst, einem Computer (z.B. in einem Roboter) eine Folge von Befehlen zu erteilen, damit er genau das macht, was du von ihm willst. Computer verstehen nur ganz bestimmte einfache Befehle, deshalb brauchst du oft viele davon.

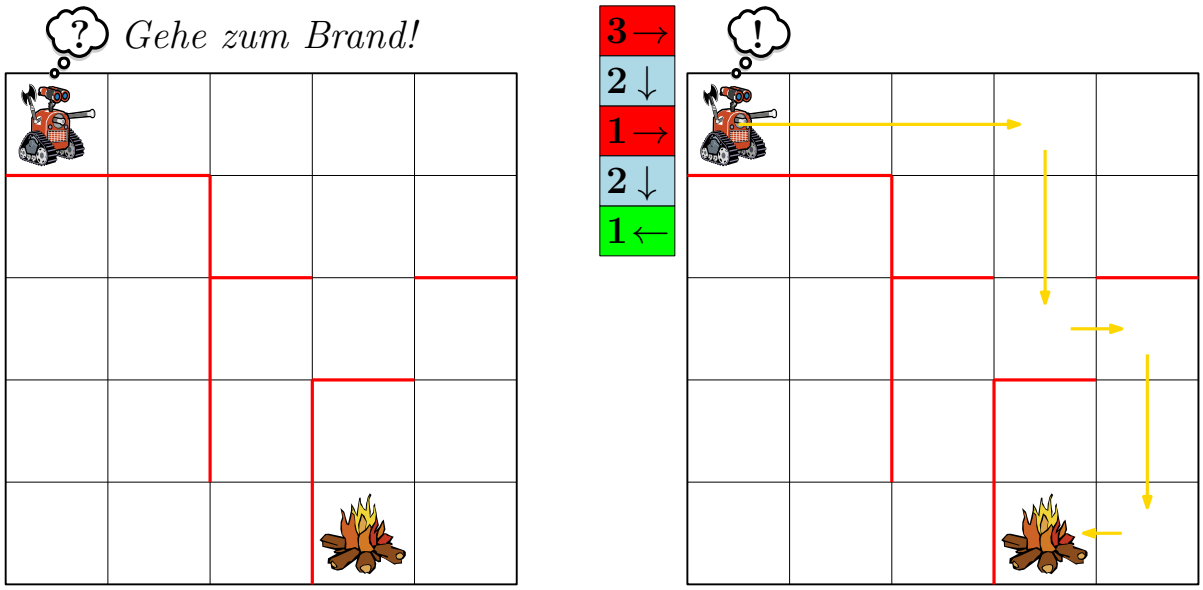


Abbildung 1.1: Computer verstehen unsere Sprache nicht (links), deshalb müssen wir ihre Sprache lernen, wenn wir wollen, dass sie etwas für uns tun (rechts). Die Sprache des Löschroboters sprechen wir mit bunten Blöcken, die von oben nach unten gelesen werden. Kannst du herausfinden, was die bunten Blöcke bedeuten?

Aufgabe 1.2

Hast du oder jemand in deiner Familie schon einmal einen Computer programmiert? Wenn ja: wie und zu welchem Zweck?

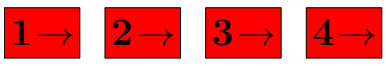
Aufgabe 1.3

In vielen Haushaltsgeräten ist ein Computer versteckt, den du programmieren kannst. Nenne ein paar Beispiele!

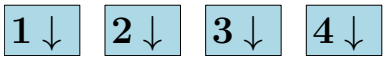
1.3 Programmiersprachen

Eine *Programmiersprache* ist eine Sprache, die der Computer versteht. Sie besteht aus Befehlen, die du im Umgang mit dem Computer benutzen kannst.

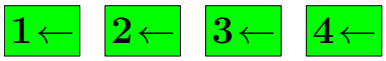
Die Programmiersprache, die der Löschroboter versteht, besteht aus den folgenden 17 Befehlen. Der Roboter kann nicht durch die roten Wände gehen, kann sie aber vorher mit seiner Axt zerstören. Der Roboter soll auch das Spielfeld nicht verlassen.



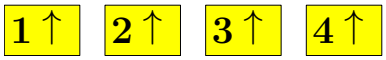
Gehe 1, 2, 3, 4 Kästchen nach rechts



Gehe 1, 2, 3, 4 Kästchen nach unten

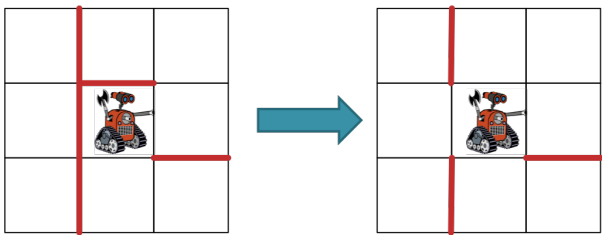


Gehe 1, 2, 3, 4 Kästchen nach links

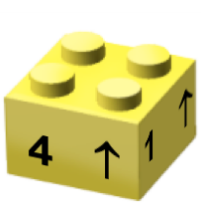


Gehe 1, 2, 3, 4 Kästchen nach oben

Benutze die Axt:
Alle Wände, die direkt um das aktuelle Kästchen des Löschroboters stehen, werden zerstört:



Du kannst dir die Befehle ganz einfach aus Legosteinen basteln. Beschrifte jede der vier Seiten eines Steins mit einem der vier Befehle für die Richtung.



Richtungsbefehle

Axt

Aufgabe 1.4

Der Löschroboter versteht 17 verschiedene Befehle (siehe vorherige Seite). Könnte man auch mit weniger Befehlen auskommen? Welche Befehle sind unbedingt notwendig, damit der Roboter seine Löschaufgabe immer erledigen kann?

Aufgabe 1.5

Bist du in Aufgabe 1.4 zum Schluss gekommen, dass es mehr Befehle gibt als nötig? Dann versuche zu erklären, warum es sinnvoll sein kann, mehr Befehle zur Verfügung zu haben, als man eigentlich braucht.

1.4 Programme

Ein *Programm* besteht aus einer oder mehreren Befehlsfolgen. Programme werden in einer Programmiersprache aufgeschrieben und danach vom Computer ausgeführt.

Nicht jedes Programm ist korrekt. Es gibt Programme, die den Löschroboter nicht zum Brand bringen, aber auch Programme, bei denen er kaputt geht (wenn er über den Rand des Gitters oder gegen eine Wand fährt).

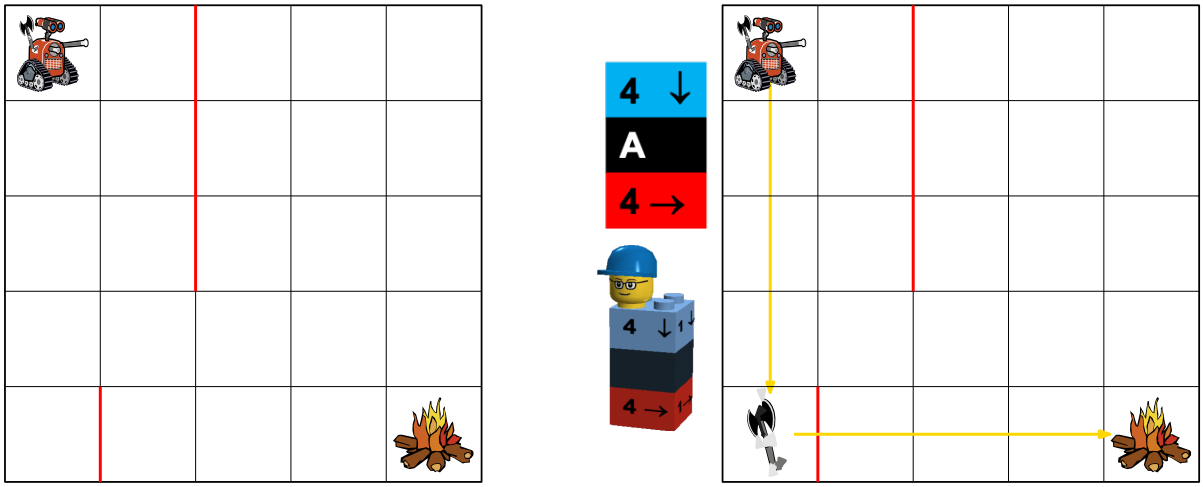


Abbildung 1.2: Links: Die Ausgangssituation; Mitte: das Programm – Befehle werden von oben nach unten gelesen; Rechts: Ausführung des Programms durch den Löschroboter.

Aufgabe 1.6

Welche der folgenden Programme bringen den Löschroboter in der Ausgangssituation von Abbildung 1.2 zum Brand? Bei welchen Programmen kommt er nicht beim Brand an, und bei welchen geht er sogar kaputt?

a)

4 →
A
4 ↓

b)

4 ↓
A
4 →

c)

3 ↓
3 →
2 ↑
1 →
3 ↓

d)

1 →
A
2 →
4 ↓
A
1 →

e)

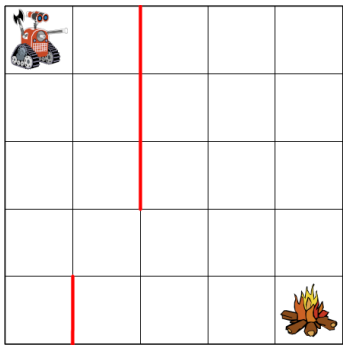
3 ↓
3 →
1 ↑
1 →

f)

1 →
A
2 ↓
3 →
1 ↓

Aufgabe 1.7

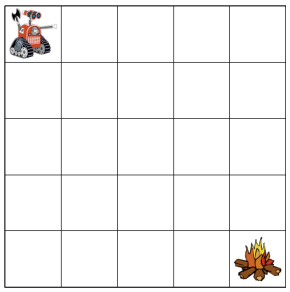
Schreibe alle Programme auf, die den Löschroboter in der Ausgangssituation unten mit drei Befehlen zum Brand steuern!



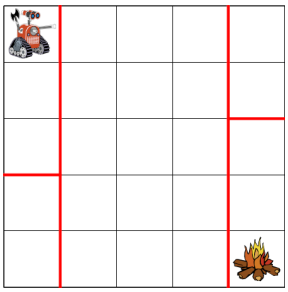
Aufgabe 1.8

Schreibe für jede der folgenden Ausgangssituationen ein Programm, das den Löschroboter zum Brand steuert. Versuche dabei jeweils, ein Programm zu finden, das aus möglichst wenigen Befehlen besteht!

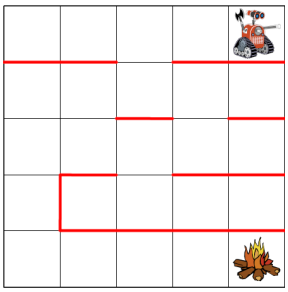
a)



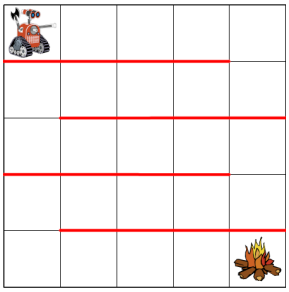
b)



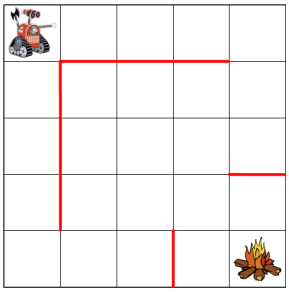
c)



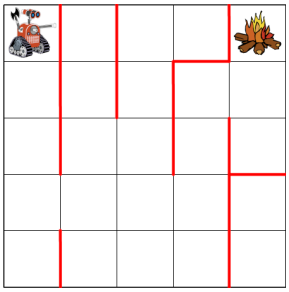
d)



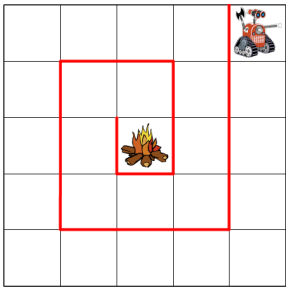
e)



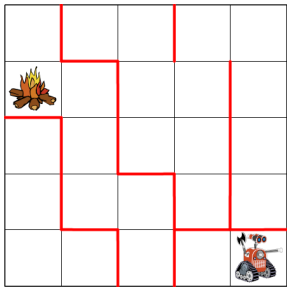
f)



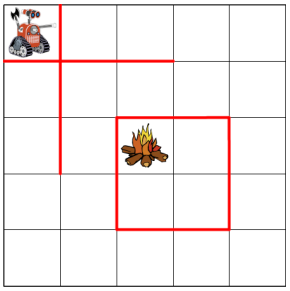
g)



h)



i)



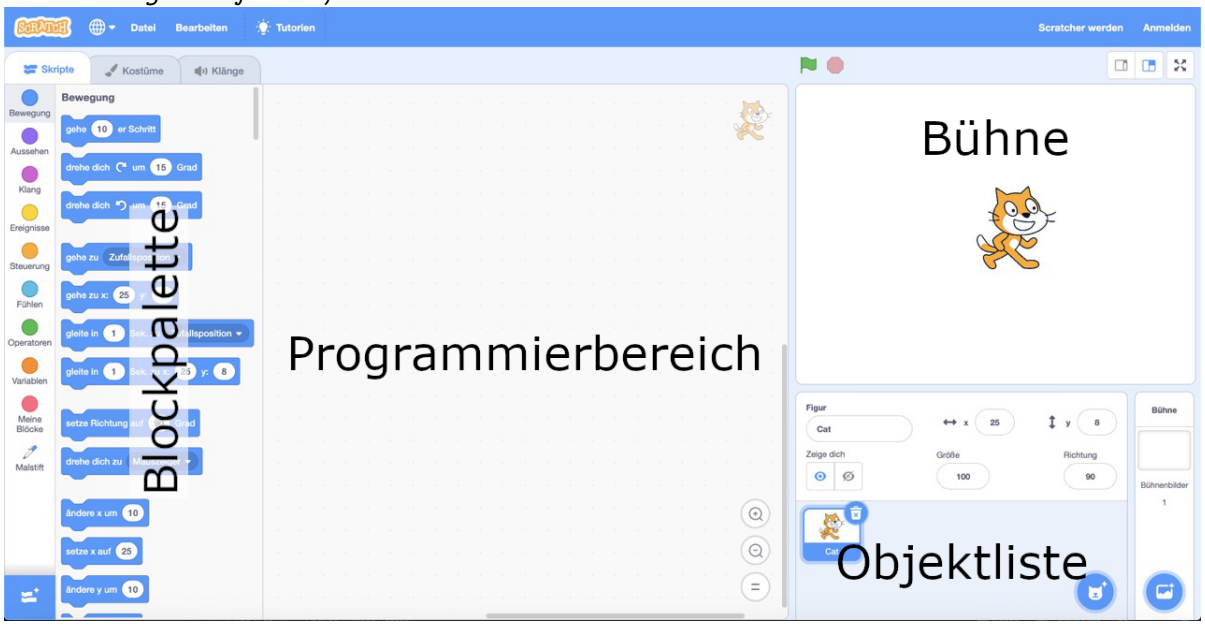
Kapitel 2

Erste Schritte mit Scratch

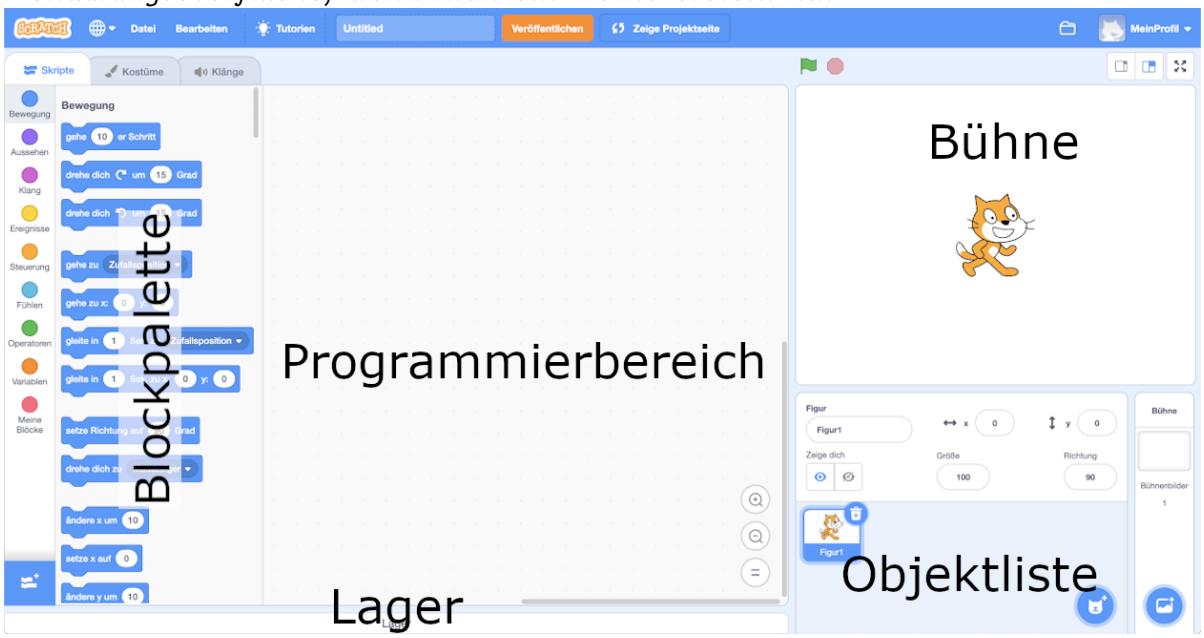
Scratch ist eine Programmiersprache, die es dir ermöglicht, deine eigenen interaktiven Geschichten, Animationen, Spiele, Musik und Kunstwerke zu erstellen und sie als *Scratch-Projekte* anderen über das Internet mitzuteilen. Bei Scratch programmierst du mit Blöcken, die du wie Legosteine stapelst, um dein Programm zusammenzubauen.
Für weitere Informationen gehe zu: http://scratch.mit.edu/
Um Scratch zu benutzen musst du keinen Account erstellen. Du kannst damit aber deine Projekte mit anderen teilen und sie auch von anderen Computern aufrufen.

2.1 Die Benutzungsoberfläche

Je nachdem, ob du einen Account erstellt hast oder nicht, sieht die Benutzungsoberfläche etwas anders aus.
Benutzungsoberfläche, wenn man kein Konto erstellt hat:



Benutzungsoberfläche, wenn man ein Konto erstellt hat:



Bühne Hier läuft dein *Projekt* (Geschichte, Animation, Spiel, Musik...) ab. Die handelnden Lebewesen und Gegenstände in deinem Projekt heissen *Objekte*.

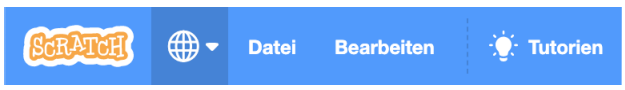
Blockpalette Von hier holst du die Blöcke, aus denen du dann im Programmierbereich dein Programm zusammenbaust.

Programmierbereich Hier baust du Skripte (Blockstapel), die den Objekten sagen, was sie tun sollen. Die Stapel aller Objekte zusammen bilden dein *Programm*.

Objektliste Um ein Objekt zu programmieren, wählst du es hier aus. Am Anfang gibt es nur ein Objekt, die Katze Scratch.

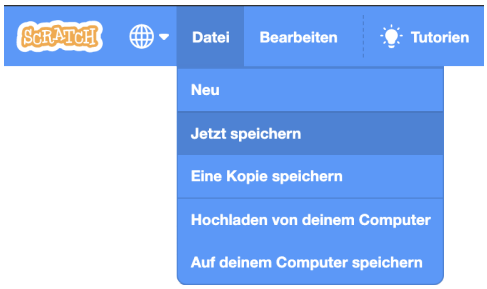
Lager (nur mit Account) Im Lager kannst du Skripte (Blockstapel) speichern, die du in anderen Projekten benutzen willst.

Mit einem Klick auf die Weltkugel oben links kannst du die Sprache ändern:

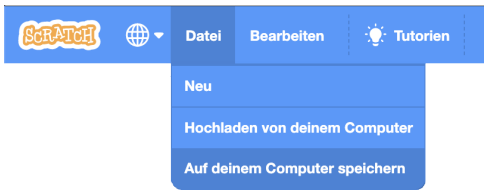


2.2 Projekte speichern

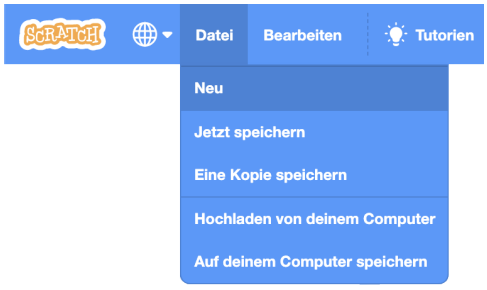
Wir werden unsere Programme immer wieder etwas verändern und erweitern. Denke daran, dein Projekt regelmässig zu speichern! Scratch speichert aber auch ab und zu dein Projekt automatisch ab!



Wenn du keinen Account hast, kannst du deine Projekte auf dem Computer als Datei speichern.

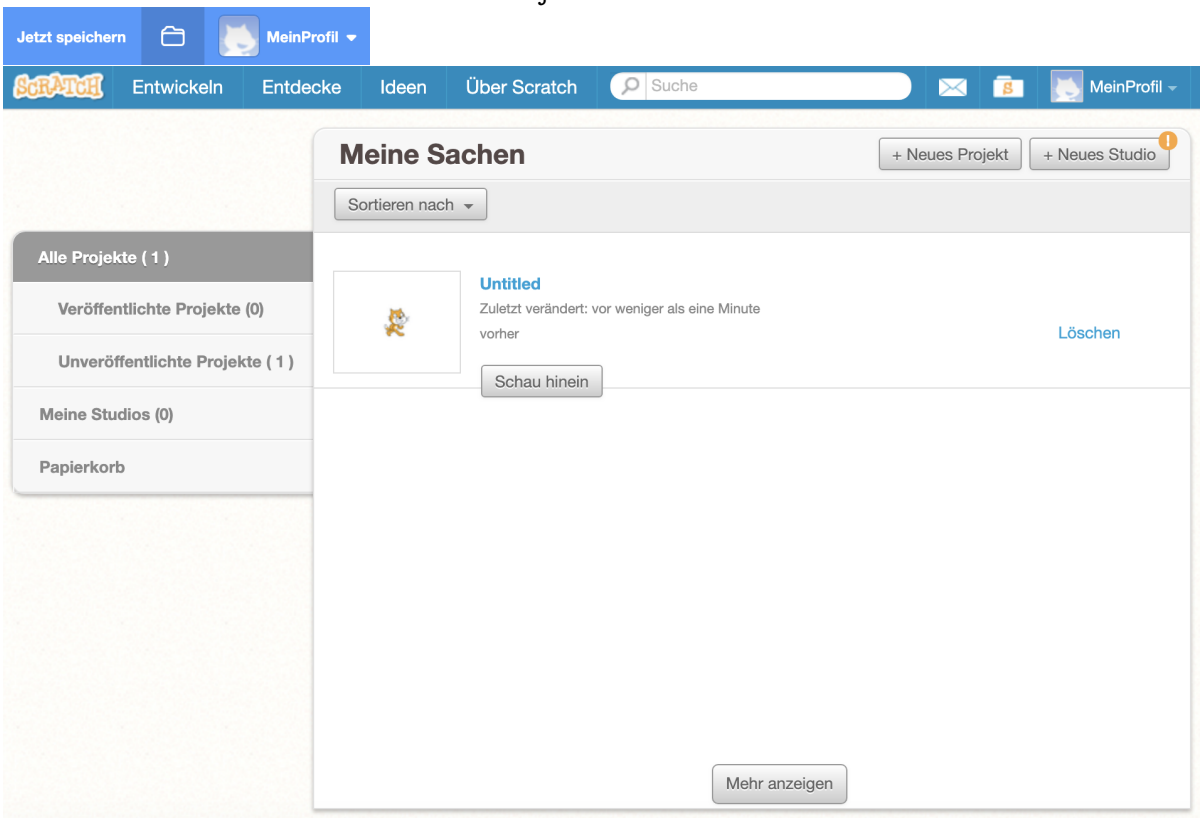


Lege am besten eine Ordnerstruktur an, damit du deine Projekte gut wiederfindest. Wenn du ein Projekt weiter bearbeiten willst, klicke auf "Hochladen von deinem Computer" und wähle dann die Datei aus.
Projekt-Dateien haben die Dateiendung ".sb3". Mit Scratch kannst du aber nicht nur ganze Projekte sondern auch einzelne Objekte oder Kostüme speichern. Achte beim Speichern und Hochladen darauf, welche Dateiendungen du entdeckst und zu welcher Art von Darstellung sie gehören (Bild, Ton, Text, Video).
Wenn du ein neues Projekt starten willst (und das jetzige nicht mehr überschrieben werden soll), wähle das folgendermassen aus:



Gib jedem neuen Projekt einen passenden Namen, damit du es schnell wiederfindest, wenn du es weiter bearbeiten willst.

Wenn du einen Account erstellt hast, kannst du all deine Projekte mit einem Klick auf das Ordner-Symbol oben links anschauen. Mit einem Klick auf "Schau hinein" kommst du dann wieder zurück zu deinem Projekt und kannst es weiter bearbeiten.



Aufgabe 2.1

Was für Vor- und Nachteile gibt es, wenn man sein Projekt auf dem Computer speichert gegenüber dem Speichern über das Internet? Gibt es noch andere Möglichkeiten, wo man ein Projekt speichern könnte?

Kapitel 3

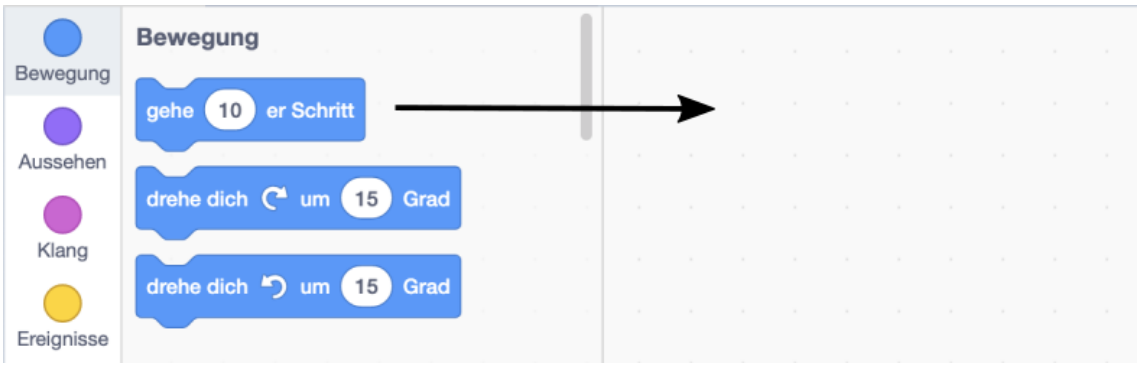
Dein Erstes Projekt

In diesem Kapitel realisierst du dein erstes interaktives Projekt. Du bringst einem Objekt das Laufen und Malen bei und steuerst es über die Tastatur.
Die Blöcke, die du dabei neu kennenlernst:

Die Begriffe, die hier neu eingeführt werden:

- (Scratch-)Befehl
- Parameter
- Skript (Befehlsfolge)
- Attribut (Merkmal)
- Hut (Ereignisbehandlung)
- Ansichten: Entwicklungsansicht und Präsentationsmodus
- Versionierung, Versionsnummer
- Studio

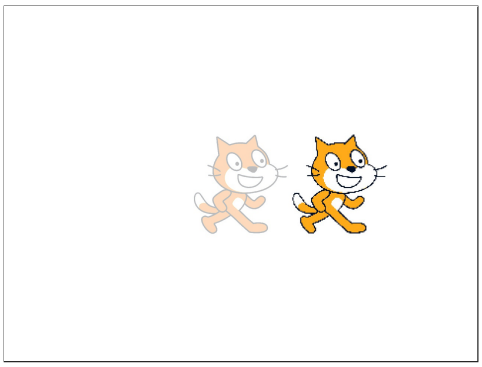
3.1 Geradeaus gehen



Ziehe den Block 

aus der Blockpalette in den Programmierbereich. Klicke ihn dann ein paar Mal an und beobachte die Bühne!

Die Katze Scratch geht bei jedem Klick 10 Schritte (= Bildpunkte oder Pixel) geradeaus. Nach zehnmal Klicken und 100 Schritten ist Scratch hier gelandet:



Die 10 kannst du auch durch eine beliebige andere Zahl ersetzen, um Scratch mehr oder weniger Schritte laufen zu lassen.

Jeder Block mit einer Vertiefung  oder Ausbuchtung  ist ein *Befehl*. Der Text im Block beschreibt den Befehl. Bei jedem Anklicken des Blocks führt dein Objekt den Befehl einmal aus.

Die Zahl im Block ist ein *Parameter*. Die Grundfunktion eines Befehls wird durch Ändern des Parameters nicht verändert, man kann damit aber dem Computer genauer mitteilen, was er tun soll. Ohne den Parameter weiss der Computer bei den meisten Befehlen nicht, was er tun soll, und es passiert nichts. Bei manchen Befehlen kann man den Parameter nur aus einer Liste auswählen.
Beispiel: Mit dem 

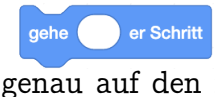
Block bewegt sich Scratch und der Parameter (hier 10) definiert um wieviele Schritte.

Aufgabe 3.1

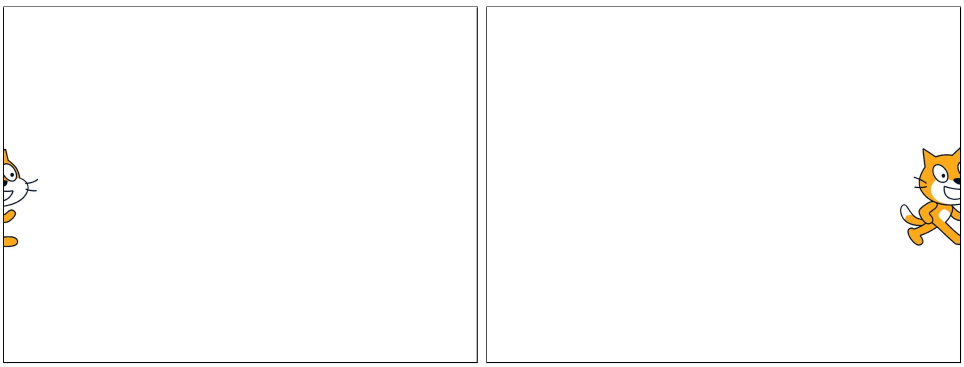
Wie kannst du die Katze Scratch dazu bringen, rückwärts zu laufen? Probiere deine Lösung aus und schreibe sie hier auf! Du darfst dabei aber nur den Befehl 

benutzen!

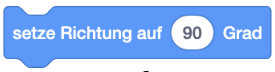
Aufgabe 3.2

Versuche herauszufinden, wie viele Schritte (=Bildpunkte oder Pixel) die Bühne misst (von ganz links bis ganz nach rechts).
Tipp: Verschiebe Scratch so, dass ein bestimmter Körperteil (z.B. die Nasenspitze, Ohrenspitze oder ein Auge) genau auf dem linken Bühnenrand liegt. Nun finde durch Ausprobieren (mit dem 

Befehl) die Anzahl von Schritten, die den gewählten Körperteil genau auf den rechten Rand bringt. Dann ist der Körperteil (und damit auch Scratch) so viele Schritte gelaufen, wie die Bühne breit ist. Wieviele Schritte sind es vom linken zum rechten Bühnenrand?



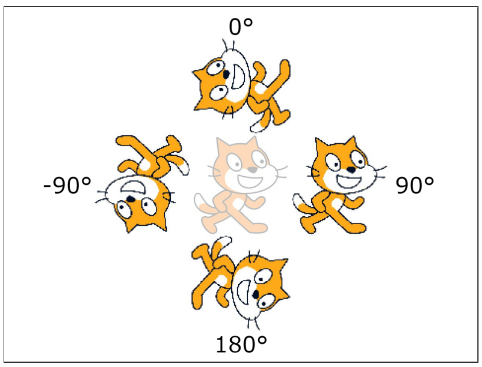
3.2 Die Richtung wählen

Halte den Block 

über den ersten Block und lasse beide zu einem Stapel zusammenschnappen. Wähle im Blockmenü eine Richtung aus, indem du den Pfeil in die gewünschte Richtung ziehst:

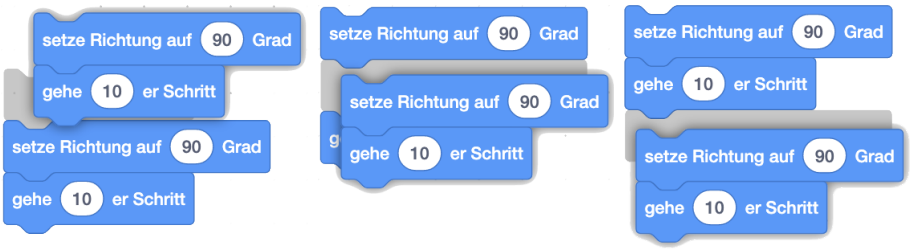
Klicke den Stapel ein paarmal an und beobachte dabei wieder die Bühne!

Scratch zeigt bei jedem Klick zuerst in die ausgewählte Richtung und geht in dieser Richtung dann 10 Schritte geradeaus. Das Bühnenbild unten zeigt Scratch nach 100 Schritten in die jeweilige Richtung.

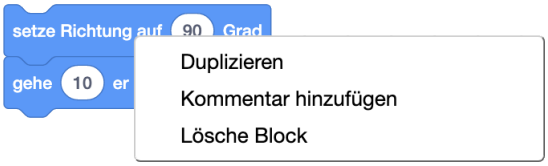


Ein Stapel von Befehlen ist ein *Skript*. Bei jedem Anklicken des Stapels führt dein Objekt alle Befehle des Skripts einmal aus (von oben nach unten).

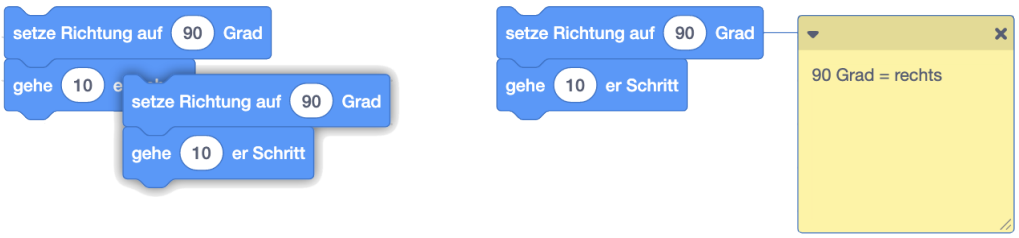
Du kannst Stapel von oben, in der Mitte oder von unten an ein bestehendes Skript anbauen:



Zum Auftrennen der Blöcke ziehst du einfach den unteren Block weg. Du kannst ein Skript auch löschen, indem du es wieder zurück in die Blockpalette ziehst. Einen einzelnen Block kannst du löschen, indem du den Block mit Rechtsklick auswählst und auf "Lösche Block" klickst.



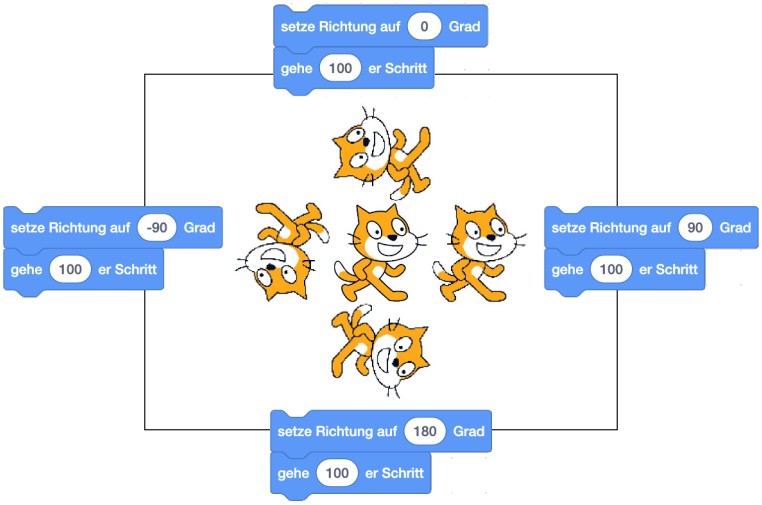
Mit Rechtsklick auf den Block kannst du ausserdem einen Block duplizieren oder einen Kommentar hinzufügen.



Kommentare sind hilfreich, wenn wir anfangen, etwas kompliziertere Programme zu schreiben. Sie helfen vor allem auch, wenn andere deine Programme lesen. Zu viele Kommentare können aber auch schnell unübersichtlich werden. Mit der Zeit wirst du ein Gefühl dafür bekommen, wann Kommentare sinnvoll sind.

Aufgabe 3.3

Dieses Bild zeigt Position und Richtung von Scratch nach einmaliger Ausführung des jeweiligen Skripts:



1. Schreibe zu den folgenden Skripten auf, wieviele Schritte Scratch läuft. Wenn Scratch rückwärts läuft, sollst du diese Schritte auch dazuzählen. Wenn Scratch also zuerst 100 Schritte vorwärts geht und dann 100 Schritte rückwärts, ist Scratch insgesamt 200 Schritte gegangen, auch wenn Scratch am Schluss wieder an der gleichen Position landet.

2. Zeichne zu den folgenden Skripten ein, wo sich Scratch nach Ausführen des jeweiligen Skripts befindet. Du kannst Scratch dabei durch ein Strichmännchen darstellen, aber Standort und Richtung müssen erkennbar sein! Bei jeder Teilaufgabe kannst du davon ausgehen, dass Scratch zu Beginn in der Mitte der Bühne steht und nach rechts (Richtung 90 Grad) schaut.

a)

setze Richtung auf 180 Grad
gehe 100 er Schritt

b)

gehe 100 er Schritt
setze Richtung auf 180 Grad

c)

gehe 100 er Schritt
setze Richtung auf -90 Grad
gehe 200 er Schritt

d)

setze Richtung auf 90 Grad
gehe 100 er Schritt
setze Richtung auf 0 Grad
gehe 100 er Schritt

e)

gehe -100 er Schritt
setze Richtung auf -90 Grad

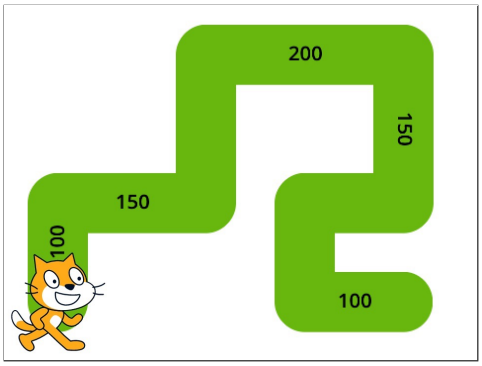
f)

setze Richtung auf 0 Grad
gehe 100 er Schritt
setze Richtung auf 90 Grad
gehe 100 er Schritt
setze Richtung auf -90 Grad
gehe 200 er Schritt
setze Richtung auf 180 Grad
gehe 200 er Schritt

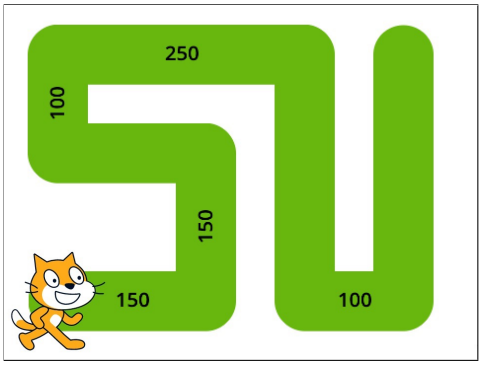
Aufgabe 3.4

In den folgenden Aufgaben soll Scratch dem grünen Weg folgen. Die Wegstücke sind dabei immer mit der Länge (in Anzahl Schritte) angeschrieben. Mit den folgenden Skripten verläuft sich Scratch - kannst du die Skripte korrigieren, damit Scratch richtig dem Weg folgt?

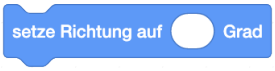
1.



2.



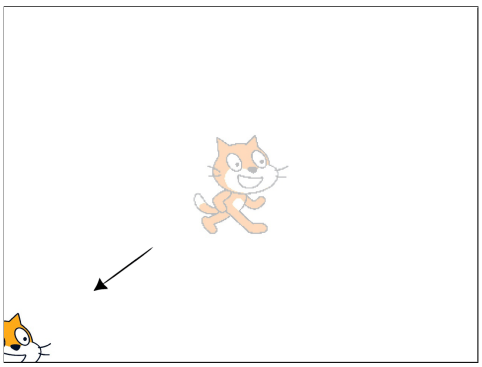
Aufgabe 3.5

Wieviele Schritte misst die Bühne von ganz unten bis ganz oben? Tipp: Drehe Scratch mit dem 

-Befehl und gehe dann ähnlich vor wie in Aufgabe 3.2: Verschiebe Scratch so, dass ein bestimmter Körperteil (z.B. die Nasenspitze, Ohrenspitze oder ein Auge) genau auf dem oberen Bühnenrand liegt und finde dann die Anzahl von Schritten, die den gewählten Körperteil genau auf den unteren Rand bringt.

Aufgabe 3.6

Schreibe ein Skript auf, mit dem du Scratch wie auf dem Bild gezeigt von der Mitte genau in die linke untere Ecke der Bühne bringen kannst!

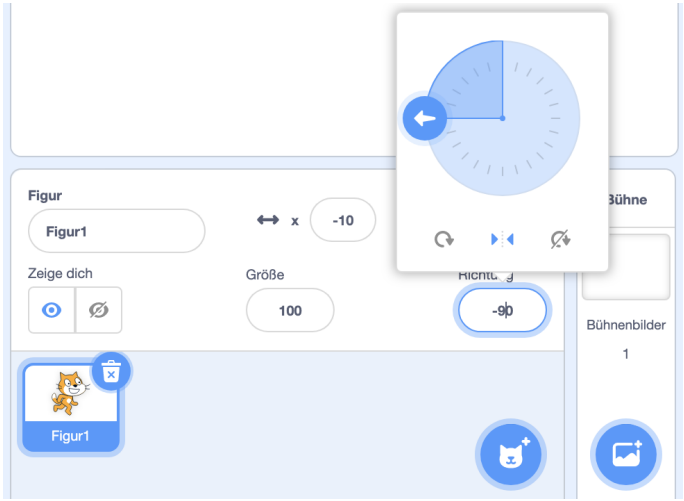


3.3 Ein Objekt animieren

Baue dir im Programmierbereich diese beiden Skripte zusammen:

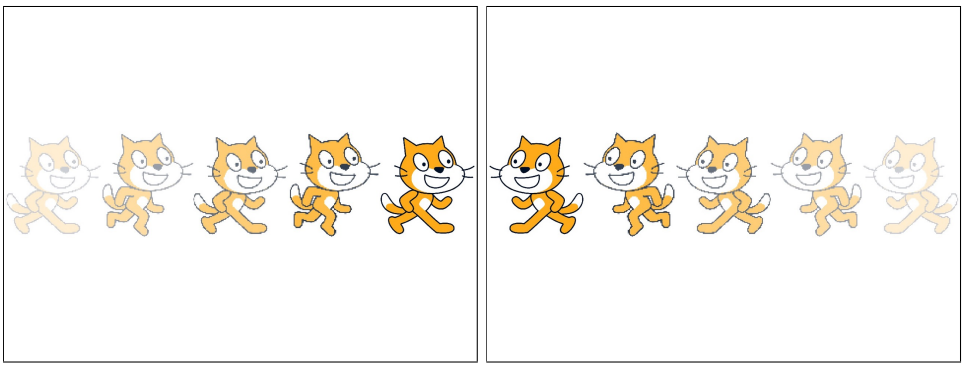
Die violetten Blöcke findest du in der Blockpalette unter „Aussehen“, die blauen bei „Bewegung“.
Setze ausserdem den Drehtyp auf links-rechts, indem du entweder auf den Block 

klickst oder in der Objektliste auf das Symbol ▶◀ klickst:



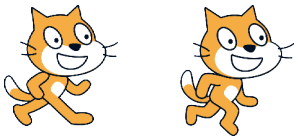
Klicke beide Skripte jeweils ein paarmal an. Was passiert dabei auf der Bühne?

Scratch macht „echte“ Laufbewegungen in die jeweilige Richtung (90 Grad = rechts, -90 Grad = links).

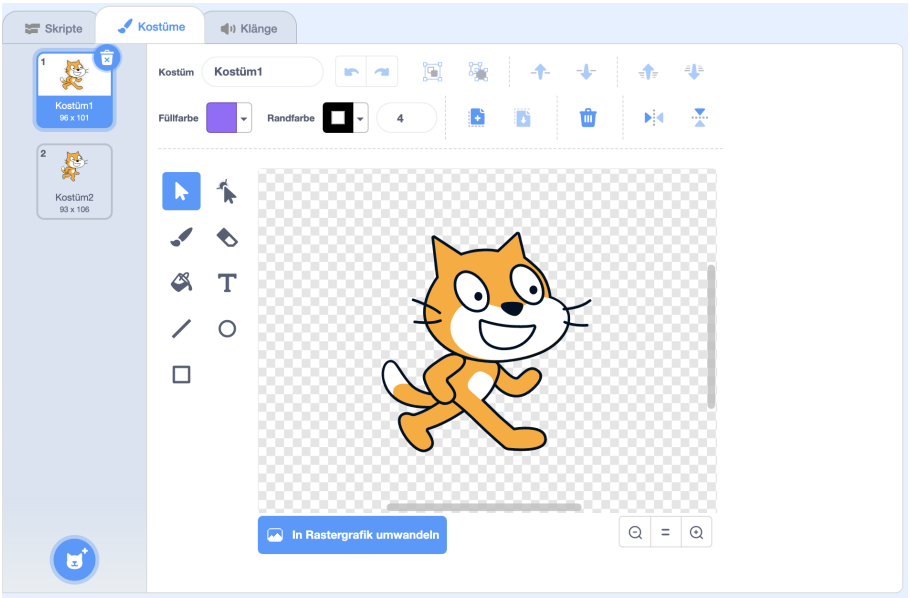


Erreicht wird das durch Kostümwechsel. Scratch hat am Anfang zwei Kostüme zur Auswahl:

Kostüm 1: Kostüm 2:



Du kannst neue Kostüme malen, laden (importieren) oder per Kamera aufnehmen. Du kannst bestehende Kostüme bearbeiten und kopieren.

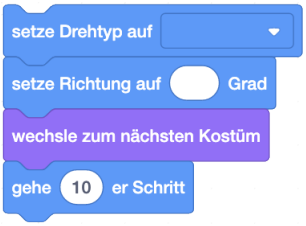


Das Kostüm ist ein *Attribut (Merkmal)* des Objekts. Andere Attribute sind Drehtyp, Standort, Richtung und Verhalten des Malstifts (siehe Kapitel 3.5). Alle Attribute zusammen beschreiben den aktuellen Zustand des Objekts, der beim Speichern des Projekts mitgespeichert wird.

Gib jedem Kostüm einen Namen, der es gut beschreibt, z.B. für die beiden Kostüme von Scratch "stehen" und "laufen". Dann kannst du beim Programmieren später besser entscheiden, welches Kostüm in welcher Situation verwendet werden soll. Eine gute Namensgebung von Attributen und Objekten ist auch wichtig, damit andere (oder du selbst, nach ein paar Wochen) dein Programm schnell verstehen können.

Aufgabe 3.7

Überlege dir, was wohl passiert, wenn du im Skript unten den Drehtyp auf "rundherum" oder "nicht drehen" setzt, bevor du das Skript mit verschiedenen Richtungen ausführst.

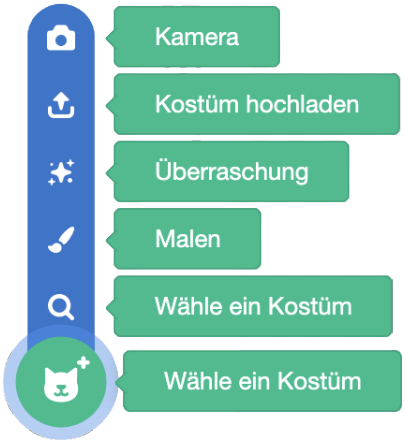


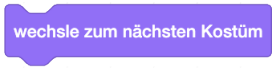
Dann probiere aus, ob deine Vermutung stimmt!

Aufgabe 3.8

Indem du einem Objekt verschiedene Kostüme gibst, kannst du es animieren. Eine Animation besteht aus einer Folge von Bildern, die zusammen einen kleinen Film ergeben.

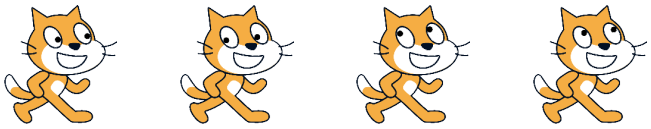
„Verkleide“ Scratch durch Kopieren und Bearbeiten der bestehenden Kostüme, oder durch Malen neuer Kostüme. Kostüme kannst du auch mit der Webcam aufnehmen oder als Bild hochladen. Ausserdem gibt es in der Bibliothek von Scratch schon ganz viele Kostüme, die du ausprobieren kannst.



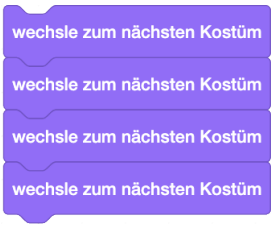
Um die Animation zu sehen, klicke dich mit dem Befehl 

durch die Kostüme. Die zwei Originalkostüme kannst du löschen, wenn du sie nicht mehr brauchst.
Beispiel: In dieser Animation verdreht Scratch die Augen.

Kostüm 1: Kostüm 2: Kostüm 3: Kostüm 4:



Was wären bessere Namen für die Kostüme?
Warum nehmen wir nicht einfach dieses Skript?



Probiere es aus und erkläre deine Beobachtung!

3.4 Auf Tasten reagieren

Baue dir im Programmierbereich diese beiden Skripte zusammen:

Die gelben Blöcke findest du im Menü „Ereignisse“. Die gewünschte Taste wählst du jeweils aus dem Menü des Blocks aus. Drücke dann die Pfeiltasten (← und →) und beobachte, was Scratch auf der Bühne macht.

Wenn du eine der Pfeiltasten gedrückt hältst, läuft Scratch flüssig in die zugehörige Richtung.

Jeder Block von der Form  ist ein *Hut*. Wenn er einem Skript „aufgesetzt“ wurde, führt das Objekt das Skript jedes Mal aus, wenn das im Hut angegebene Ereignis eintrifft. Hüte erlauben dir, Objekte gezielt zu steuern. Wenn das Ereignis erneut auftritt, während das Skript unter dem entsprechenden Hut noch ausgeführt wird, gibt es zwei mögliche Reaktionen: Das Skript wird abgebrochen und sofort von Anfang an ausgeführt oder das Skript wird normal zu Ende ausgeführt und das Ereignis wird ignoriert. Welche dieser Reaktionen eintritt ist vom Hut abhängig.

Aufgabe 3.9

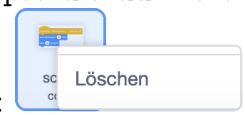
Baue dir zwei Skripte, so dass du Scratch mit den Pfeiltasten ↑ und ↓ flüssig hoch- und runterbewegen kannst. Scratch soll dabei auch wieder in die jeweilige Richtung schauen.

Aufgabe 3.10

Verändere dein in Aufgabe 3.8 erstelltes Animationsprojekt so, dass du die Animation durch wiederholtes Drücken einer Taste ablaufen lassen kannst! Um die Animation zu sehen, kannst du die Taste dann einfach gedrückt halten. Welches

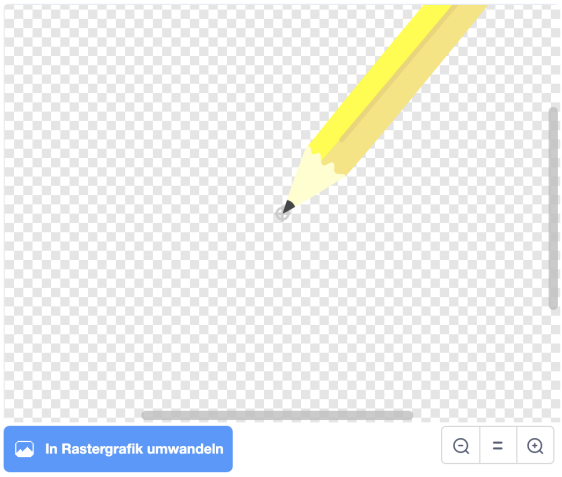
Skript benutzt du dazu? Wieso funktioniert dein Skript, aber das am Ende von Aufgabe 3.8 nicht?

Die Befehle, mit denen du Scratch über die Bühne bewegen kannst sind hilfreich für viele verschiedene Projekte. Damit du sie nicht in jedem Projekt wieder neu zusammenbauen musst, kannst du sie in dein Lager ziehen. Dies geht allerdings nur, wenn du ein Benutzerkonto erstellt hast und nicht die Offline-Version von Scratch benutzt. Klicke unten auf "Lager", um es zu öffnen. Das Skript, das du lagern willst, ziehst du dann nach unten in das Lager.

Um das Skript in einem anderen Projekt zu benutzen, ziehst du es dort einfach wieder heraus in den Programmierbereich. Skripte die du nicht mehr im Lager haben willst, kannst du mit Rechtsklick löschen: 

3.5 Den Malstift benutzen

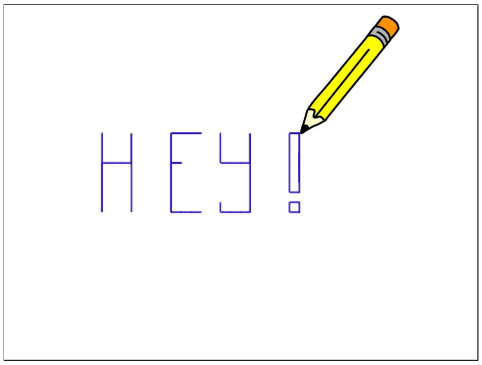
Male für Scratch ein Stift-Kostüm oder wähle ein bereits vorhandenes Stiftkostüm aus, indem du im Kostümmenü auf die entsprechende Taste klickst. Gib diesem neuen Kostüm einen sinnvollen Namen (z.B. "Stift"), damit du die verschiedenen Kostüme leichter unterscheiden kannst. Verschiebe dann das Kostüm so, dass die Stiftspitze auf dem Kostüm-Drehpunkt liegt. Wenn du ein bereits vorhandenes Kostüm verwenden willst, wählst du dazu das ganze Kostüm aus und verschiebst es. Der Kostüm-Drehpunkt ist zu Beginn in der Mitte des Kostüms und daher nicht direkt sichtbar.



Erstelle nun im Programmierbereich die vier Skripte, mit denen du Scratch mit den Pfeiltasten auf der Bühne bewegen kannst. Füge dann diese drei Skripte hinzu:

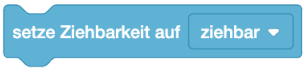
Die Malstift-Blöcke kannst du hinzufügen, wenn du unten in der Blockpalette auf  klickst und dann “Malstift” auswählst. Wechsle durch einen Klick auf das Symbol  oben rechts über der Bühne in den *Präsentationsmodus*. Bewege den Stift mit den Pfeiltasten herum und beobachte, was nach dem Drücken von ’e’, ’a’ und ’l’ passiert.

Du kannst mit dem Stift auf der Bühne ein Bild malen. Durch Drücken der Taste ’l’ werden alle Malspuren wieder gelöscht.



Jedes Objekt hat einen Malstift dabei. Wenn er abgesenkt ist, zeichnet er die Bewegungen des Objekts nach. Es gibt Befehle zum Wählen und Ändern der Stiftdicke, Stiftfarbe und Farbstärke. Der Malstift hinterlässt eine Spur beim gesetzten Drehkreuz des Objekts.

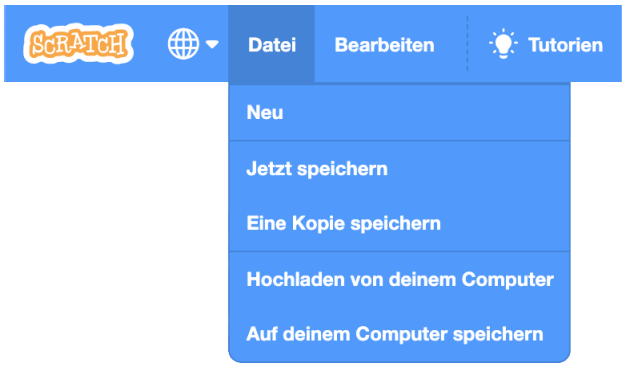
Es gibt verschiedene *Ansichten* deines Programms. Die Ansicht, die wir bis jetzt verwendet haben, heisst *Entwicklungsansicht*. In der Entwicklungsansicht kannst du neue Blöcke zu deinem Programm hinzufügen, sie durch Anklicken ausführen und die Objekte auf der Bühne an die richtige Stelle ziehen. Diese Ansicht dient dazu, dein Programm zu erstellen und einzelne Blöcke oder Skripte zu testen.
Im *Präsentationsmodus* wird dein Programm ausgeführt. Du siehst hier nur die Bühne und die Start- und Stopp-Taste. Es werden nur Skripte ausgeführt, die durch ein Ereignis ausgelöst wurden. Stelle also vorher sicher, dass jedes Skript einen Hut hat! Ausserdem kannst du deine Objekte nicht mehr einfach mit der Maus auf der Bühne umherziehen, wenn du das nicht vorher mit dem



Block definiert hast.

3.6 Ein Projekt erweitern

Speichere eine Kopie von deinem Projekt, um dein Projekt zu erweitern, ohne die jetzige Version zu verlieren.

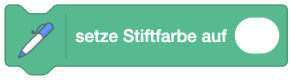


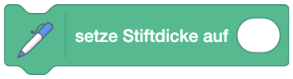
Auf diese Weise kannst du sicherstellen, dass du immer eine funktionierende Version deines Projektes hast, auch wenn sich in der neuen Version irgendwo ein Fehler einschleicht. Denk daran, die Kopie umzubenennen, damit du weisst, was du in welcher Version hinzugefügt hast.

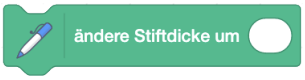
Häufig gibt man in der Benennung einer neuen Version eines Projektes auch die *Versionsnummer* an. Diese zählt mit, wieviele Versionen man bereits vorher erstellt hat. So kann man schnell vergleichen, welches die neuste Version ist. Die Dokumentation und Speicherung von Änderungen nennt man *Versionierung* oder auch *Versionsverwaltung*. Du kannst dieses Konzept auch für andere Dokumente (zum Beispiel Präsentationen oder Aufsätze) verwenden. Vor allem wenn man mit anderen zusammenarbeitet, kann es sonst schwierig sein, den Überblick zu behalten, welches die neuste Version ist.

Aufgabe 3.11

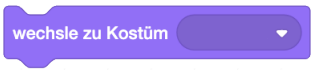
Erweitere das Malprogramm! Vorschläge dafür:

1. Der Stift soll sich beim Zeichnen nicht drehen.
2. Lasse den Malstift auf Tastendruck die Farbe wechseln. Benutze dazu den Befehl 

!
3. Lasse den Malstift auf Tastendruck die Stiftdicke wählen oder ändern. Benutze dazu die Befehle 

und 

!
4. Wenn du malst, kannst du dem Malstift nicht ansehen, ob er gerade abgesenkt oder angehoben ist. Erstelle für den Malstift ein weiteres Kostüm, das

du immer dann benutzt, wenn er angehoben ist (z.B. einen durchgestrichenen Stift). Um jeweils das richtige Kostüm zu wählen, benutze den Befehl 

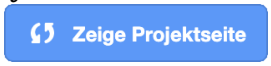
.

5. Programmiere einen „Radiergummi“: Auf Tastendruck soll der Malstift ein Radiergummi-Kostüm anziehen. Dann soll er radieren, d.h. mit Stiftfarbe weiss malen, wobei die Stiftdicke etwas grösser gewählt ist. Die passende Stiftdicke kannst du z.B. durch Ausprobieren herausbekommen.

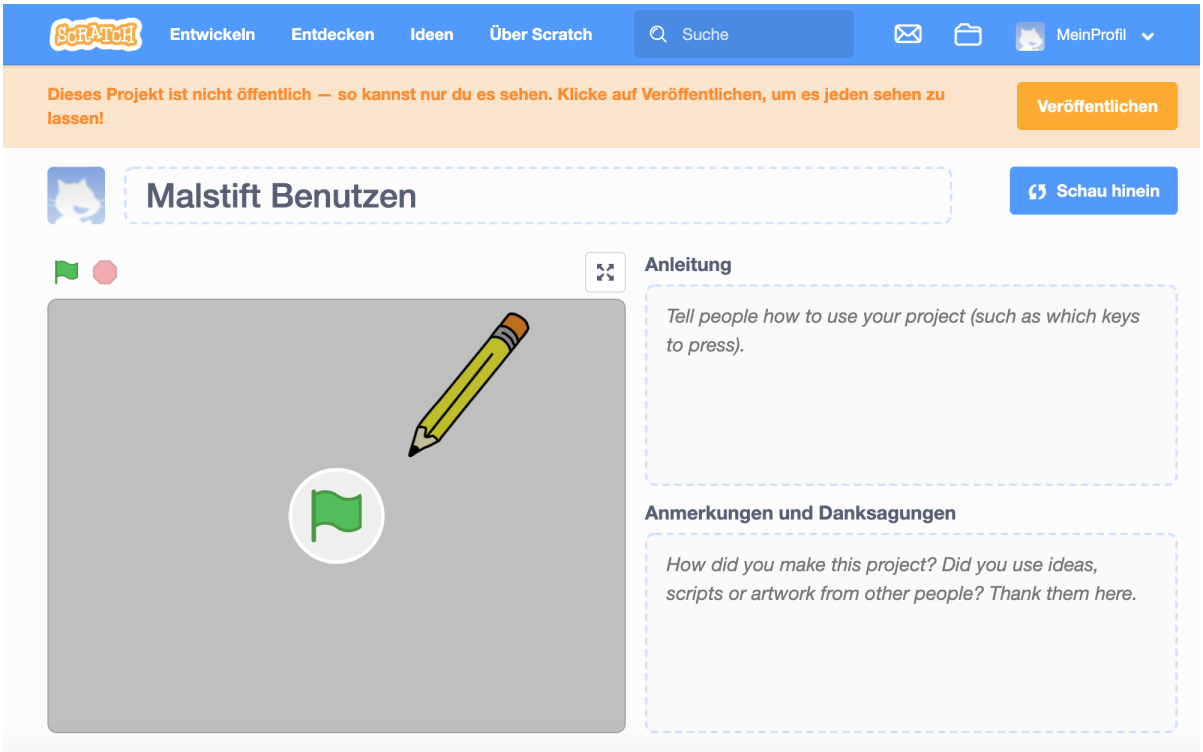
3.7 Ein Projekt veröffentlichen

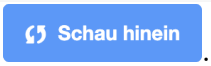
Wenn du mit einem Programm (zum Beispiel dem Malprogramm aus dem vorherigen Kapitel) zufrieden bist, kannst du es veröffentlichen. Zum Veröffentlichen brauchst du ein Benutzerkonto. Erst wenn du dein Projekt veröffentlichst, können es andere anschauen. Frage aber vor dem Veröffentlichen deine Eltern oder deine Lehrperson, ob du das darfst.

Räume dein Projekt aber vor dem Veröffentlichen etwas auf! Es sollten keine unbenutzten Blöcke mehr auf dem Programmierbereich sein.

Wenn du dein Projekt mit anderen teilen willst, musst du ausserdem eine kurze Beschreibung hinzufügen, damit man weiss, wie man dein Projekt steuern kann (z.B. mit den Pfeiltasten) und was das Ziel des Projekts ist (z.B. ein Malprogramm). Ausserdem kannst du hier angeben, wenn dir jemand bei deinem Projekt geholfen hat, und dich dafür bedanken. Klicke dazu auf 

und erstelle eine Beschreibung. Du siehst hier auch eine Vorschau, wie dein Projekt für andere aussehen würde, wenn du es veröffentlichst.



Um wieder zur Entwicklungsansicht zurück zu wechseln, klicke auf 

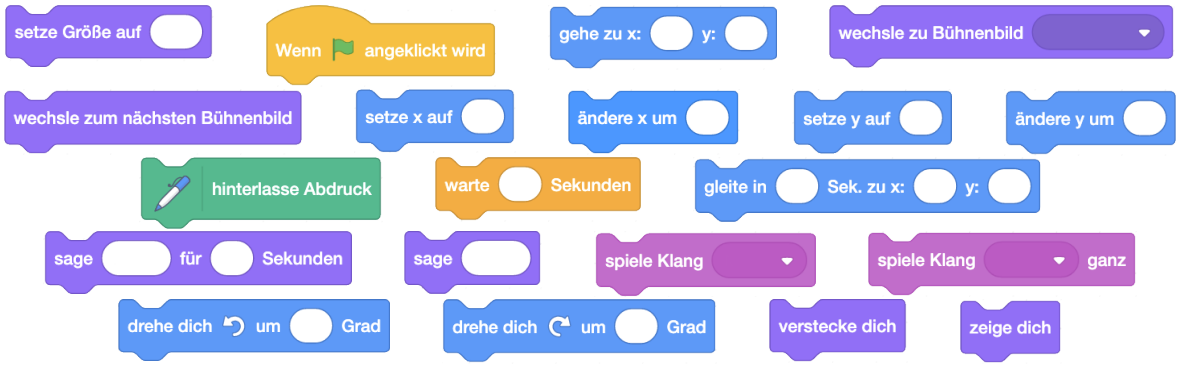
.

Um den Überblick zu behalten, welches die neusten Versionen von deinen Projekten sind, kannst du ein *Studio* erstellen. Ein Studio ist eine Sammlung von Projekten, die zum Beispiel ein gemeinsames Thema haben. Dort fügst du die jeweils neuste Version eines Projektes hinzu und löschst die vorherige Version aus dem Studio. Das Projekt wird dadurch nicht gelöscht. Um ein Studio zu erstellen, klicke auf der "Meine Sachen" Seite auf die Taste "+ Neues Studio" oben rechts. Gib deinem neuen Studio einen beschreibenden Namen. Du kannst ausserdem in der linken Spalte noch genauer beschreiben, worum es in diesem Studio geht. Nun kannst du deine veröffentlichten Projekte zum Studio hinzufügen. Du kannst natürlich auch ein Studio erstellen für Projekte die dich inspirieren oder die du hilfreich findest.

Kapitel 4

Orientierung

Hier lernst du, wie du dich auf der Bühne orientieren kannst: Du bringst einem Objekt bei, seine Position und Richtung zu setzen und zu ändern. Damit kannst du per Tastendruck leicht geometrische Muster malen.
Die Blöcke, die du dabei neu kennenlernst:



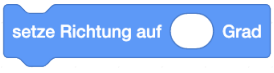
Die Begriffe, die hier neu eingeführt werden:

- Aufräumskript
- Punkt, x- und y-Koordinaten, Richtung, Drehwinkel (Geometrie)
- Nebenläufigkeit

4.1 Die Position wählen

Mache Scratch mit dem Befehl 

ganz klein. Ziehe nun Scratch über die Bühne und beobachte dabei die Werte für x und y in der Objektliste. Versuche, möglichst viel über diese Werte herauszufinden. Du darfst dazu auch die Befehle 

und 

(mit verschiedenen Pa-

rametern) benutzen. Schreibe deine Beobachtungen auf und vergleiche mit deinen Klassenmitgliedern!

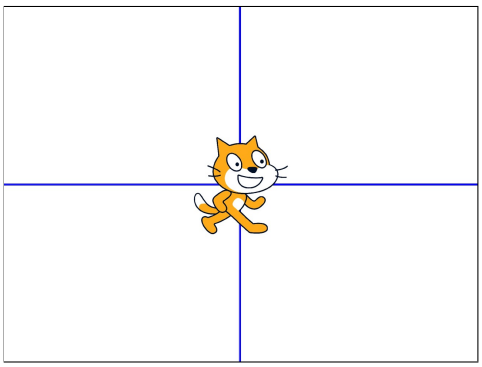
Minimaler Wert von x	-240
Maximaler Wert von x	240
Minimaler Wert von y	-180
Maximaler Wert von y	180
Werte in der Mitte der Bühne	x=0, y=0
Setze Richtung auf	Keine Auswirkung auf x und y
10er Schritt in Richtung 90 Grad (rechts)	x wird um 10 grösser
10er Schritt in Richtung -90 Grad (links)	x wird um 10 kleiner
10er Schritt in Richtung 0 Grad (oben)	y wird um 10 grösser
10er Schritt in Richtung 180 Grad (unten)	y wird um 10 kleiner

In der unteren Hälfte der Bühne ist $y < 0$, in der oberen Hälfte ist $y > 0$.
Je weiter oben Scratch sich befindet, desto grösser ist y.
In der linken Hälfte der Bühne ist $x < 0$, in der rechten Hälfte ist $x > 0$.
Je weiter rechts Scratch sich befindet, desto grösser ist x.

Ein *Punkt* wird mit **x**- und **y**-Koordinaten beschrieben (**x**: links $\rightarrow$ rechts, **y**: unten $\rightarrow$ oben). **Beispiel**: Punkt (-54, 106) ist 54 Schritte links und 106 Schritte oberhalb der Mitte.

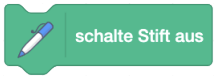
Was denkst du passiert, wenn man für Scratch das folgende Skript erstellt und dann auf die grüne Flagge über der Bühne klickt? Zeichne deine Vermutung auf und probiere dann aus, ob das stimmt!

Scratch malt das Koordinatenkreuz seiner Welt (der Bühne) und geht danach wieder zum Punkt (0,0), der Mitte der Bühne.

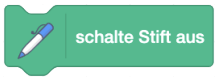


Aufgabe 4.1

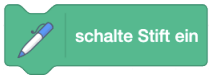
Welches Bild malt Scratch, wenn du aus diesem Skript den jeweils angegebenen Block entfernst?

a) Den ersten 

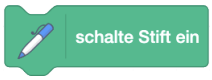
- Block

b) Den zweiten 

- Block

c) Den ersten 

- Block

d) Den zweiten 

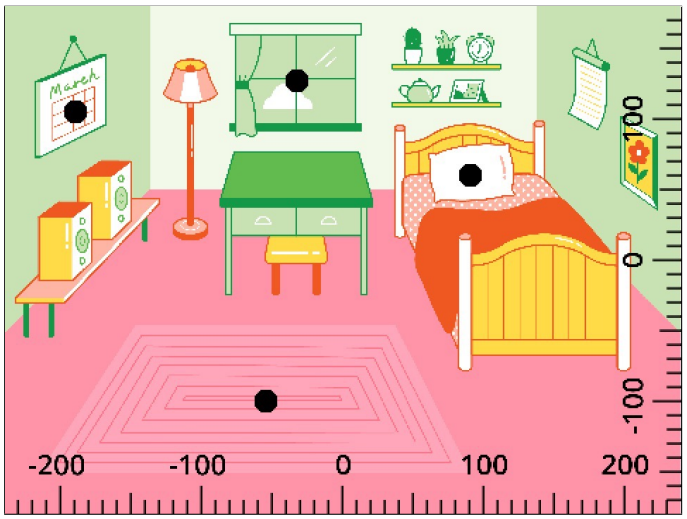
- Block

Male deine Lösung zuerst auf und probiere dann aus, ob sie stimmt! Vergiss nicht, vor jedem neuen Versuch den Stift anzuheben und die Malspuren wegzuwischen!

Aufgabe 4.2

Lade den Hintergrund “Bedroom 1”. Welche x- und y-Koordinaten haben die eingezeichneten Punkte ungefähr? Trage die Koordinaten zuerst ein und überprüfe sie dann, indem du Scratch wieder ganz klein machst und an die Stelle ziehst.

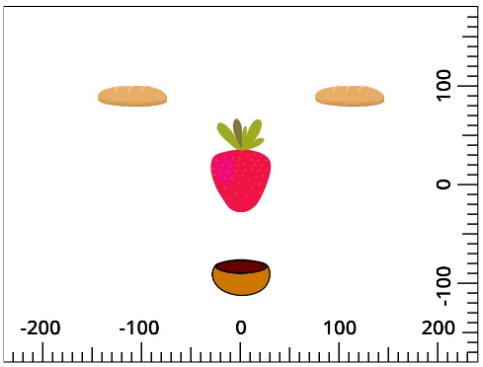
Kalender:
(,)
(,)
Fenster:
(,)
(,)
Kissen:
(,)
(,)
Teppich:
(,)
(,)



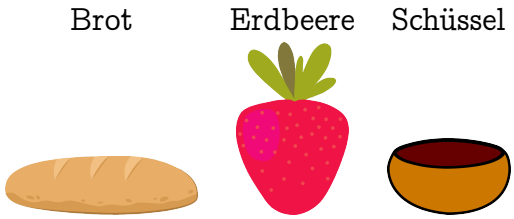
Aufgabe 4.3

In den folgenden Aufgaben soll Scratch diese “Gesichter” zeichnen. Leider stimmt aber das Skript so nicht. Wo liegt der Fehler?

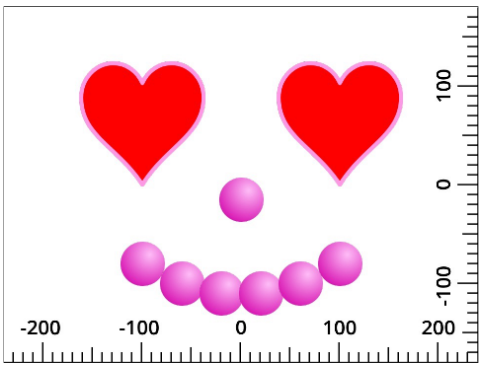
1.



Verwendete Kostüme:

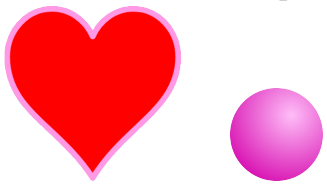


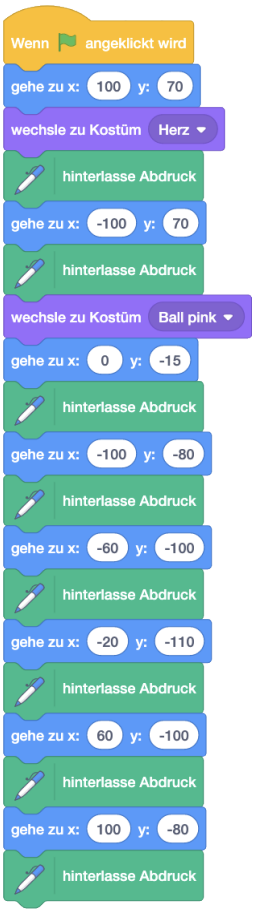
2.



Verwendete Kostüme:

Herz Ball pink

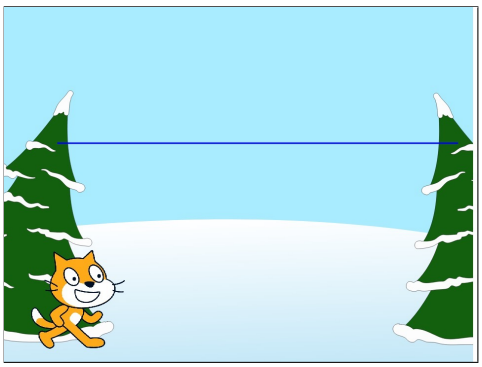




Zusatzaufgabe: Denk dir selber ein “Gesicht” aus und programmiere es!

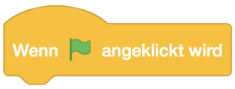
Aufgabe 4.4

Lade den Hintergrund „Winter“ und schreibe ein Skript, mit dem Scratch die blaue „Wäscheleine“ malt und sich dann vor den linken Baum stellt! Schreibe dein Skript auf! Achtung: Die Wäscheleine sollte ganz gerade sein!

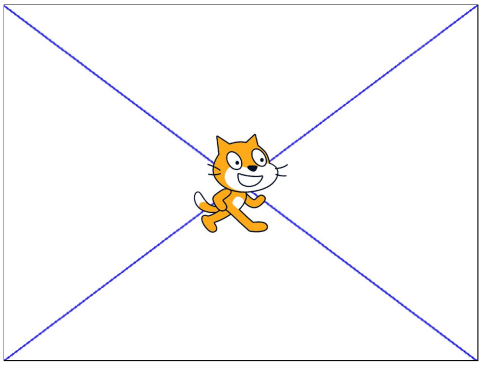


Aufgabe 4.5

Schreibe ein Skript, mit dem Scratch dieses Bühnenbild malen kann! Scratch soll am Schluss wieder genau in der Mitte stehen. Beginne mit dem

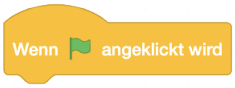


- Block.

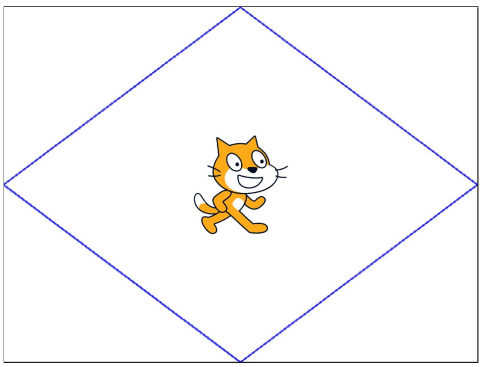


Aufgabe 4.6

Schreibe ein Skript, mit dem Scratch dieses Bühnenbild malen kann! Scratch soll am Schluss wieder genau in der Mitte stehen. Beginne mit dem

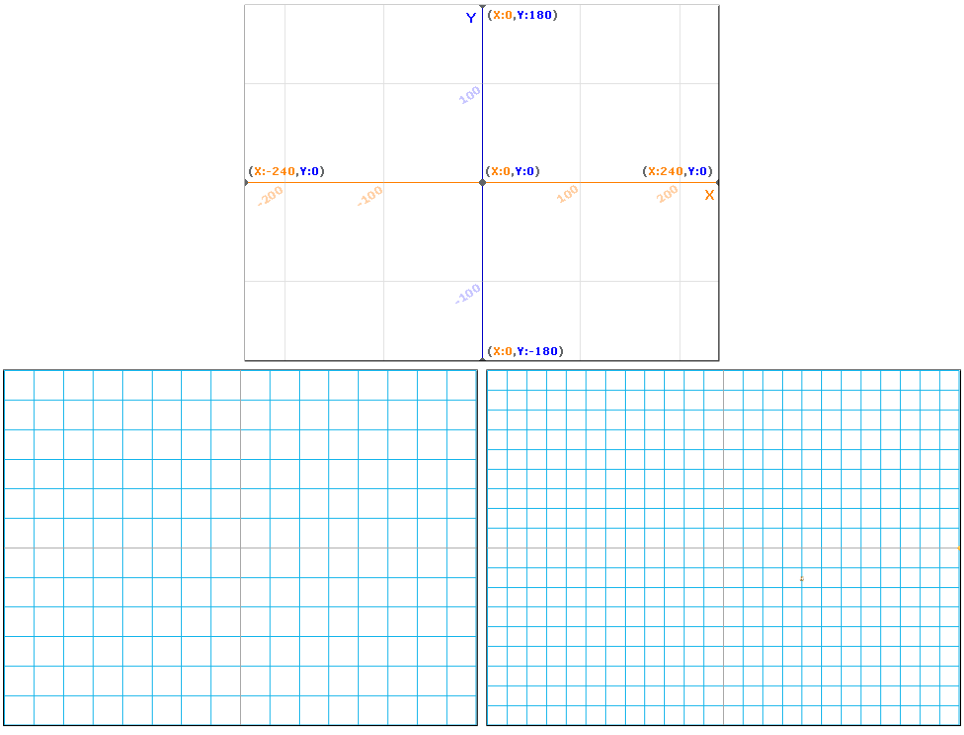


- Block.



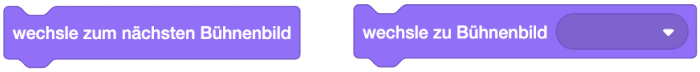
4.2 Den Hintergrund wechseln

Wenn du am Anfang noch Schwierigkeiten damit hast, die Koordinaten eines Punktes zu bestimmen, dann kannst du als Hilfe den Hintergrund der Bühne zu einem Koordinatensystem wechseln. Klicke dazu auf "Bühnenbild wählen" unten rechts und wähle den Hintergrund "Xy-grid", "Xy-grid-30px" oder "Xy-grid-20px" aus:

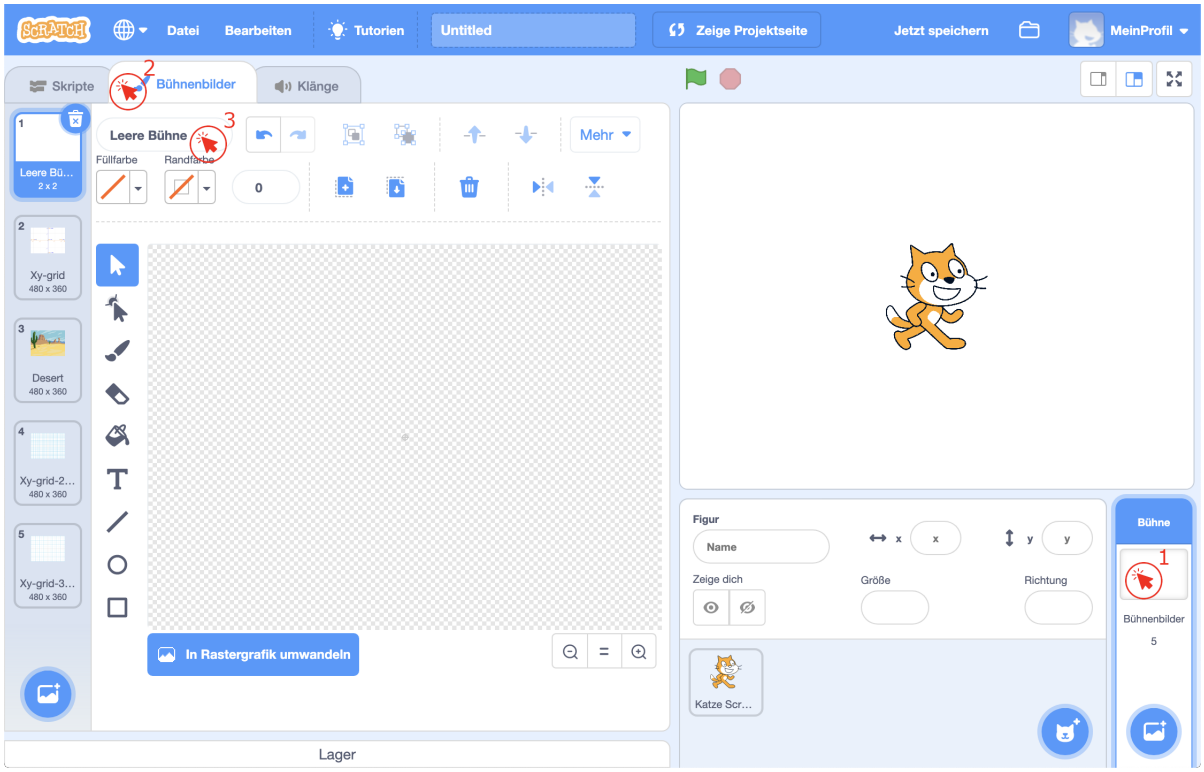


Der erste Hintergrund zeigt dir die Koordinaten in 100er Schritten und die x- und y-Achse sind angeschrieben. Der zweite Hintergrund ist in 30er Schritten gekachelt und der letzte in 20er Schritten.

Diese Hintergründe sind zwar nützlich, aber wenn wir zum Beispiel etwas zeichnen wollen, sieht das auf einem leeren Hintergrund viel schöner aus. Um zwischen den Hintergründen zu wechseln, kannst du diese zwei Befehle verwenden:



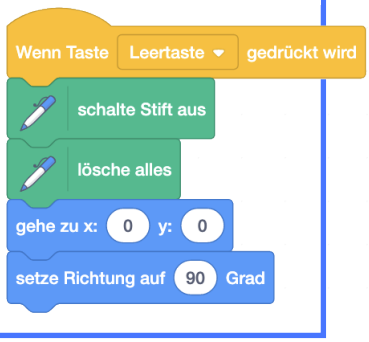
Auch bei Bühnenbildern ist es wieder wichtig, einen guten Namen zu wählen, damit man beim Wechseln genau weiss, welches Bild man auswählen soll. Hier ist zum Beispiel "Bühnenbild1" nicht sehr aussagekräftig - besser wäre vielleicht "Leere Bühne" oder "Alles Weiss". Den Namen der Bühnenbilder kannst du wechseln, indem du zuerst die Bühne als Objekt auswählst und dann auf das Tab "Bühnenbilder" klickst:



Um nach dem Umbenennen des Bühnenbilds wieder zurück in den Programmierbereich zu wechseln, klick oben links auf das Tab "Skripte". Wenn du bereits vorher etwas für Scratch programmiert hast, erschrickst du vielleicht - alle Skripte sind verschwunden! Keine Angst, die Skripte sind noch da. Der Programmierbereich zeigt momentan die Skripte für die Bühne an! Die Bühne ist ein spezielles Objekt. Sie kann sich nicht bewegen und auch nicht die Richtung wechseln. Man kann sie aber trotzdem programmieren. Mit einem Klick auf ein Objekt in der Objektliste kommst du wieder zum Programmierbereich für dieses Objekt.

4.3 Aufräumen

Lasse Scratch auf der Bühne herumlaufen oder ein Bild malen. Erstelle dann dieses Skript und drücke die Leertaste! Was passiert?



Alle Zeichnungen werden gelöscht und Scratch bewegt sich wieder in die Mitte der Bühne und schaut nach rechts, wie im Ausgangszustand eines neuen Projekts. So kannst du von vorne anfangen, ohne alle Skripte zu verlieren.

Ein Skript, das ein Objekt und/oder die Bühne in einen ganz bestimmten Zustand bringt, heisst *Aufräumskript*. Aufräumskripte sorgen dafür, dass dein Projekt (Geschichte, Animation, Spiel,...) immer „richtig“ anfängt oder weitergeht. Je nach Projekt kann dieser Ausgangszustand, und somit auch das Aufräumskript, ganz unterschiedlich aussehen. Weil viele Projekte mit dem

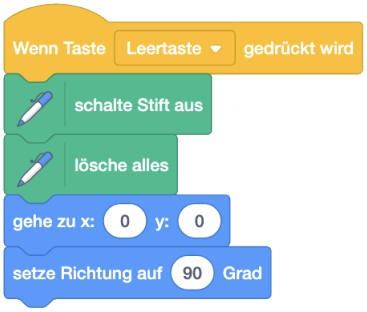
Block beginnen, sind auch Aufräumskripte oft unter diesem Block zu finden.

Du kannst auch den Programmierbereich automatisch aufräumen lassen, damit du einen besseren Überblick über deine Skripte hast. Klicke mit der rechten Maustaste in den Programmierbereich, und wähle dann “Blöcke aufräumen” aus. Aufgepasst: Durch das Aufräumen kann es passieren, dass deine Skripte nicht mehr in der gleichen Reihenfolge wie zuvor im Programmierbereich sind.
Mit der Tastenkombination Command (⌘) / Control (Ctrl) + Z kannst du deine letzten Schritte aber wieder rückgängig machen. Mit Shift (⇧) + Command (⌘) / Control (Ctrl) + Z kannst du den “Rückgängig” Befehl umkehren. Welche Taste (Command (⌘) oder Control (Ctrl) du benutzen musst hängt davon ab, was du für einen Computer hast.

Aufgabe 4.7

1. Ist die Reihenfolge der Befehle in diesem Aufräumskript egal? Wenn nicht, worauf musst du achten?

2. Das gezeigte Aufräumskript ist nicht vollständig im Hinblick auf die in Unterkapitel 3.3 behandelten Attribute (Merkmale eines Objekts). Welches Merkmal wird nicht beachtet? Behebe das Problem durch Ergänzen des Skripts.



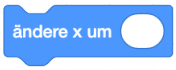
4.4 Die Position ändern

Probiere die folgenden Befehle mit verschiedenen Parametern aus. Welchen Effekt haben sie auf die x- und y-Position von Scratch? Ist das Resultat immer das gleiche?

Mit diesen Befehlen verschiebst du ein Objekt ohne Richtungsänderung *horizontal* (setze und ändere x) oder *vertikal* (setze und ändere y).

Befehl	Effekt auf Position
setze x auf Zahl	(x, y) → (Zahl, y)
ändere x um Zahl	(x, y) → (x + Zahl, y)
setze y auf Zahl	(x, y) → (x, Zahl)
ändere y um Zahl	(x, y) → (x, y + Zahl)
gehe zu x: Zahl1 y: Zahl2	(x, y) → (Zahl1, Zahl2)

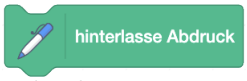
Aufgabe 4.8

Was ist der Unterschied zwischen dem 

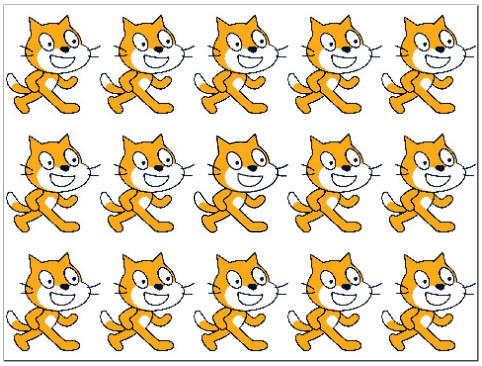
Befehl und dem 

Befehl? Nenne für beide Befehle ein Beispiel, wo du diesen Block verwenden würdest.

Aufgabe 4.9

Erstelle das "Stempelbild" unten, das aus 3 Zeilen und 5 Kolonnen besteht. Programmiere die Pfeiltasten so, dass Scratch an der aktuellen Position einen Abdruck hinterlässt (der Befehl dafür ist 

) und dann die Kolonne (← und →) oder die Zeile (↑ und ↓) wechselt.



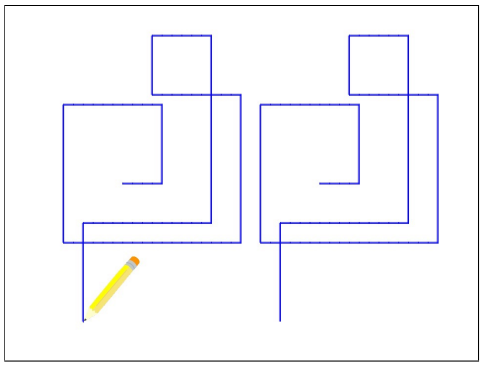
Beachte, dass die Stempelbilder von Scratch alle in die gleiche Richtung schauen!

Programmieraufgaben brauchen Planung, damit du sie gut lösen kannst. Zuerst musst du dir aufschreiben, was das Programm eigentlich können soll. Dann zer-

teilst du diese Aufgabe in kleinere Schritte, für die du dann jeweils noch die passenden Blöcke heraussuchen musst. Probiere es mit der nächsten Aufgabe aus!

Aufgabe 4.10

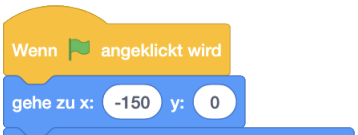
Wir bauen uns einen Kopierer: Erstelle das Programm rechts, mit dem du per Pfeiltasten die linke Bühnenhälfte bemalen kannst. Ergänze nun jedes der vier Pfeiltastenskripte so, dass beim Malen automatisch eine Kopie deines Bildes in der rechten Bühnenhälfte entsteht, wie unten gezeigt.



Tipp: Die Ergänzung ist für alle vier Skripte gleich! Schreibe deinen Plan und die passende Skriptergänzung auf! Wenn du Schwierigkeiten hast, schaue dir zuerst den Plan der Lösung an und versuche dann, die passenden Blöcke dazu zu finden.

4.5 Mehrere Objekte gleichzeitig steuern

Gib Scratch ein Fisch-Kostüm und erstelle dieses Skript:

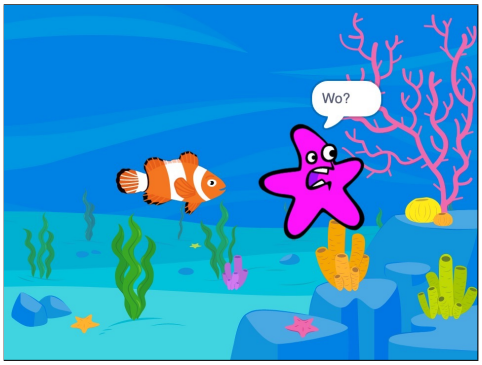


Füge nun den Seestern als Objekt hinzu, indem du in der Objektliste auf die "Figur wählen"-Taste klickst.

Benenne die Figur dann sinnvoll und erstelle im Programmierbereich des Seesterns dieses Skript:

Gib ausserdem den beiden Kostümen des Seesterns bessere Namen. Was passiert, wenn du auf die grüne Flagge klickst?

Beide Objekte bewegen sich auf die Mitte zu. Der Fisch sagt "Hi!" (wird ausgesprochen wie "Hai"), woraufhin der Seestern sich umdreht und erschreckt "Wo?" fragt.



In Scratch können mehrere Objekte gleichzeitig Skripte ausführen. Mit Befehlen mit Zeitangabe wie zum Beispiel dem 

Block oder dem 

Block kannst du die Aktionen dieser Objekte aufeinander abstimmen, damit sie zum Beispiel miteinander reden können. Wenn Skripte gleichzeitig ausgeführt werden, spricht man von *Nebenläufigkeit*. Auch Programme mit nur einem Objekt können nebenläufig sein, wenn zum Beispiel zwei Skripte den gleichen Hut haben. Solche Programme sind nicht immer einfach zu schreiben, denn es ist nicht immer klar, was passiert, wenn zwei Skripte gleichzeitig laufen.

Wenn mehrere Skripte gleichzeitig ausgeführt werden, ist es oft schwierig, den Fehler in einem fehlerhaften Programm zu finden. Nutze das Lager, um einzelne Skripte zwischenzeitlich aus deinem Programm zu entfernen und dein Programm zu testen:

1. Öffne das Lager, indem du es anklickst.
2. Ziehe das Skript ins Lager.
3. Lösche das Skript aus dem Programmierbereich, indem du es in die Blockpalette ziehst.
4. Teste dein Programm: Funktionieren alle anderen Skripte wie erwartet oder besteht der Fehler immer noch? Wenn die übrigen Skripte funktionieren, kannst du das Skript wieder zurück in den Programmierbereich ziehen und genauer untersuchen. Ansonsten überprüfst du die anderen Skripte auf die gleiche Weise.

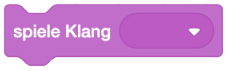
5. Lösche dein Skript wieder aus dem Lager.

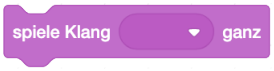
Aufgabe 4.11

1. Erkläre genau den Unterschied zwischen dem Befehl 

und dem Befehl 

!

2. Erkläre genau den Unterschied zwischen dem Befehl 

und dem Befehl 

!

Aufgabe 4.12

Denk dir eine kurze Geschichte oder einen Witz aus, und programmiere dazu eine Animation mit den neuen Befehlen! Wenn dir gerade nichts einfällt kannst du auch zwei zufällige Figuren und einen zufälligen Hintergrund auswählen lassen (klicke auf "Überraschung"). Überlege dir dann dazu, wieso diese Figuren dort sind und was sie wohl zueinander sagen würden.

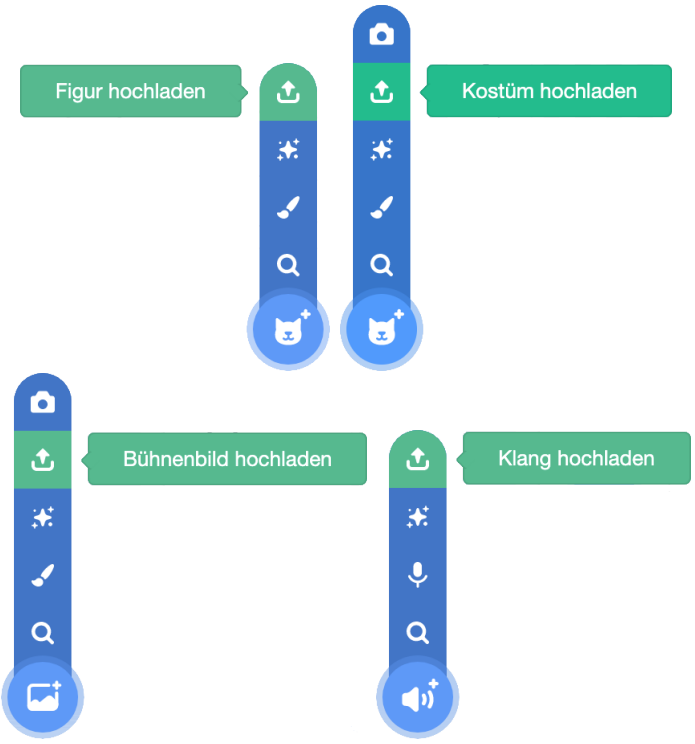
Aufgabe 4.13

Löse Aufgabe 4.10, indem du zwei Objekte verwendest: Ein Objekt zeichnet die Original-Malspuren, das andere die Kopien. Das zweite Objekt soll dabei nicht sichtbar sein.

Tipp: Mit Rechtsklick auf ein Objekt kannst du es duplizieren. Dadurch erhältst du ein zweites Objekt mit den gleichen Skripten und Kostümen.

Mit Rechtsklick auf ein Objekt kannst du es auch exportieren. Dadurch speicherst du das Objekt mit all seinen Kostümen, Klängen und Skripten auf dem Computer. So kannst du es in einem anderen Projekt wiederverwenden. Wenn du nur

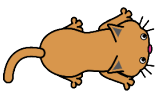
bestimmte Kostüme oder Klänge des Objekts brauchst, kannst du diese auch einzeln mit einem Rechtsklick auf das Kostüm oder den Klang exportieren. Auch Bühnenbilder kannst du so herunterladen. Um ein heruntergeladenes Objekt, Kostüm, Bühnenbild oder einen Klang zu einem Projekt hinzuzufügen, wähle im jeweiligen Menü "Figur/Kostüm/Bühnenbild/Klang hochladen":



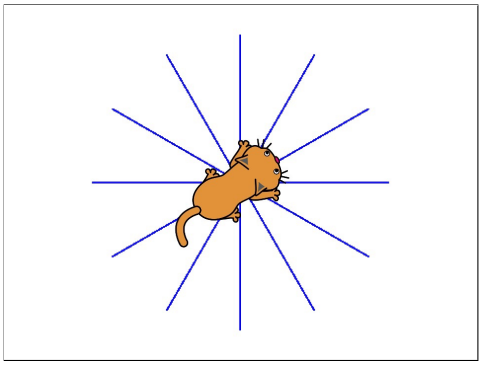
Achte auf die verschiedenen Dateiformate, die du hier verwenden kannst - welche kommen dir bekannt vor und welche sind neu?

4.6 Die Richtung ändern

Gib Scratch das Kostüm "Cat 2" und benenne es sinnvoll. Erstelle dann das Aufräumskript aus Kapitel 4.3 und die drei Tastenskripte unten. Was passiert beim Drücken der Pfeiltasten →, ← und der Leertaste?

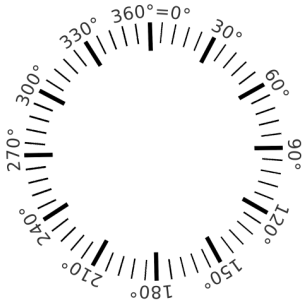


Bei den Pfeiltasten dreht sich Scratch ein Stück nach rechts oder nach links. Die Leertaste „feuert“ eine Linie in die aktuelle Richtung ab.



Drehwinkel und Richtungen werden in *Grad* (°) gemessen. 360° ist eine volle Drehung. Man dreht *im* (↻) oder *gegen* (↺) den Uhrzeigersinn.

Richtungen:

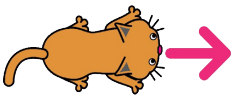


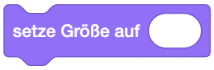
Drehwinkel:

=

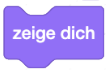
“+ 5 Minuten”

Das Kostüm "Cat 2" (rechts) ist hilfreich, wenn man genau sehen will, in welche Richtung das Objekt gerade zeigt, da die Nasenspitze immer in die gewählte Richtung zeigt. In den Lösungen zu den folgenden Übungen wirst du daher häufig dieses Kostüm sehen.



Manchmal ist das Objekt zudem mit dem Befehl 

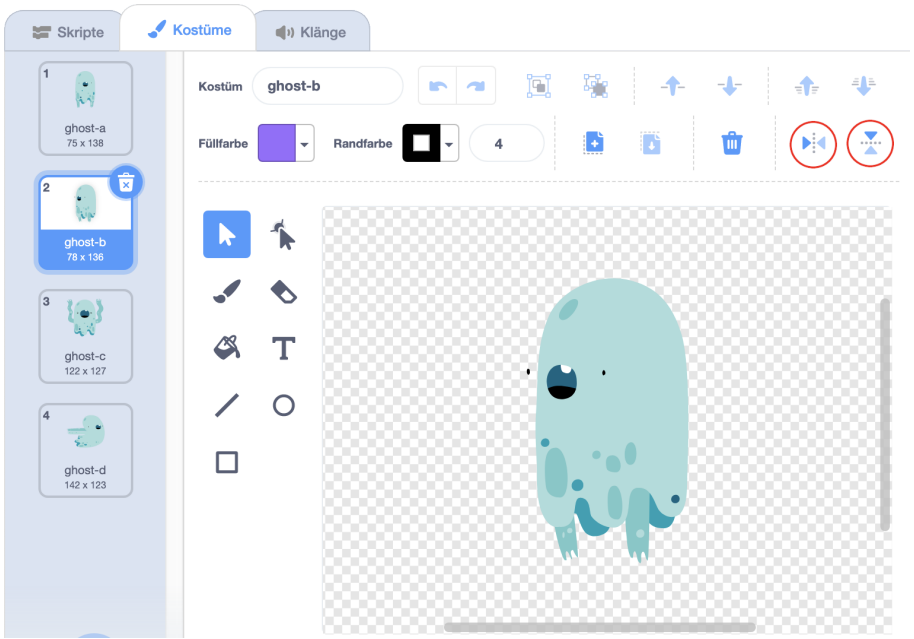
verkleinert worden oder es versteckt sich mit 

, damit man das gezeichnete Bild besser sieht. Mit dem Befehl 

kannst du das Objekt wieder sichtbar machen.

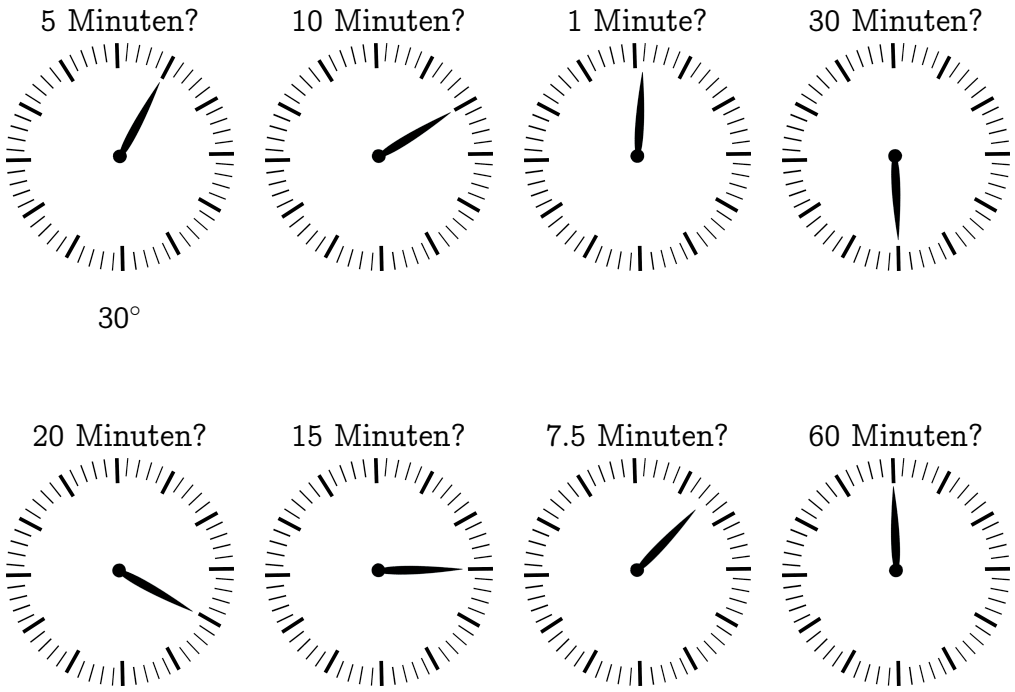
Aufgepasst: Andere Kostüme, wie zum Beispiel das zweite Kostüm der Figur "Ghost" (unten) schauen in eine andere Richtung, auch wenn die Richtung auf 90 Grad gesetzt ist. Die Richtung eines Objekts gibt nur an, in welche Richtung es mit dem 

Befehl läuft, und nicht, wo es hinschaut! Man kann Kostüme aber auch horizontal und vertikal spiegeln:



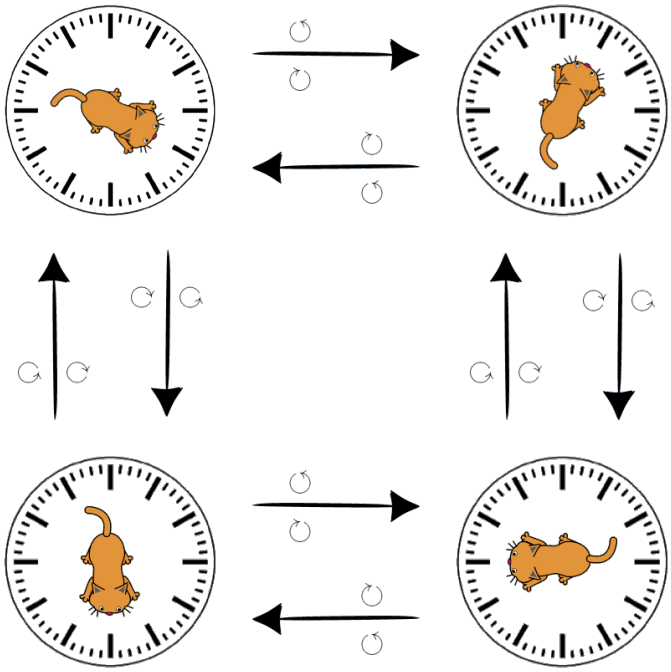
Aufgabe 4.14

Um welchen Winkel dreht sich der Minutenzeiger der Uhr in...



Aufgabe 4.15

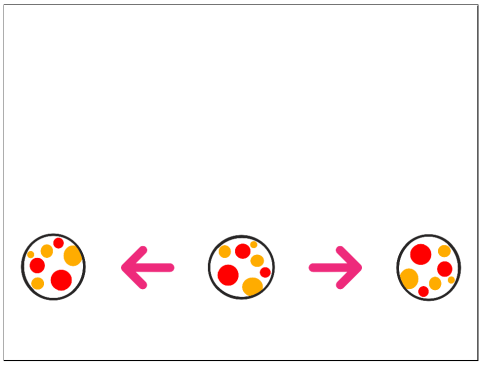
Trage jeweils den Winkel ein, um welchen Scratch sich drehen muss, um von einem Bild zum nächsten zu kommen. Schreibe dabei jeweils den Drehwinkel für die Drehung im Uhrzeigersinn (↻) und gegen den Uhrzeigersinn (↺) auf.



Erkennst du einen Zusammenhang zwischen den Winkeln?

Aufgabe 4.16

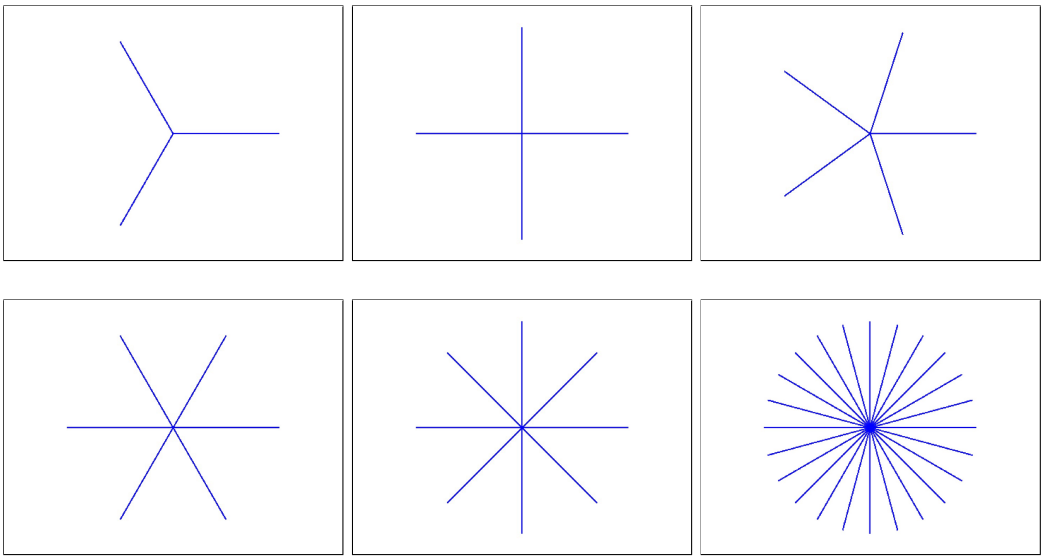
Lade für Scratch das Beachball-Kostüm und erstelle zwei Skripte, mit denen du den Ball mit den Pfeiltasten ← und → auf der Bühne nach links und nach rechts rollen kannst. Der Ball muss sich dabei drehen!



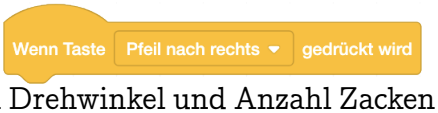
Aufgabe 4.17

Zeichne die folgenden Sterne! Tipp: Du kannst alle Sterne mit dem gleichen "Grundskript" malen und musst jeweils nur den Drehwinkel verändern. Welches

Grundskript und welche Drehwinkel nimmst du?



Beginne das Skript mit diesem Hut:

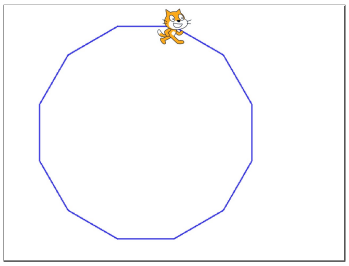


Gibt es einen Zusammenhang zwischen Drehwinkel und Anzahl Zacken des Sterns?

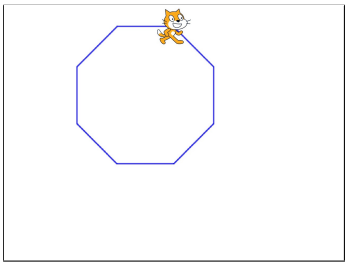
4.7 Vielecke malen

Gib Scratch diese zwei Skripte und probiere im unteren Skript die Winkel 30°, 45°, 60°, 72°, 90°, 120°. Wie oft musst du jeweils die Taste „Pfeil nach rechts“ drücken, bis Scratch wieder am Ausgangspunkt ist, und wie sehen die Bilder aus?

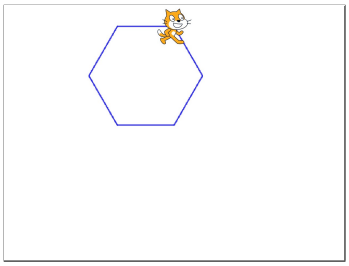
Scratch malt Vielecke. Für jede Ecke musst du einmal → drücken.



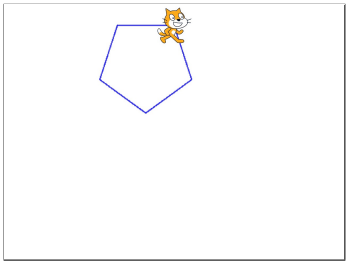
30° / Zwölfeck



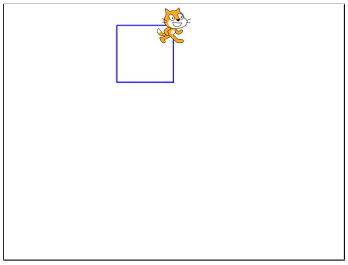
45° / Achteck



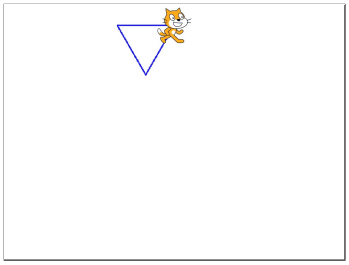
60° / Sechseck



72° / Fünfeck

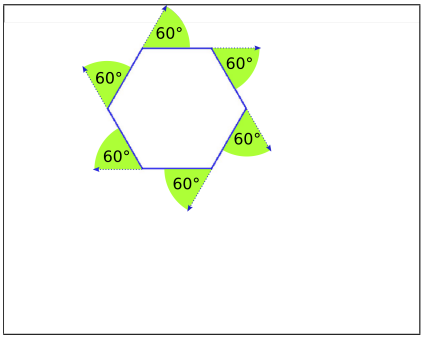


90° / Viereck



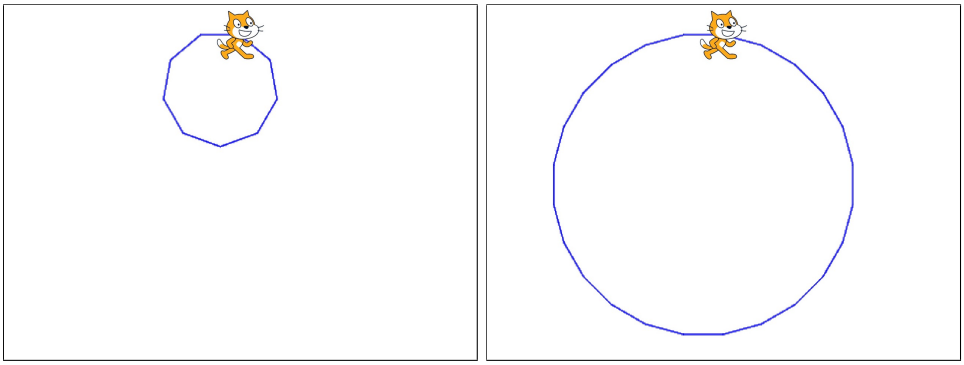
120° / Dreieck

30	×	12	=	360
45	×	8	=	360
60	×	6	=	360
72	×	5	=	360
90	×	4	=	360
120	×	3	=	360
Winkel	×	**Eckenzahl**	=	360



Aufgabe 4.18

Wie kannst du Scratch dazu bringen, mit mehrmaligem Drücken der →-Taste dieses Neuneck und diesen Kreis zu malen?

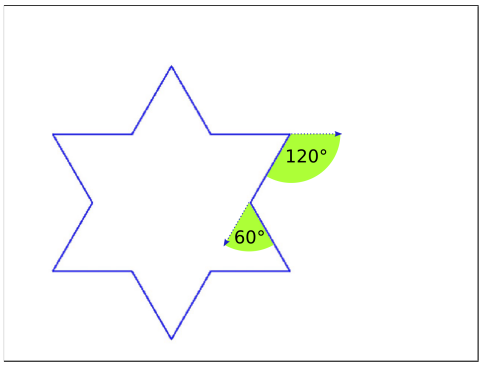


Tipp: Der Kreis ist in Wirklichkeit kein Kreis, sondern ein 24-Eck! Du kannst für beide Bilder das gleiche "Grundskript" nehmen und musst jeweils nur den Drehwinkel ändern.

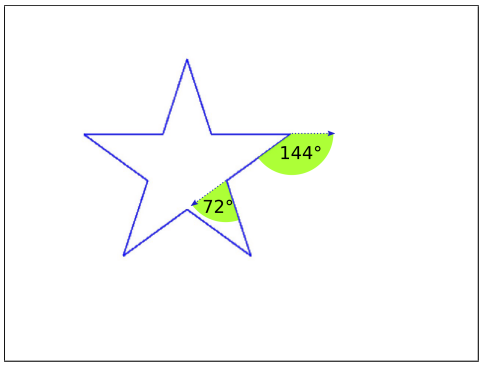
Aufgabe 4.19

Lass Scratch mit mehrmaligem Drücken der →-Taste diese beiden Sterne zeichnen!

a)



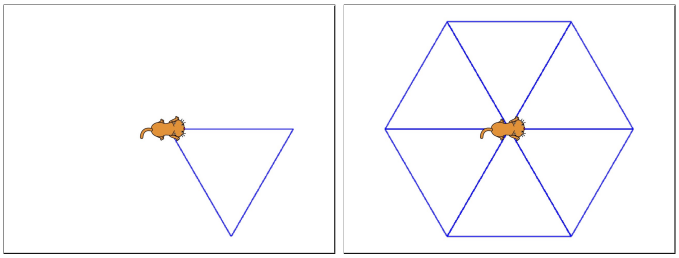
b)



Tipp: ein Tastendruck = *zwei* Linien!

Aufgabe 4.20

Mit den beiden Skripten rechts malt Scratch (nach dem Aufräumen durch Klick auf die grüne Flagge) auf Druck der Leertaste immer wieder das gleiche Dreieck (Bild unten links). Wie musst du das Skript für die Leertaste ergänzen, damit durch wiederholtes Drücken der Leertaste das Bild unten rechts entsteht? Schreibe deine Ergänzung zuerst auf und überprüfe dann, ob sie stimmt!
Tipp: Denke zurück an die Aufgabe 4.17!



Aufgabe 4.21

Erstelle ein Skript, mit dem du deine Initialen (Anfangsbuchstaben deines Vor- und Nachnamens) malen kannst (gross oder klein)! Unten sind drei Beispiele. Der Punkt zeigt jeweils, wo die Zeichnung beginnt.


Wenn Taste q gedrückt wird
setze Richtung auf 0 Grad
schalte Stift ein
gehe 100 er Schritt
drehe dich um 90 Grad
gehe 50 er Schritt
drehe dich um 90 Grad
gehe 50 er Schritt
drehe dich um 90 Grad
gehe 50 er Schritt
schalte Stift aus
Wenn Taste e gedrückt wird
setze Richtung auf 90 Grad
schalte Stift ein
gehe 50 er Schritt
drehe dich um 90 Grad
gehe 25 er Schritt
drehe dich um 90 Grad
gehe 50 er Schritt
drehe dich um 90 Grad
gehe 50 er Schritt
drehe dich um 90 Grad
gehe 50 er Schritt
schalte Stift aus
Wenn Taste r gedrückt wird
setze Richtung auf 0 Grad
schalte Stift ein
gehe 50 er Schritt
gehe -10 er Schritt
drehe dich um 45 Grad
gehe 14 er Schritt
drehe dich um 45 Grad
gehe 40 er Schritt
schalte Stift aus

Kapitel 5

Wiederholungen

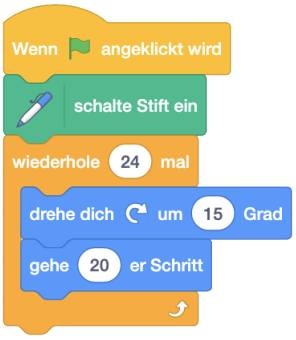
Hier erfährst du, wie du Befehle und Befehlsfolgen mehrfach ausführen kannst. Dadurch entstehen auf einfache Weise "komplizierte" Abläufe, die du aber gut planen musst. Du erfährst, wie du Objekte miteinander interagieren lassen kannst.
Die Blöcke, die du dabei neu kennenlernst:

Die Begriffe, die hier neu eingeführt werden:

- Wiederholung (Schleife, Schleifenkörper)
- Ablauf (Kontrollfluss)
- Eigener Block

5.1 Dinge mehrmals tun

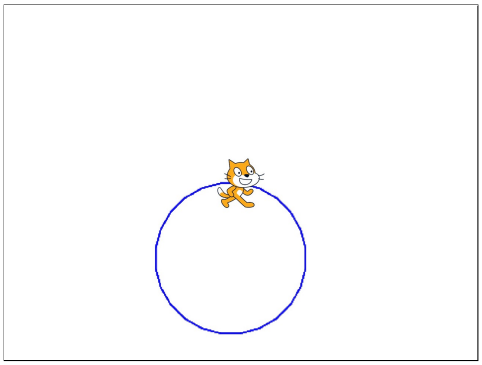
Erstelle dieses Skript und klicke auf die grüne Flagge über der Bühne. Was passiert?



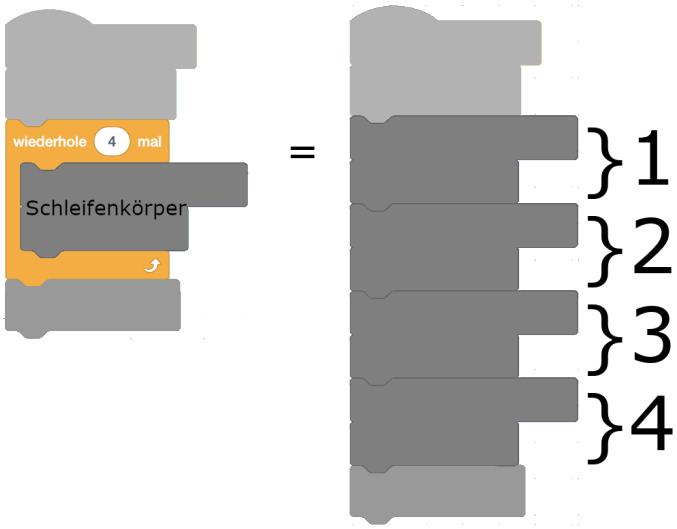
Wenn es dir zu schnell geht, kannst du den Block 

einfügen:

Beim Ereignis "Grüne Flagge angeklickt" malt Scratch "von selbst" einen Kreis (genauer: ein 24-Eck).



Der *Wiederhole-Mehrmals* Befehl führt die von ihm umschlossene Befehlsfolge so oft hintereinander aus, wie der Parameter des Blocks angibt. In der Informatik nennt man diese Wiederholungen auch *Schleife*. Die umschlossene Befehlsfolge nennt man *Schleifenkörper*.



Aufgabe 5.1

Nenne Beispiele aus dem Alltag, wo man einen bestimmten Ablauf wiederholt hintereinander ausführen muss. Dabei muss das Resultat einer Wiederholung nicht immer dasselbe sein. Einkaufen gehen ist ein gutes Beispiel: Zuerst gehst du in den Laden. Dann wiederholst du für jedes Lebensmittel auf deiner Einkaufsliste die folgenden Schritte:

1. Gehe zur passenden Abteilung
2. Suche das Lebensmittel im Regal
3. Packe das Lebensmittel in deinen Einkaufskorb oder -wagen

Aufgabe 5.2

Schreibe auf, was Scratch mit dem folgenden Programm sagt, wenn man auf die

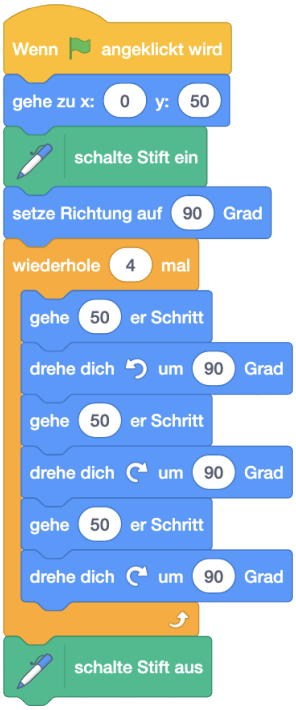
grüne Flagge klickt!

Aufgabe 5.3

Welche Figuren zeichnet Scratch mit den folgenden Skripten? Zeichne deine Lösung auf, bevor du sie mit Programmieren kontrollierst.

a)

b)



c)

Tipp: Um die Wiederholungen leichter mitzuzählen kannst du für jede Wiederholung eine andere Farbe verwenden.

Aufgabe 5.4

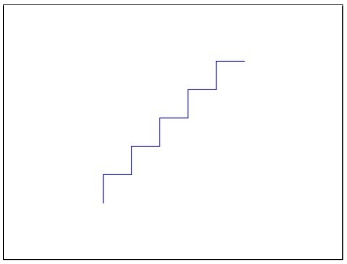
Wie kannst du diese Muster malen? Arbeite hier nicht mit den Tasten sondern

mit dem Hut 

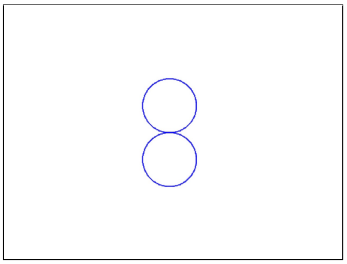
und dem Wiederhole-Mehrmals-Befehl.

Tipp: Mache dir einen Plan! Nicht alle Arten, um die Figuren zu zeichnen, sind gleich einfach zu programmieren!

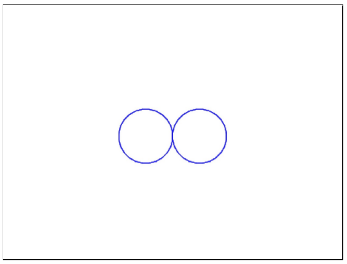
Treppe



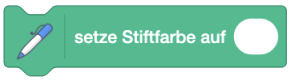
Acht



Liegende Acht



Aufgabe 5.5

Verkleide Scratch als Schneemann und erstelle ein Skript, mit dem man an seiner Position eine Schneeflocke malen kann (durch *einmaligen* Druck auf die Leertaste)! Durch Herumziehen des Schneemanns auf der Bühne kannst du so eine Winterlandschaft malen. Du kannst dafür die beiden Befehle 

und 

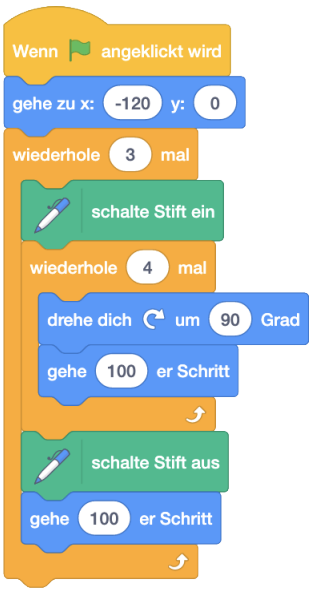
benutzen! Wie lautet dein Skript?

Zusatzaufgaben:

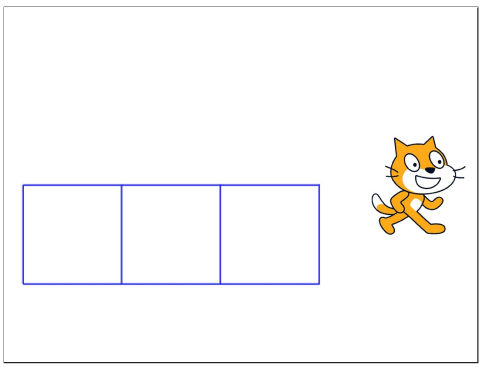
1. Schaffst du auch diese Schneeflocke mit einem einzigen Skript?
2. Denke dir eine eigene Schneeflocke aus und versuche, sie zu programmieren!

5.2 Verschachtelte Schleifen

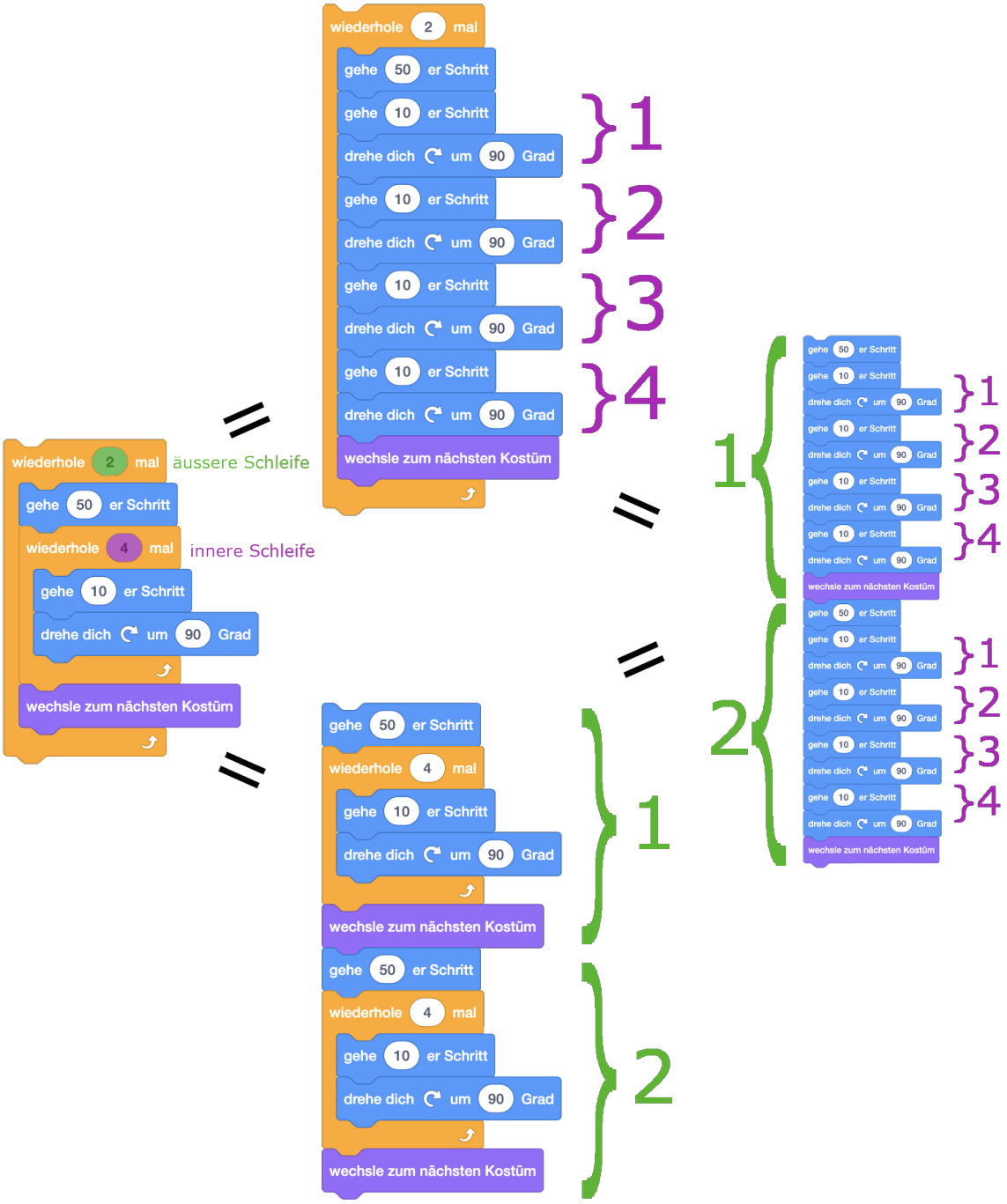
Erstelle dieses Skript und klicke auf die grüne Flagge! Was passiert?



Scratch zeichnet dieses Muster mit drei Quadraten:



Wiederholungen können auch verschachtelt vorkommen:



Man spricht dann auch von einer *inneren* und einer *äusseren* Schleife. Die äussere Schleife umschliesst dabei die innere Schleife.

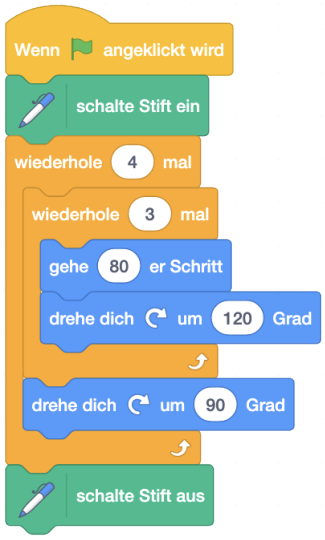
Aufgabe 5.6

1. Wieviele Schritte läuft Scratch mit den folgenden Programmen? Schreibe deinen Rechenweg auf! Wie bereits in Aufgabe 3.3 zählen hier auch Schritte

rückwärts (100 Schritte vorwärts + 100 Schritte rückwärts = 200 Schritte).

2. Zeichne die Bilder, die Scratch mit den folgenden Programmen zeichnet! Für das zweite Programm reicht es, wenn du für jedes Stempelbild eine Strichfigur zeichnest. Es muss aber erkennbar sein, in welche Richtung Scratch schaut!

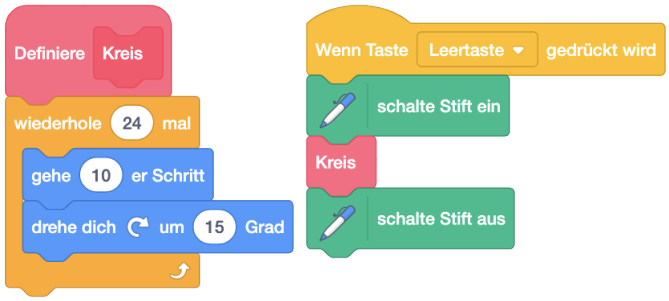
a)



b)

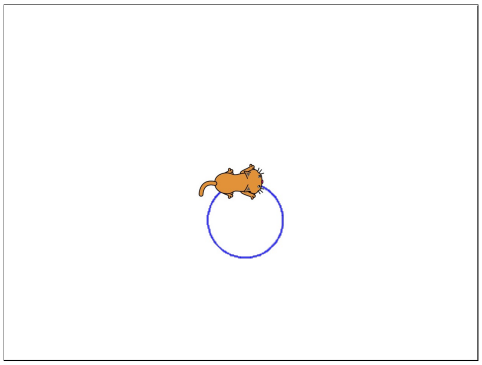
5.3 Eigene Blöcke

Erstelle einen neuen Block mit dem Namen “Kreis” indem du in der Blockpalette unter “Meine Blöcke” auf “Neuer Block” klickst und dann als Blocknamen “Kreis” eingibst. Erstelle dann dieses Programm:

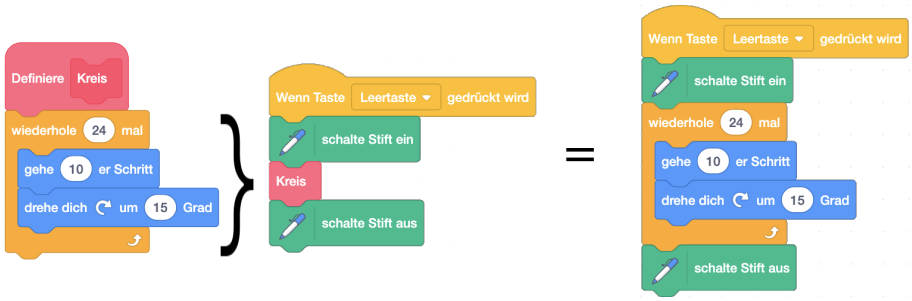


Was passiert, wenn du die Leertaste drückst?

Scratch malt einen Kreis (genauer: ein 24-Eck).



Mit einem *eigenen Block* kannst du einen eigenen Befehl definieren, der überall dort eingesetzt wird, wo du den eigenen Block einfügst:



Unter dem "Definiere"-Hut schreibst du das Skript, das überall dort ausgeführt werden soll, wo du den eigenen Block einfügst. Dadurch kannst du im Programmierbereich Platz sparen und dein Programm wird besser lesbar, wenn du deinen Blöcken gute Namen gibst. Sobald du einen neuen Block definiert hast, kannst du ihn überall wie die anderen Scratch-Blöcke benutzen, auch in anderen Definitionen!

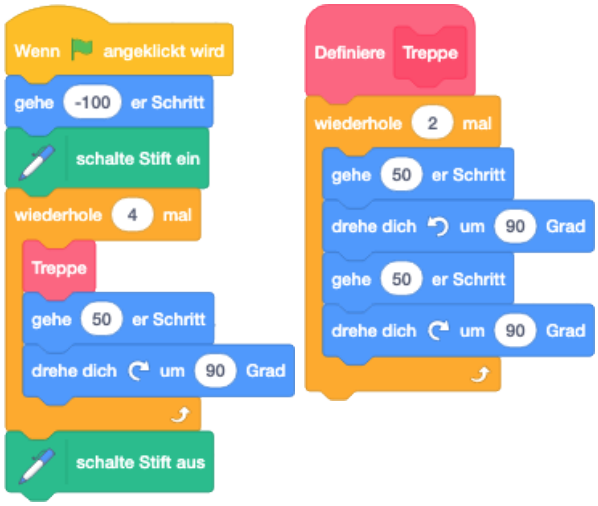
Mit eigenen Blöcken erweiterst du die Sprache von Scratch um neue Befehle. Du kannst so deine Programme leichter lesbar machen. Zum Beispiel kannst du mit eigenen Blöcken verschachtelte Schleifen vermeiden.

Aufgabe 5.7

1. Wieviele Schritte läuft Scratch mit den folgenden Programmen? Schreibe deinen Rechenweg auf! Wie bereits in Aufgabe 3.3 zählen hier auch Schritte rückwärts (100 Schritte vorwärts + 100 Schritte rückwärts = 200 Schritte).

2. Zeichne die Bilder, die Scratch mit den folgenden Programmen zeichnet!

a)

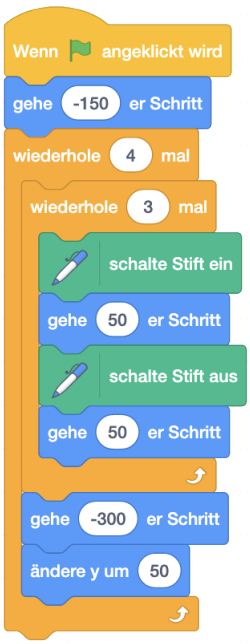


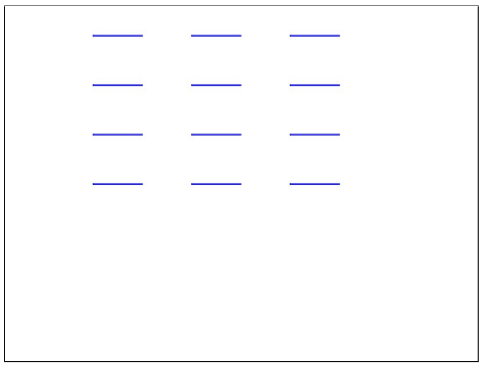
b)

Aufgabe 5.8

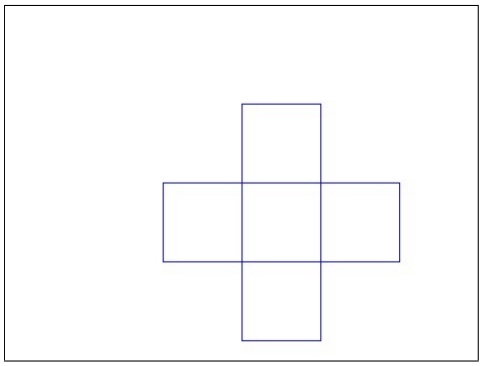
Scratch zeichnet mit den beiden Skripten unten die jeweiligen Bilder. Definiere eigene Blöcke, um das Programm lesbarer zu machen!

a)



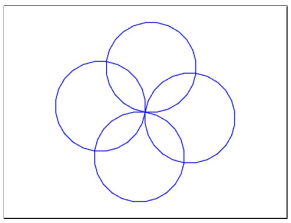


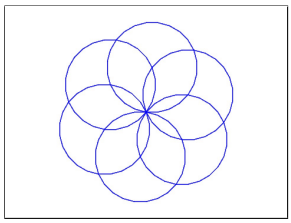
b)

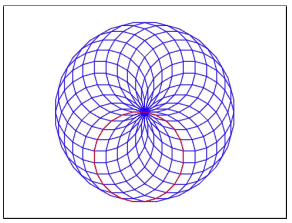


Aufgabe 5.9

Kannst du jedes dieser Kreismuster mit einem einzigen Tastendruck malen? Verwende dazu einen eigenen Block! Du kannst immer das gleiche Grundskript benutzen und musst jeweils nur einen Drehwinkel und die Anzahl der Wiederholungen ändern!

a) 
4 Kreise

b) 
6 Kreise

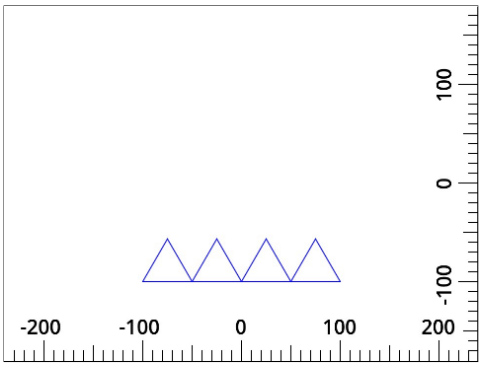
c) 
24 Kreise

Ablauf: Wiederhole mehrmals diese beiden Schritte:

1. Male einen Kreis (im rechten Bild rot)
2. Drehe dich

Aufgabe 5.10

Scratch soll diese vier Dreiecke zeichnen, doch im Programm hat sich ein Fehler eingeschlichen. Findest du den Fehler und kannst du das Programm korrigieren? Als Hilfestellung sind die Koordinaten angeschrieben.



Aufgabe 5.11

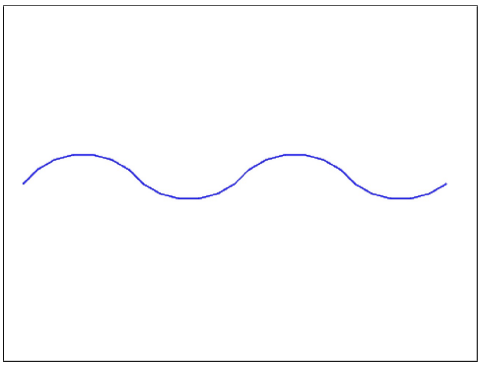
Lass verschiedene Figuren tanzen! Verwende dabei eigene Blöcke, um gewisse Teile des Tanzes zu definieren (zum Beispiel einen Salto machen oder nach rechts

hüpfen).

Aufgabe 5.12

Kannst du diese Welle mit einem einzigen Skript malen? Den gleichen Ablauf kannst du zum Beispiel verwenden, um ein Schiff langsam übers Wasser schaukeln zu lassen (rechts). Benutze dabei den Befehl 

.

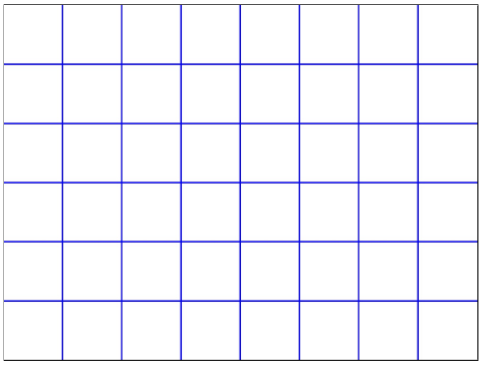


Ablauf:
Drehe dich zuerst um 45° ↻ und wiederhole dann mehrmals diese beiden Schritte:

1. Male einen Viertelkreis ↻
2. Male einen Viertelkreis ↺

Aufgabe 5.13

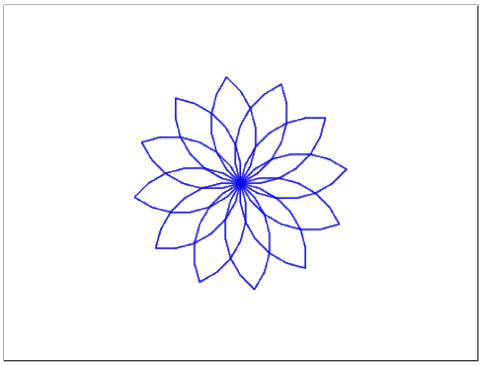
Erstelle ein Skript, das die Bühne mit diesem Gittermuster bemalt (8 Kästchen in der Breite, 6 in der Höhe. Die Gitterlinien haben also jeweils 60 Schritte Abstand voneinander). Nach aussen ist das Gitter offen, am Bühnenrand sollen also keine Linien gezeichnet werden! Versuche hier, selbst den Ablauf herauszufinden.



Tipp: Male erst die Linien von links nach rechts, dann die von unten nach oben!

Aufgabe 5.14

Kannst du die Blume unten mit einem einzigen Tastendruck malen? Verwende dazu eigene Blöcke für die einzelnen Teile der Blume!



Ablauf:
Wiederhole mehrmals diese beiden Schritte:

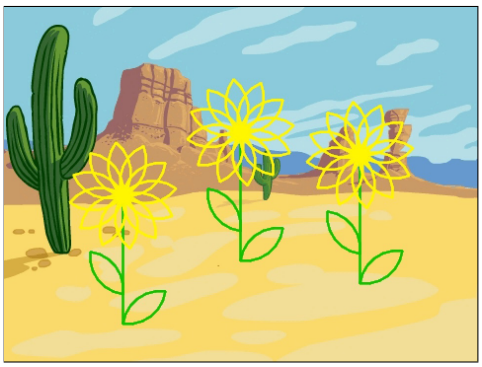
1. Male ein Blütenblatt
2. Drehe dich

Ablauf für ein einzelnes Blütenblatt:
Wiederhole mehrmals diese beiden Schritte:

1. Male im Uhrzeigersinn einen Viertelkreis
2. Drehe dich um 90° im Uhrzeigersinn

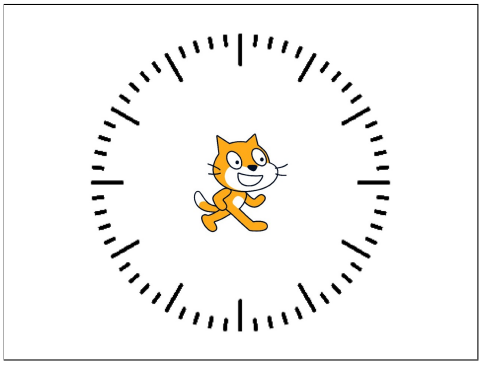
Den gleichen Ablauf kannst du verwenden, um Blumen in der Wüste wachsen zu lassen! Du musst dabei nur vorher jeweils noch zwei Blätter und einen Stiel

malen! In welchem Winkel und Abstand die beiden Blätter sein sollen, kannst du selbst mit etwas Ausprobieren festlegen.



Aufgabe 5.15

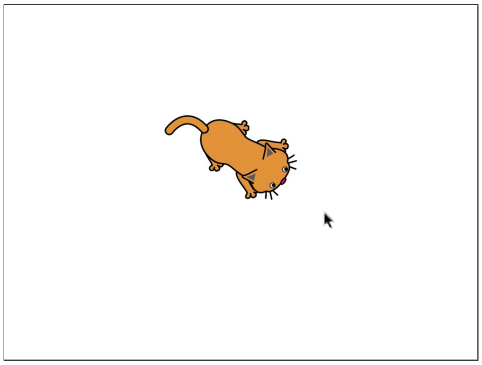
Male das Ziffernblatt einer Uhr! Stiftfarbe und Stiftdicke kannst du frei wählen. Versuche hier zuerst selbst, den Ablauf herauszufinden!



5.4 Dinge endlos tun

Erstelle für Scratch dieses Skript. Klicke dann die grüne Flagge an und bewege den Mauszeiger über die Bühne. Was passiert?

Scratch versucht die Maus zu fangen: Scratch dreht sich fortlaufend zur Maus und geht dabei jedes Mal fünf Schritte auf die Maus zu.



Der *Wiederhole-Fortlaufend*-Befehl sorgt dafür, dass die von ihm umschlossene Befehlsfolge immer wieder und ohne Ende ausgeführt wird. Klicken auf das Skript oder auf den roten Punkt über der Bühne (stoppe *alle* Skripte) stoppt auch den Wiederhole-Fortlaufend-Befehl.

Aufgabe 5.16

Verändere dein Projekt aus Aufgabe 5.11 so, dass deine Figuren fortlaufend tanzen, wenn sie angeklickt werden.

Aufgabe 5.17

Wir können Scratch statt den Mauszeiger auch eine "richtige" Maus verfolgen lassen: 

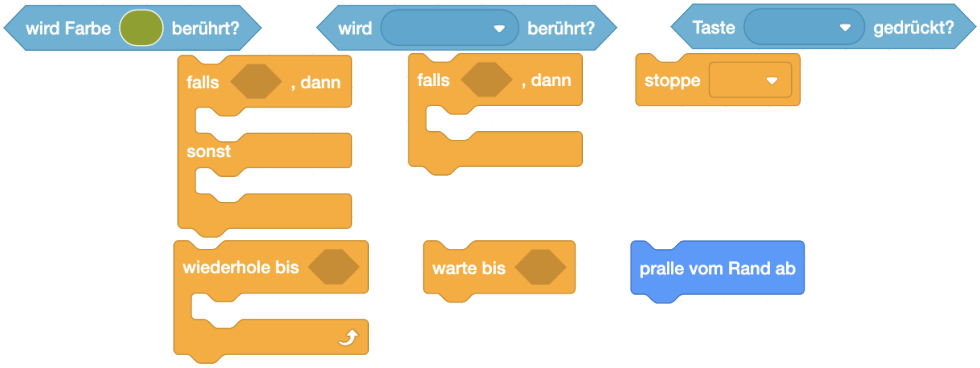
Lade eine Maus als zweites Objekt und erstelle ein Skript mit dem die Maus beim Klick auf die grüne Flagge selbst vor Scratch davonlaufen kann (zum Beispiel im Kreis)! Gib Scratch das "Verfolgungsskript" und wähle im Menü des Befehls statt Mauszeiger die Maus aus (nachdem du das Objekt sinnvoll umbenannt hast):

Schalte den Stift ein, um den von Scratch gelaufenen Weg zu sehen.

Kapitel 6

Bedingungen

Hier lernst du, wie du etwas tun kannst, falls eine bestimmte Bedingung gilt, und wie du darauf warten kannst, dass sie gilt. Damit bist du schon in der Lage, Spiele zu programmieren, in denen Objekte auf Berührungen von Farben und anderen Objekten reagieren.
Die Blöcke, die du dabei neu kennenlernst:

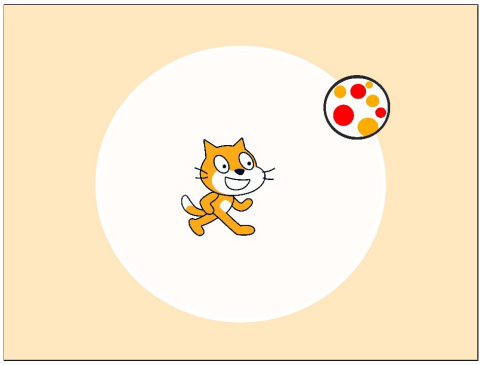


Die Begriffe, die hier neu eingeführt werden:

- Bedingung (Prädikat)
- Befehl mit Bedingung (Verzweigung)
- Falls-Zweig und Sonst-Zweig
- Flussdiagramm

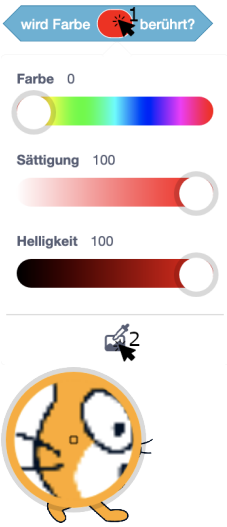
6.1 Bedingungen prüfen

Lade den Hintergrund “Light” und den Beachball als zweites Objekt.

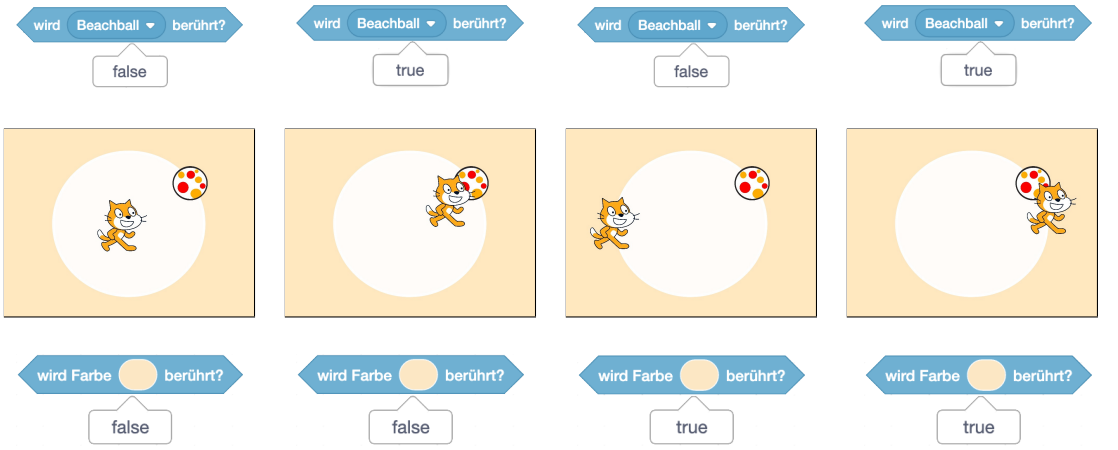


Ziehe diese drei Blöcke in den Programmierbereich von Scratch und klicke sie an. Versuche bei allen drei Blöcken, zwei verschiedene Antworten zu bekommen!

Tipp: mit der Pipette kannst du eine bestimmte Farbe der Bühne oder eines Objekts auswählen:



Die Antwort ist “true” (Englisch für “wahr”) oder “false” (englisch für “falsch”), abhängig von der Situation, die beim Klicken auf den jeweiligen Block vorliegt.

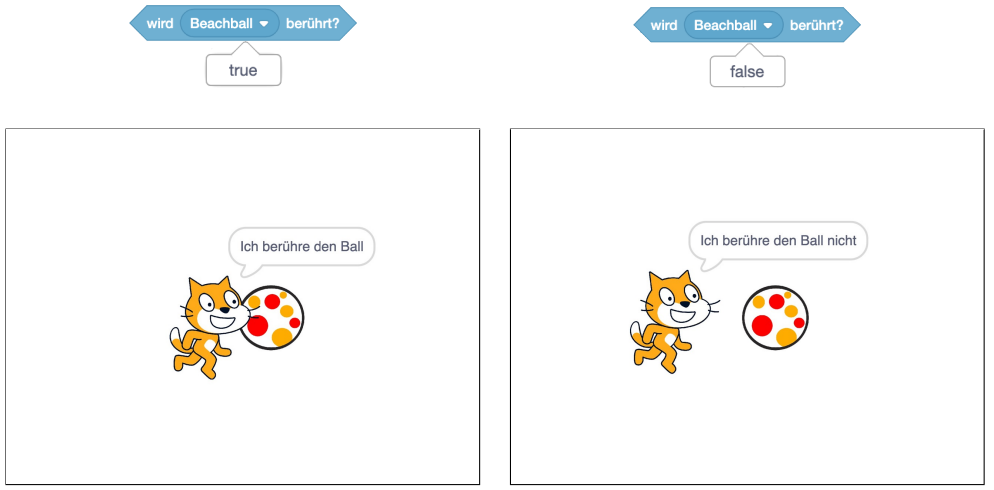


Jeder Block der Form [redacted] ist eine *Bedingung*. Der Text in der Bedingung ist eine Ja-Nein-Frage. Bei jedem Anklicken der Bedingung wird die Ja-Nein-Frage der Situation entsprechend beantwortet. Antwort “true” bedeutet “Ja, das stimmt, die Bedingung gilt!”. Antwort “false” heisst “Nein, das stimmt nicht, die Bedingung gilt nicht!”

6.2 Dinge tun, falls. . . , sonst. . .

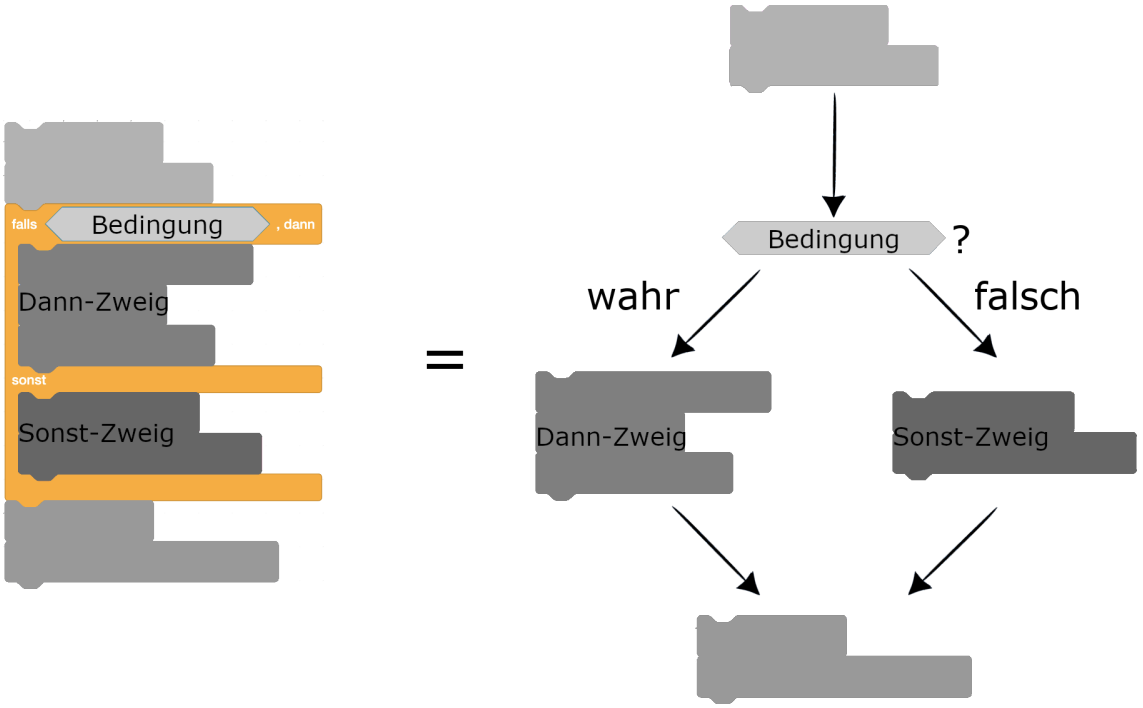
Lade wieder den Beachball als zusätzliches Objekt und erstelle dieses Skript für Scratch:
Ziehe dann Scratch auf der Bühne herum und drücke die Leertaste. Was passiert?

Auf Drücken der Leertaste sagt Scratch “Ich berühre den Ball’, wenn der Beachball berührt wird. Wenn der Beachball nicht berührt wird, sagt Scratch “Ich berühre den Ball nicht”.



Auf Deutsch können die Wörter “Wenn” und “Falls” als Synonyme benutzt werden. In Scratch haben sie aber jeweils eine spezifische Aufgabe: Hüte beginnen mit “Wenn” und werden immer dann ausgeführt, wenn das Ereignis eintritt. Bedingungen beginnen mit “Falls” und werden nur an der Stelle im Skript überprüft wo sie vorkommen.

Jeder Befehl mit einer Aussparung dieser Art  ist ein *Befehl mit Bedingung*. Der *Falls-Dann-Sonst-Befehl* sorgt dafür, dass die erste von ihm umschlossene Befehlsfolge ausgeführt wird, falls seine Bedingung gilt. Diese Befehlsfolge nennt man auch *Dann-Zweig*. Gilt die Bedingung nicht, wird stattdessen die zweite Befehlsfolge ausgeführt. Diese Befehlsfolge nennt man *Sonst-Zweig*. In beiden Fällen geht es danach mit dem nächsten Befehl im Skript weiter. Welcher Teil ausgeführt wird, lässt sich gut mit einem *Flussdiagramm* darstellen:



Aufgabe 6.1

Bilde Sätze der Form "Falls ..., dann ..., sonst ...".
Zum Beispiel:
"Falls ich genug Geld habe, dann gehe ich ins Kino, sonst lese ich ein Buch."

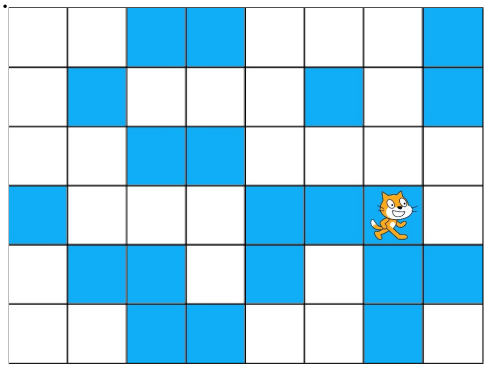
Aufgabe 6.2

Zeichne das Flussdiagramm für den Wiederhole-Fortlaufend-Befehl:

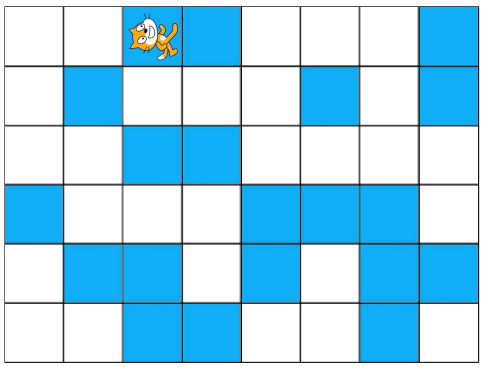
Aufgabe 6.3

Zeichne für jede der unten stehenden Ausgangslagen ein, auf welchem Feld Scratch landet und in welche Richtung Scratch schaut, wenn das Programm rechts ausgeführt wird. Ein Feld ist jeweils 60 Schritte hoch und 60 Schritte breit.

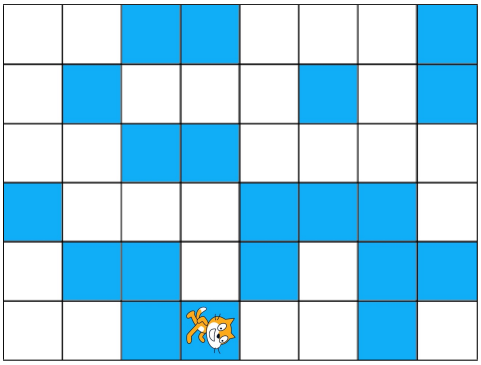
1.



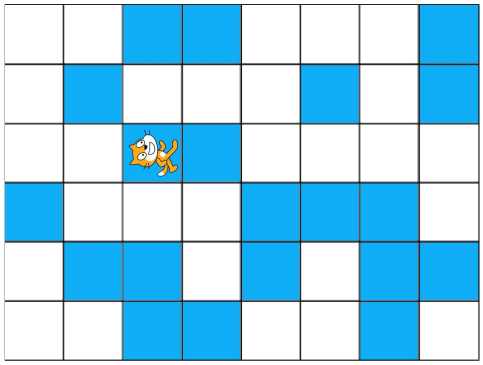
2.



3.

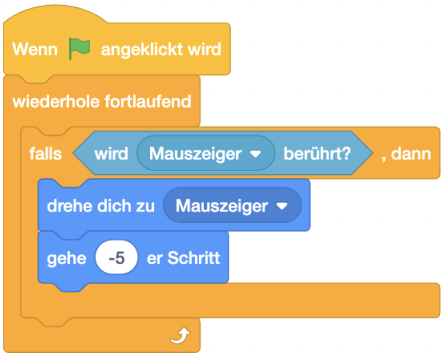


4.

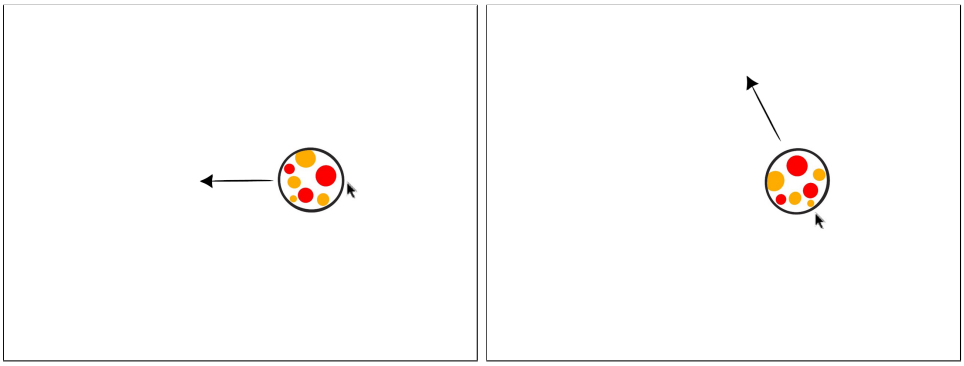


6.3 Dinge tun, falls...

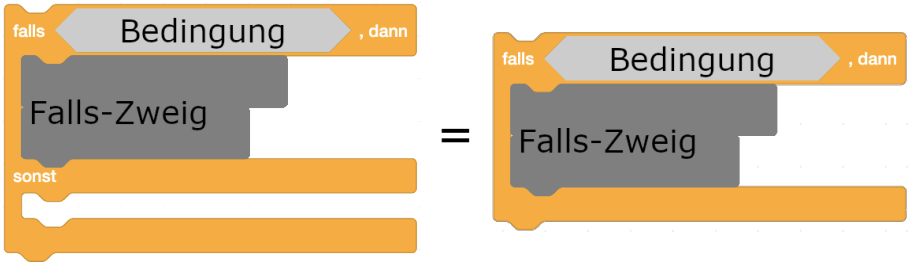
Erstelle für den Ball dieses Skript und klicke die grüne Flagge an! Was passiert, wenn du mit dem Mauszeiger den Ball berührst?



Jedes Mal, wenn er vom Mauszeiger berührt wird, springt der Ball ein kleines Stück vom Mauszeiger weg. Auf diese Weise kannst du den Ball mit der Maus auf der Bühne herumschieben.



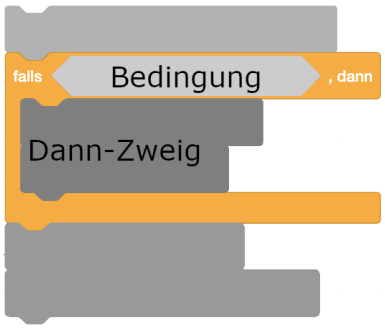
Den *Falls-Dann*-Befehl kannst du verwenden, wenn du nur etwas tun willst, wenn die Bedingung gilt. Wenn die Bedingung nicht gilt, wird die Befehlsfolge im Dann-Zweig nicht ausgeführt. In beiden Fällen geht es danach mit dem nächsten Befehl im Skript weiter. Im gezeigten Beispiel soll der Ball nichts tun, wenn er nicht vom Mauszeiger berührt wird.



Die Kombination von einem *Falls-Dann-* oder *Falls-Dann-Sonst*-Befehl mit dem *Wiederhole Fortlaufend*-Befehl ist sehr nützlich, wenn du eine Bedingung laufend überprüfen willst.

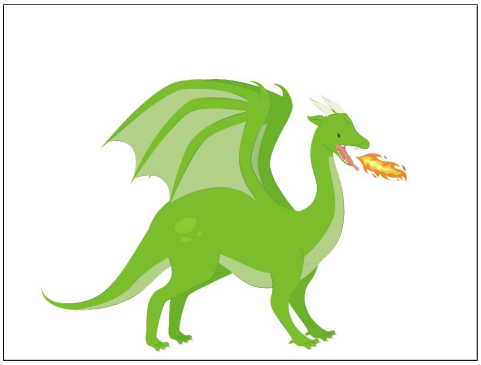
Aufgabe 6.4

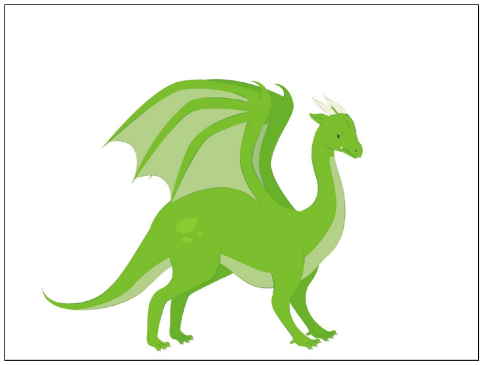
Zeichne das Flussdiagramm für den Falls-Dann-Befehl:

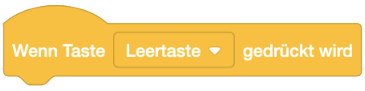


Aufgabe 6.5

Programmiere einen Drachen, der mit Druck der Leertaste Feuer spuckt (durch Kostümwechsel). Wenn die Leertaste nicht gedrückt ist, soll der Drache kein Feuer spucken.





Tipp: Starte nicht mit 

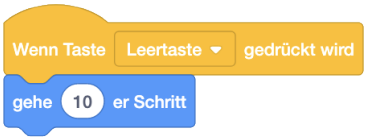
sondern mit 

!

Brauchst du für diese Aufgabe den Falls- oder den Falls-Sonst-Befehl?

Aufgabe 6.6

Erkläre die Unterschiede zwischen diesen beiden Skripten:

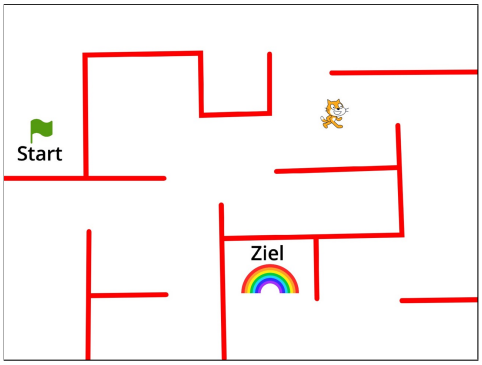


Was passiert, wenn die Leertaste gedrückt wird? Was passiert, wenn das Skript angeklickt wird?

Aufgabe 6.7

Male als Hintergrund ein Labyrinth (in einer bestimmten Farbe) und füge jeweils ein Objekt als Start und als Ziel hinzu (in einer anderen Farbe, du kannst zum Beispiel die grüne Flagge für den Start verwenden und den Regenbogen für das Ziel). Programmiere Scratch so, dass Scratch beim Anklicken der grünen Flagge zum Start geht und dann mit den Pfeiltasten in alle vier Richtungen laufen kann. Verwende dazu das folgende Skript:

Programmiere nun, dass Scratch bei Berührung der Wände des Labyrinths zurück zum Start geht. Zusätzlich soll dann ein lustiger Klang abgespielt werden! Wenn Scratch das Ziel erreicht, gewinnst du! Lass Scratch dann einen triumphierenden Klang spielen und "Geschafft" sagen.

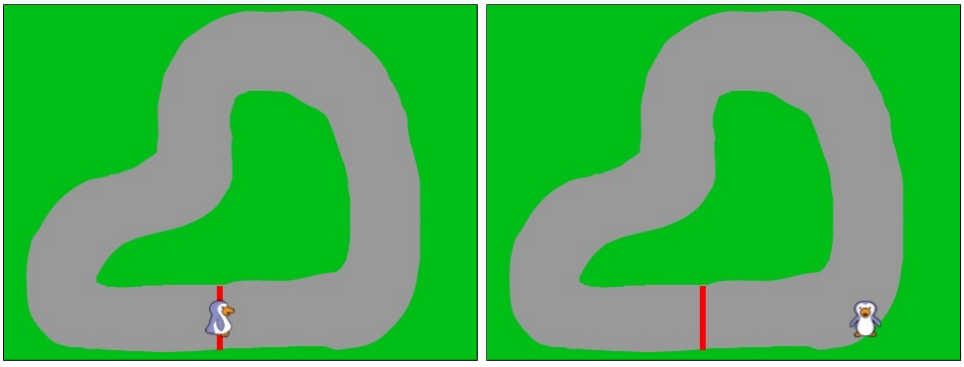


Tipps: Um das Spiel zu beenden, benutze den Befehl 

.

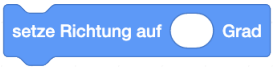
Aufgabe 6.8

Male als Hintergrund eine Rennbahn, lade ein neues Objekt (zum Beispiel einen Pinguin, eine Rakete oder ein Einhorn) und schrumpfe es auf die passende Grösse. Programmiere das Objekt so, dass es von selbst fortlaufend geradeaus läuft (oder fliegt), du es aber mit den Tasten ← und → nach links oder nach rechts drehen kannst. Berührt das Objekt den Rand der Rennbahn (Farbe überprüfen!), soll es einen Klang abspielen oder traurig aussehen (neues Kostüm!) und das Spiel ist vorbei.
Varianten: Du kannst auch ein Ziel einbauen, bei dessen Berührung das Spiel gewonnen ist (Erfolgsklang!), und du kannst während des Spiels fortlaufend passende Geräusche abspielen oder die Kostüme wechseln, um eine Laufanimation zu kreieren.



Tipp: Um das Spiel zu beenden, benutze den Befehl 

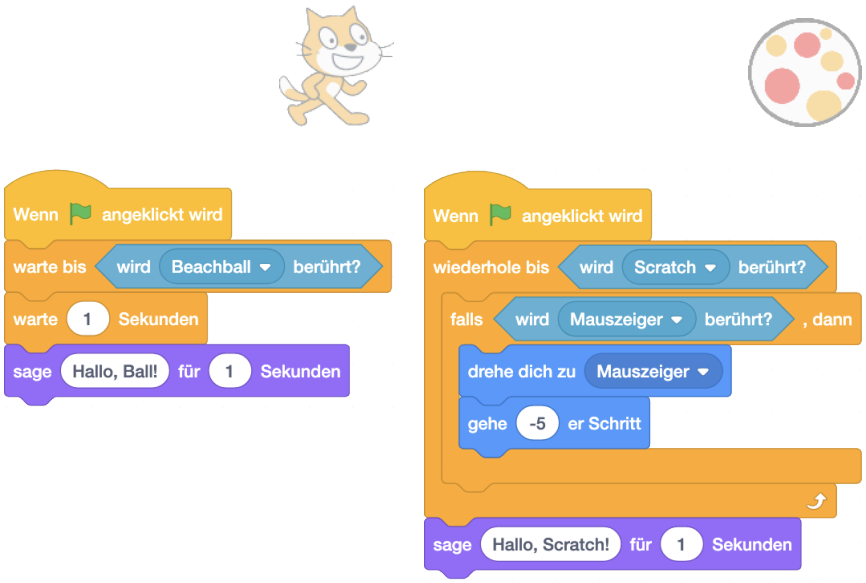
! Zu Beginn ("Grüne Flagge angeklickt") solltest du das Objekt mit Hilfe der Befehle 

und 

in eine sichere Startposition bringen.

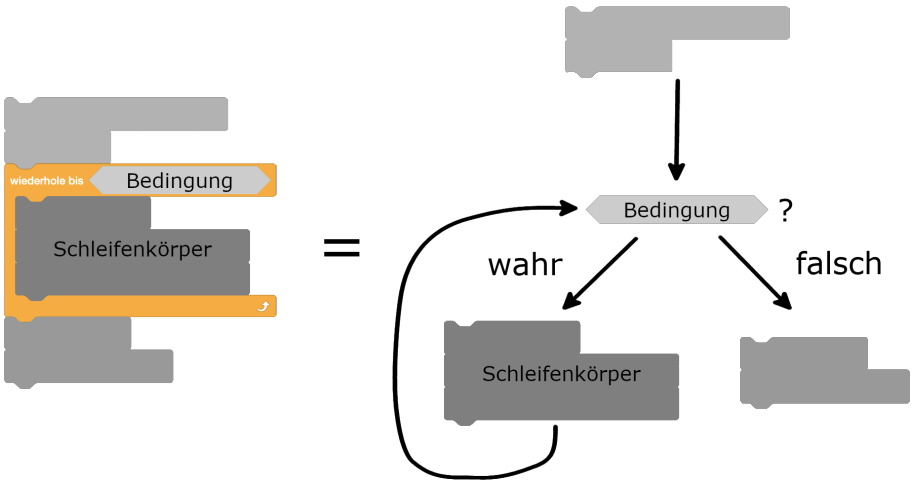
6.4 Dinge tun, bis...

Erstelle das jeweilige Skript für Scratch und den Beachball. Aufgepasst: Scratch (ursprünglich "Figur1") wurde hier passend umbenannt. Klicke dann die grüne Flagge an und schiebe den Ball zu Scratch. Was passiert?



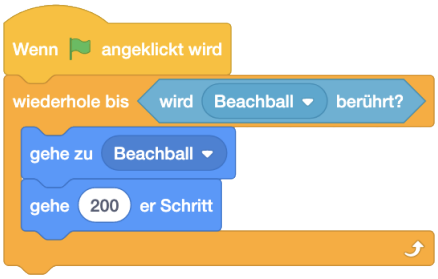
Scratch wartet auf den Ball, während der Ball sich vom Mauszeiger wegbewegt. Ist der Ball bei Scratch angekommen, begrüssen sich die beiden.

Der *Wiederhole-Bis Befehl* sorgt dafür, dass die von ihm umschlossene Befehlsfolge solange wiederholt wird, bis seine Bedingung gilt. Danach geht es mit dem nächsten Befehl im Skript weiter. Vor jeder Wiederholung wird die Bedingung neu geprüft. Es kann also auch passieren, dass die Befehlsfolge gar nicht oder endlos ausgeführt wird.



Innerhalb des Schleifenkörpers wird die Bedingung aber nicht überprüft. Es kann

also durchaus vorkommen, dass die Bedingung zwischendurch erfüllt ist, aber die Wiederholung nicht aufhört, weil die Bedingung direkt vor der nächsten Wiederholung nicht mehr gilt.
Beispiel: In diesem Skript wird die Wiederholung entweder gar nicht (wenn der Beachball vor dem Wiederhole-Bis Befehl berührt wird) oder endlos wiederholt (da Scratch den Beachball zum Zeitpunkt der Überprüfung der Bedingung nie berührt):

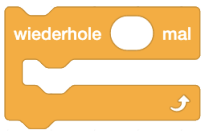


Nimm den *Warte-Bis Befehl*, wenn du *nichts* tun willst, bis die Bedingung erfüllt ist:

Aufgabe 6.9

Verändere das Rennbahn-Spiel aus Aufgabe 6.8!

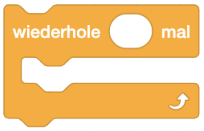
1. Man hat drei (oder mehr) Leben: Wenn der Rand der Rennbahn berührt wird, springt das Objekt wieder zum Start zurück, und das Spiel beginnt von neuem. Erst nach dreimaliger Berührung des Randes heisst es wirklich "Game over". Benutze den 

Befehl, um das Objekt fortlaufend geradeaus zu bewegen, bis der Rand berührt wird. Das kannst du dann mit Hilfe des 

Befehls so oft wiederholen, wie dein Objekt Leben hat.

2. Verändere das Rennbahn-Spiel so, dass du erst nach einer festgelegten An-

zahl von Runden gewonnen hast. Erstelle dazu ein Zielobjekt, das mit Hilfe des Befehls 

wartet, bis es vom Objekt berührt wird. Das kannst du dann mit dem 

Befehl so oft wiederholen, wie es Runden geben soll. Du musst dabei aufpassen, dass eine Zielberührung nicht gleich auch für alle weiteren Runden zählt. Um das zu verhindern, kannst du das Zielobjekt mit dem 

Befehl erst nach einer kurzen Zeit mit dem Warten auf die nächste Zielberührung beginnen lassen.

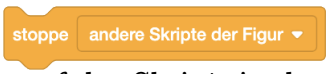
Lassen sich die beiden Erweiterungen gleichzeitig benutzen? Unterscheide dabei, ob die Rundenzahl zurückgesetzt werden soll, wenn der Rand berührt wird, oder nicht. Überlege dir auch, was passiert, wenn du das Objekt nach einer Randberührung wieder zurück auf die Start-/Ziellinie setzt. Programmiere und teste deine Lösung oder erkläre, wieso das nicht geht.

Aufgabe 6.10

Schau dir das folgende fehlerhafte Programm an:

Skript für Scratch: *Skript für die Maus:*

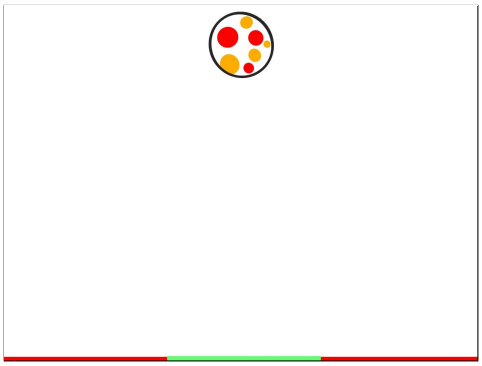
Scratch soll mit dem Mauszeiger bewegt werden können. Die Maus soll sofort stehenbleiben und um Hilfe rufen, sobald sie von Scratch berührt wird. Mit diesem Programm kann es aber vorkommen, dass Scratch die Maus berührt und die Maus sich einfach weiterbewegt. Kannst du erklären, wo hier der Fehler liegt? Hast du auch eine Idee, wie man diesen Fehler beheben kann?

Tipp: Mit dem 

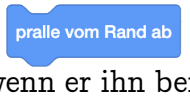
Befehl kannst du alle Skripte des Objekts beenden, bis auf das Skript, in dem der Befehl eingebaut ist!

Aufgabe 6.11

Hier lernst du, wie man eine vereinfachte Version des Spieleklassikers Pong programmiert. Wir werden zuerst eine 1-Spieler-Version programmieren, in der es darum geht, einen Ball in der Luft zu halten. Später kannst du das Spiel noch erweitern.
Zuerst müssen wir definieren, was das Programm erreichen soll. Wir werden zwei Objekte benutzen: Einen Ball und einen "Schläger". Zusätzlich zeichnen wir einen Bühnenhintergrund mit einem roten Lava-Boden, den der Ball nicht berühren soll.



Ball Der Ball soll bei Klick auf die grüne Flagge im oberen Teil der Bühne starten und sich schräg nach unten bewegen. Wo genau der Ball starten soll und in welche Richtung er sich nach unten bewegen soll kannst du selbst entscheiden.

Der Ball soll von den Wänden und vom Schläger (unterer Bühnenrand) abprallen. Dazu kannst du diesen Befehl verwenden: 

Mit diesem Befehl prallt der Ball vom Rand der Bühne ab, wenn er ihn berührt. Wenn der Ball den Rand nicht berührt, passiert nichts. [1]

Wenn der Ball auf den Lava-Boden trifft ist das Spiel vorbei. Der Ball soll dann "Game Over" sagen und sich nicht mehr bewegen. Ob der Ball den Lava-Boden berührt, kannst du mit der 

Bedingung überprüfen. Die richtige Farbe findest du mit der Pipette. Mit dem Befehl 

beendest du das Spiel. Dadurch hört auch der Schläger auf, sich zu bewegen.

Schläger Der Schläger soll sich mit der Maus mitbewegen, aber seine Höhe (y-Position) und Richtung dabei nicht verändern. Tipp: Hier kannst du ausnutzen, dass der Computer Befehle sehr schnell ausführt!

Der Schläger soll sich ganz unten am Boden der Bühne bewegen und den Lava-Boden wie im Bild oben überdecken.

Lava-Boden Für den Lava-Boden malst du auf dem leeren Hintergrund-Kostüm unten eine gerade rote Linie. Achte darauf, dass die Linie vom Schläger verdeckt werden kann.

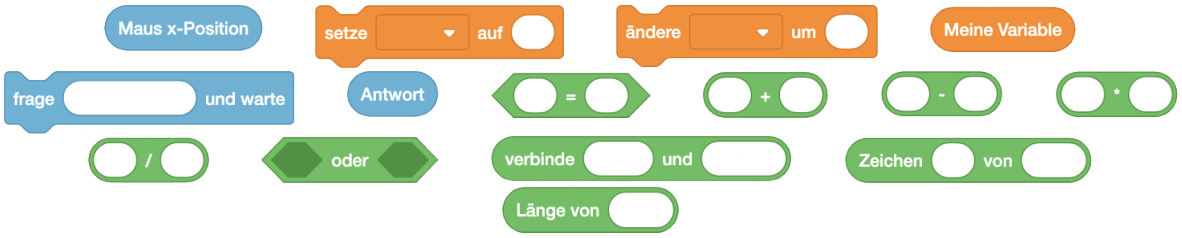
Lade die zwei Objekte: Als Ball kannst du den Beachball oder irgendeinen anderen Ball benutzen. Für den Schläger kannst du das Objekt "Line" laden. Passe dann die Grösse und Farbe der Objekte, sowie des Lava-Bodens auf dem Hintergrund an.

[1]Streng genommen hätten wir gerne, dass der Ball unten vom Schläger abprallt. Dies ist aber etwas schwieriger zu programmieren, und das Abprallen passiert so schnell, dass man den Unterschied nicht wirklich bemerkt.

Kapitel 7

Wert-Blöcke, Variablen und Operatoren

Hier lernst du, wie du verschiedene Informationen vom Computer abfragen kannst. Ausserdem wirst du deine eigenen Variablen definieren, um dir zum Beispiel einen Spielstand zu merken. Du lernst ausserdem, wie man einfache Berechnungen durchführen lässt und wie man eigene Blöcke mit Parametern definieren kann.
Die Blöcke, die du dabei neu kennenlernst:



Die Begriffe, die hier neu eingeführt werden:

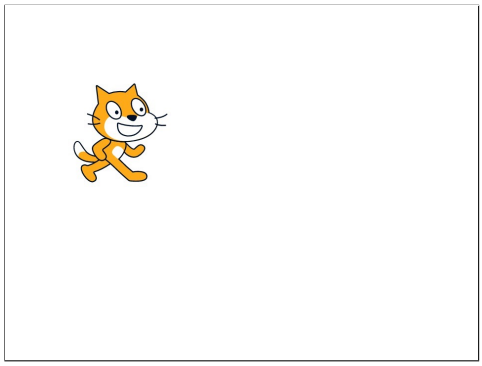
- Wert-Block
- Variable
- Operator
- Ausdruck
- formaler und tatsächlicher Parameter

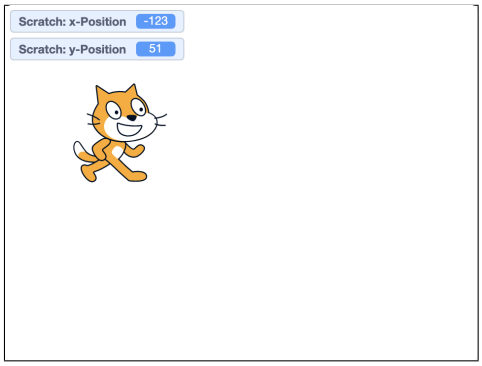
7.1 Wert-Blöcke

Erstelle dieses Skript und klicke auf die grüne Flagge. Was passiert, wenn du die Maus über die Bühne bewegst?

Scratch gleitet horizontal über die Bühne. Die y-Position verändert sich dabei nicht und die x-Position entspricht der x-Position des Mauszeigers.

Blöcke mit der Form [redacted] sind *Wert-Blöcke*. Wenn du einen solchen Block anklickst, "antwortet" der Block mit einem Wert, den der Computer gespeichert oder berechnet hat. So ähnlich haben wir das bereits bei den Wahrheits-Blöcken kennengelernt. Im Vergleich zu Wahrheits-Blöcken können Wert-Blöcke nicht nur "true" und "false" antworten. Der Wert kann sowohl eine Zahl, als auch eine Zeichenkette (zum Beispiel ein Wort) sein, je nachdem welche Funktion der Wert-Block hat. Wert-Blöcke können überall dort eingesetzt werden, wo es eine runde Aussparung hat. An dieser Stelle wird dann der Wert benutzt, den der Computer antwortet. Für manche Wert-Blöcke kann man in der Blockpalette ein Häkchen setzen. Dann wird der Wert auf der Bühne angezeigt.



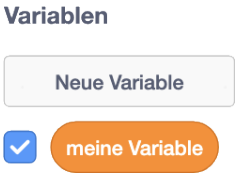


Aufgabe 7.1

Verbessere dein Pong Spiel, indem du die Bewegung des Schlägers mit den neu kennengelernten Wertblöcken programmierst.

7.2 Schritte zählen

Baue dir die folgenden Skripte zusammen und setze das Häkchen für "meine Variable". Die orangen Blöcke findest du im Blockmenü unter "Variablen".



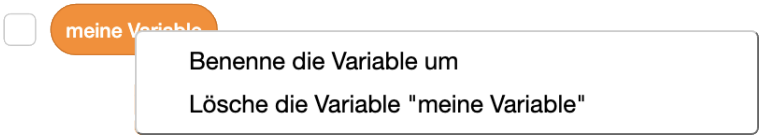
Bewege nun Scratch mit den Pfeiltasten ← und → über die Bühne und beobachte den Wert für "meine Variable", der oben links in der Bühne angezeigt wird.

Der Wert von "meine Variable" wird mit jedem 10er-Schritt um 10 vergrössert. Scratch kann so also seine Schritte zählen.

Eine *Variable* ist ein Behälter für einen Wert, den du dir merken willst. Eine Variable hat einen Namen und einen Wert. Im Beispiel oben hat die Variable mit dem Namen "meine Variable" nach Anklicken der Flagge den Wert 0. Eine Variable kann zu einem bestimmten Zeitpunkt immer nur einen Wert haben. Dieser Wert

kann sich aber verändern. Der Wert einer Variable wird im Computer an einer bestimmten Stelle gespeichert. Durch den Namen der Variable weiss der Computer, wo der Wert gespeichert ist. In Scratch können Variablen Zahlen oder Text als Wert enthalten. Mit dem Erstellen einer Variable kriegst du einen Wertblock, der jeweils den in der Variable gespeicherten Wert antwortet.

Auch bei Variablen ist es wichtig, einen sinnvollen Namen zu wählen. Eine Variable lässt sich mit Rechtsklick auf den Wertblock der Variable in der Blockpalette umbenennen:

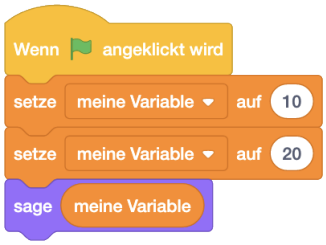


Was wäre ein besserer Name für die Variable im Einführungsbeispiel oben?

Aufgabe 7.2

Schreibe für die folgenden Skripte auf, welche Werte die verwendeten Variablen am Ende des Skriptes haben. Stelle dazu am besten eine Tabelle auf mit allen Variablen und ihren Werten. Wenn der Wert einer Variable verändert wird, streichst du den alten Wert durch und schreibst den neuen Wert auf. Schreibe ausserdem auf, was Scratch sagt!

1.



2.

3.


Wenn angeklickt wird
setze meine Variable auf 10
setze zweite Variable auf 20
setze meine Variable auf zweite Variable
setze zweite Variable auf 30

4.


Wenn angeklickt wird
setze gross auf 1
setze klein auf 100
setze gross auf klein

5.


Wenn angeklickt wird
setze Zähler auf 0
wiederhole 5 mal
ändere Zähler um 1
sage Zähler

6.


Wenn angeklickt wird
setze Zähler auf 0
wiederhole 5 mal
ändere Zähler um 1
sage Zähler für 1 Sekunden

7.

Was passiert in diesem Skript? Kannst du dir eine Anwendung ausdenken, wo man dieses Skript brauchen könnte?

Aufgabe 7.3

Was zeichnet Scratch mit diesem Programm auf die Bühne?

Kannst du das Programm auch so schreiben, dass Scratch das Bild "rückwärts" zeichnet, also mit dem letzten Strich beginnt, dann den zweitletzten Strich zeichnet, und so weiter?

Aufgabe 7.4

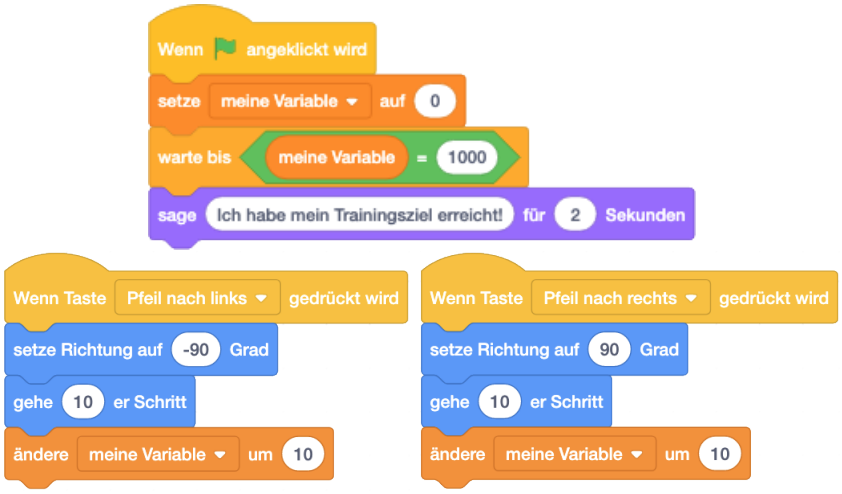
Ergänze dein Pong-Spiel um einen Punktezähler! Zu Beginn hat man 0 Punkte. Jedes Mal, wenn der Ball den Schläger berührt, soll der Punktestand um 1 erhöht werden.

Mit einem Rechtsklick auf die Anzeige der Variable in der Bühne kannst du die Anzeige verstellen. Du kannst auswählen zwischen der Standard-Anzeige, einer grossen Anzeige ohne den Namen der Variable oder einem Schieberegler, mit dem du dem Benutzer deines Projekts Kontrolle über die Variable gibst.

Wenn du den Schieberegler auswählst, kannst du ausserdem nach erneutem Rechtsklick noch auswählen, welches der minimale und der maximale Wert des Reglers sein soll.

7.3 Variablen verwenden

Ergänze das Programm aus dem vorherigen Kapitel zu diesem:



Lass Scratch wieder über die Bühne laufen. Was passiert?

Scratch sagt nach 1000 Schritten “Ich habe mein Trainingsziel erreicht!”.

Die grünen Blöcke sind *Operatoren*. Mit ihnen kannst du einen *Ausdruck* bilden, indem du die Aussparungen mit passenden Blöcken (Wert-Block oder Wahrheits-Block) füllst, und dadurch wieder einen Wert- oder Wahrheits-Block bildest. Damit kannst du mathematische Operationen durchführen, wie zum Beispiel Addition 

, Subtraktion 

, Multiplikation 

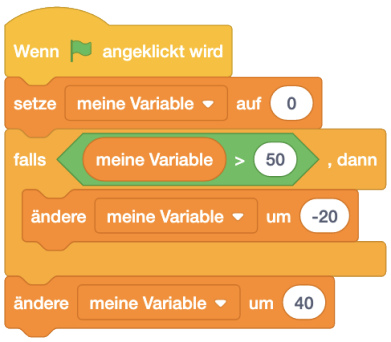
und Division 

. Ausserdem kannst du damit kompliziertere Bedingungen überprüfen.

Aufgabe 7.5

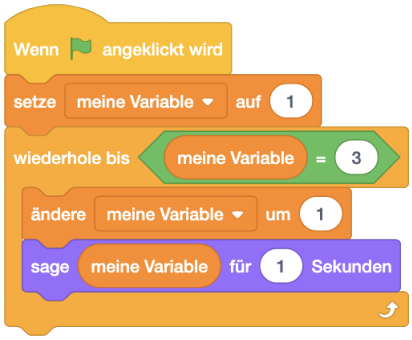
Schreibe für die folgenden Skripte auf, welche Werte die verwendeten Variablen am Ende des Skriptes haben. Stelle dazu am besten eine Tabelle auf mit allen Variablen und ihren Werten. Wenn der Wert einer Variable verändert wird, streichst du den alten Wert durch und schreibst den neuen Wert auf. Schreibe ausserdem auf, was Scratch sagt!

1.



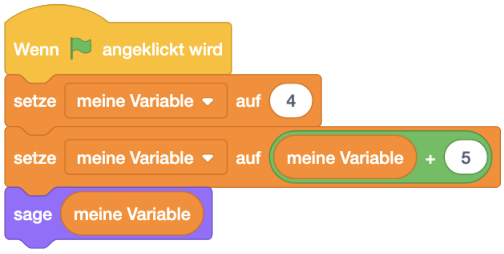
2.

3.



4.

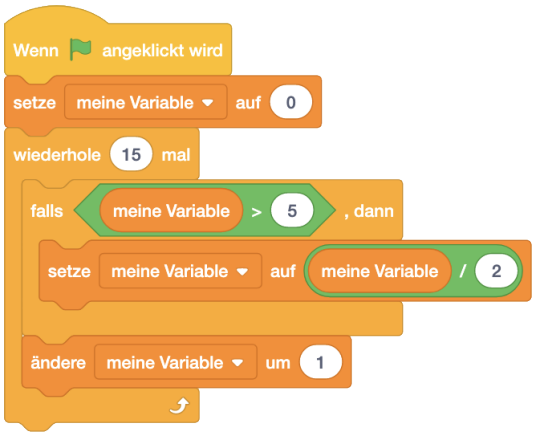
5.



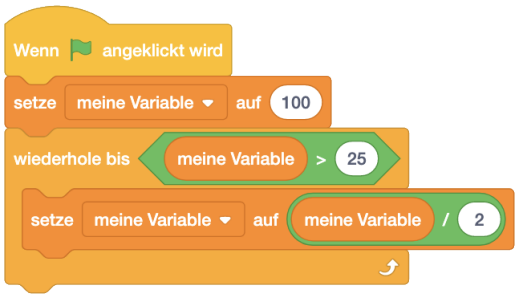
Aufgabe 7.6

Häufig will man mit Computern etwas ausrechnen. Welchen Wert hat die Variable am Ende des jeweiligen Programms?

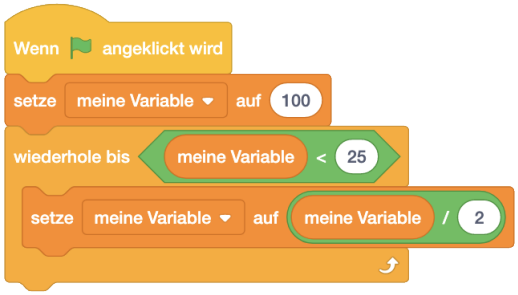
1.



2.



3.



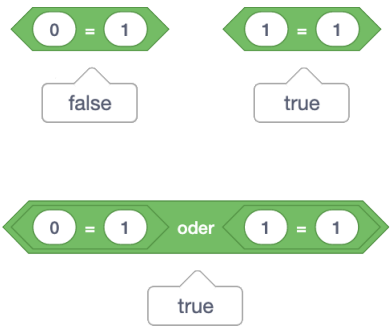
Aufgabe 7.7

In der Logik hat das Wort “oder” eine leicht andere Bedeutung als im alltäglichen Sprachgebrauch. Teste den 

Block und schreibe auf, welche Werte du erhältst:

1. Wahrheits-Block	2. Wahrheits-Block	1. Wahrheits-Block oder 2. Wahrheits-Block
false	false	
false	true	
true	false	
true	true	

Beispiel:



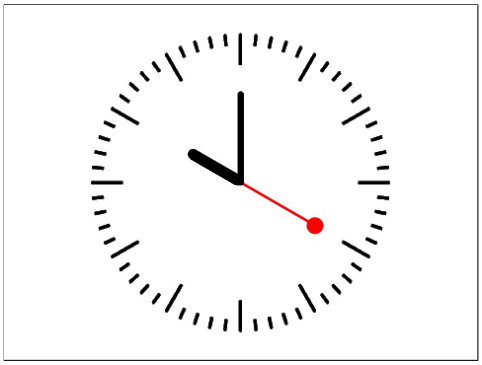
Aufgabe 7.8

Ergänze das Labyrinth aus Aufgabe 6.7 um ein Monster, welches fortlaufend an zufällige Positionen auf der Bühne gleitet. Scratch soll zurück zum Start, wenn das Monster oder die Wand des Labyrinths berührt wird.

Aufgabe 7.9

Programmiere eine Uhr! Zeichne dazu Kostüme für drei Objekte: Einen Stundenzeiger, einen Minutenzeiger und einen Sekundenzeiger. Auf den Bühnenhintergrund malst du das Ziffernblatt der Uhr (oder vielleicht verwendest du deine Lösung aus Aufgabe 5.15?). Programmiere die Zeiger so, dass sie immer in die richtige Richtung zeigen. Benutze dazu den 

Wert-Block mit den entsprechenden Parametern (Stunde, Minute und Sekunde)!

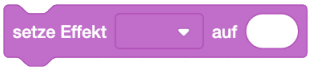


Zusatz:

1. Programmiere einen Wecker: Die genaue Weckzeit soll man mit zwei Variablen festlegen können (Schieberegler-Ansicht!), eine für die Stunde und eine für die Minuten der Weckzeit. Wenn die eingestellte Weckzeit erreicht ist, soll ein Klingelton erklingen.

2. Programmiere die Glockenschläge einer Kirchenglocke. Eine Kirchenglocke erklingt alle Viertelstunde. Die Anzahl der tieferen Glockenschläge gibt die Stunde an, die Anzahl der höheren die Viertelstunden. Die höheren Glockenschläge erklingen dabei zuerst. Die Stunden zählt man dabei im Zwölf-Stunden-System.

   Beispiele:
   15:00 Uhr: 3 tiefe Glockenschläge
   17:20 Uhr: keine Glockenschläge
   09:30 Uhr: 2 hohe Glockenschläge, dann 9 tiefe Glockenschläge
   16:45 Uhr: 3 hohe Glockenschläge, dann 4 tiefe Glockenschläge

   Tipp: Die Tonhöhe des Glockenschlags (Klang "Bell Toll") kannst du mit dem Befehl 

   verändern.

   Wie kannst du testen, ob dein Programm stimmt? Tipp: Benutze Variablen!

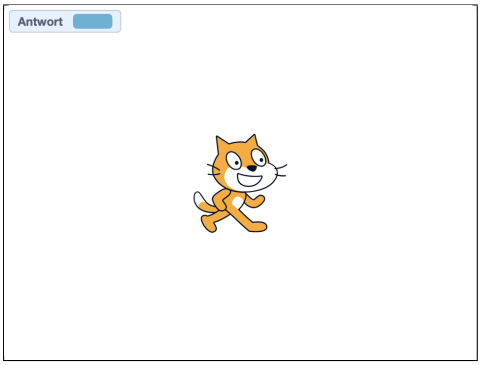
Aufgabe 7.10

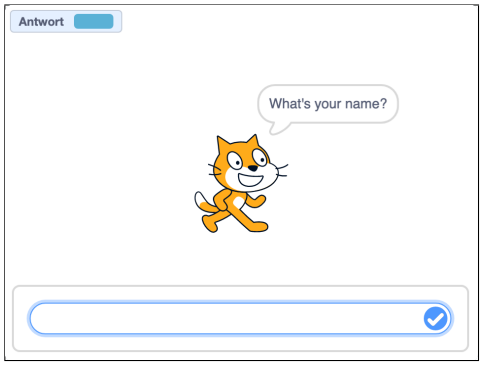
Stell dir vor, die Macher von Scratch würden den Wiederhole-Mehrmals Befehl entfernen, weil er kaputt war. Kannst du den Befehl durch andere Befehle ersetzen? Verwende dazu Variablen und Operatoren. Zeige deine Umsetzung am folgenden Beispiel:

Der 

Befehl ist ein spezieller Befehl, der eine Eingabe erfordert. Das Skript, in dem der Befehl verwendet wird, hält so lange an, bis man etwas eingegeben hat und die Return-Taste drückt oder auf das Häkchen im Eingabefeld klickt. Diese Eingabe wird dann gespeichert. Der Wert-Block 

antwortet mit der letzten Eingabe. Probiere den Befehl und den zugehörigen Wert-Block in der nächsten Aufgabe aus!

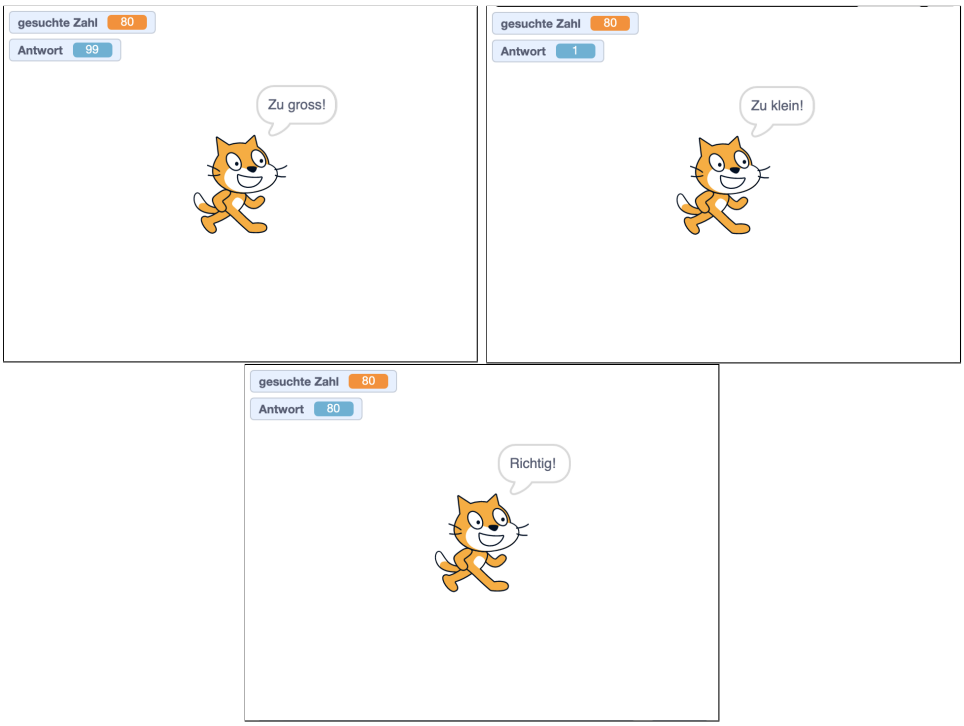





Antwort
What's your name?
Scratch
Antwort Scratch

Aufgabe 7.11

Erstelle ein Programm, in dem man die Zahl erraten soll, die sich Scratch gerade ausdenkt. Bei Klick auf die grüne Flagge soll sich Scratch eine Zufallszahl zwischen 1-100 merken. Dann darf man solange raten (Frage-und-warte Befehl), bis man die richtige Zahl herausgefunden hat oder keine Versuche mehr hat. Scratch soll ausserdem bei jedem Versuch sagen, ob die gesuchte Zahl grösser oder kleiner ist als der Rateversuch.



Die gesuchte Zahl soll natürlich während dem Raten nicht angezeigt werden, sonst

wäre das Spiel viel zu einfach.

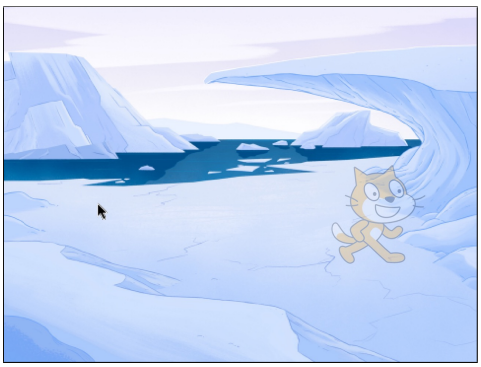
Aufgabe 7.12

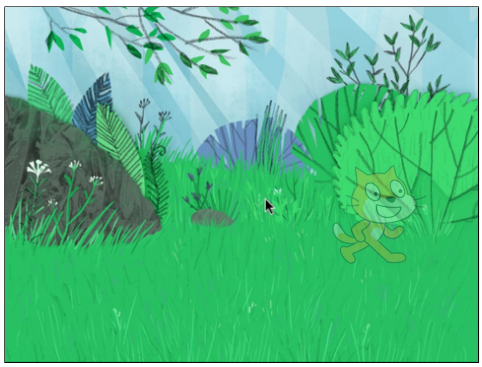
Erstelle ein Programm, in dem der Benutzer Scratch suchen muss. Scratch versteckt sich an einer zufälligen Position auf der Bühne. Bei jedem Mausklick des Benutzers soll Scratch überprüfen, wie weit der Mausklick von Scratch entfernt war und dies in einer Variable speichern, die auf der Bühne sichtbar ist. Wenn der Mausklick nah genug war (zum Beispiel weniger als 20 Bildpunkte entfernt), soll sich Scratch zeigen und "Gefunden!" sagen.

Variante: Statt den Abstand direkt zu zeigen kannst du auch mit dem Hintergrund signalisieren, ob der Mausklick sehr weit entfernt war oder sehr nah.

- Wenn der Mausklick weit weg war (zum Beispiel > 150), soll die Bühne zu einem "kalten" Hintergrund wechseln.
- Wenn der Mausklick nah war (zum Beispiel < 150 aber > 75), soll die Bühne zu einem "warmen" Hintergrund wechseln.
- Wenn der Mausklick ganz nah war (zum Beispiel < 75), soll die Bühne zu einem "heissen" Hintergrund wechseln.
- Wenn der Mausklick Scratch berührt, soll Scratch sich zeigen und "Gefunden!" sagen. Aufgepasst: Versteckte Objekte können nicht angeklickt werden und man sieht nicht, was sie sagen!

Wieso können wir Scratch nicht einfach sagen lassen, wie weit der Mausklick entfernt war?





In diesen Bildern ist Scratch nur durchsichtig, um das Spiel zu erklären. In deinem Projekt soll sich Scratch aber ganz verstecken!

7.4 Eigene Blöcke mit Parametern

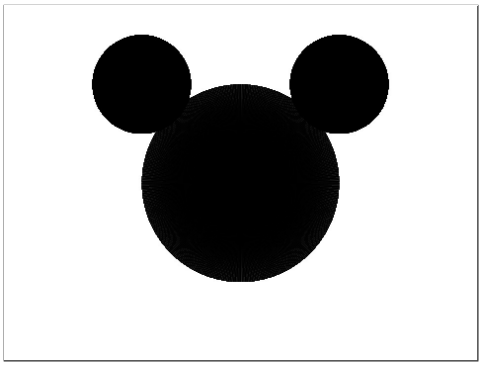
Erstelle das folgende Programm. Erstelle dazu einen eigenen Block und füge ein Eingabefeld für Zahl oder Text hinzu. Benenne dieses Eingabefeld dann wie in dem gezeigten Programm. Den Wertblock des Parameters kannst du einfach aus dem Definitionsblock herausziehen, um ihn in der Definition zu verwenden.

Was zeichnet Scratch?

Wenn wir Scratch den Kreis so zeichnen lassen, dauert es sehr lange. Mit Klick auf “Bearbeiten” oben links und dann auf “Turbo-Modus einschalten” können wir

das beschleunigen.
Man kann aber auch beim Erstellen des eigenen Blocks das Häkchen für "Ohne Bildschirmaktualisierung laufen lassen" setzen. Dann wird nur der eigene Block ganz schnell ausgeführt. Hat man bereits einen eigenen Block erstellt und möchte diese Option noch aktivieren oder deaktivieren, so kann man dies mit Rechtsklick auf den Definitionsblock und dann "Bearbeiten" auswählen.

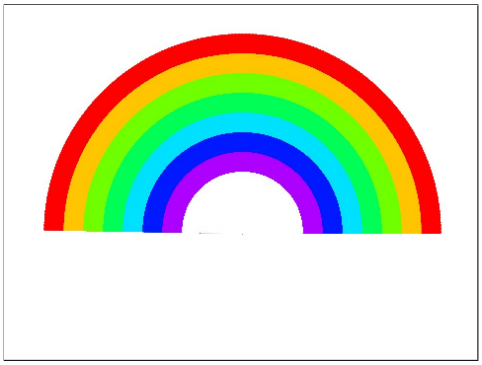
Scratch zeichnet den Umriss einer weltbekannten Maus mit einem grossen und zwei kleineren Kreisen:



In Kapitel 3 haben wir das Konzept eines Parameters kennengelernt. Auch in deinen eigenen Blöcken kannst du Parameter verwenden, um den Block zu verfeinern. Der Parameter in der Definition des eigenen Blocks ist ein *formaler Parameter*. Dieser Parameter kann überall in der Definition des eigenen Blocks wie ein Wertblock verwendet werden. Wenn der Parameter mit einem richtigen Wert ersetzt wird, spricht man von einem *tatsächlichen Parameter*. Ein Block kann mehrere Parameter haben, so wie zum Beispiel der Gehe-Zu Block.

Aufgabe 7.13

Lass Scratch einen Regenbogen zeichnen:



Tipp: Zeichne zuerst einen grossen Halbkreis mit der äussersten Farbe, dann einen etwas kleineren mit der zweitäussersten Farbe und so weiter. Zuletzt malst du noch einen Halbkreis mit weiss.

Aufgabe 7.14

Stell dir vor, das Scratch-Team würde die folgenden Blöcke löschen, weil sie nicht mehr richtig funktionieren. Könntest du die Blöcke als eigenen Block mit der gleichen Funktionsweise neu programmieren? Achtung: Es ist immer nur ein Block kaputt - für jede Teilaufgabe darfst du also alle Blöcke aus den restlichen Teilaufgaben benutzen!

1. 

2. 

- schaffst du das auch ohne den 

Block zu verwenden?

3. 

4. 

- schaffst du das auch ohne den Gehe-Zu Befehl zu verwenden?

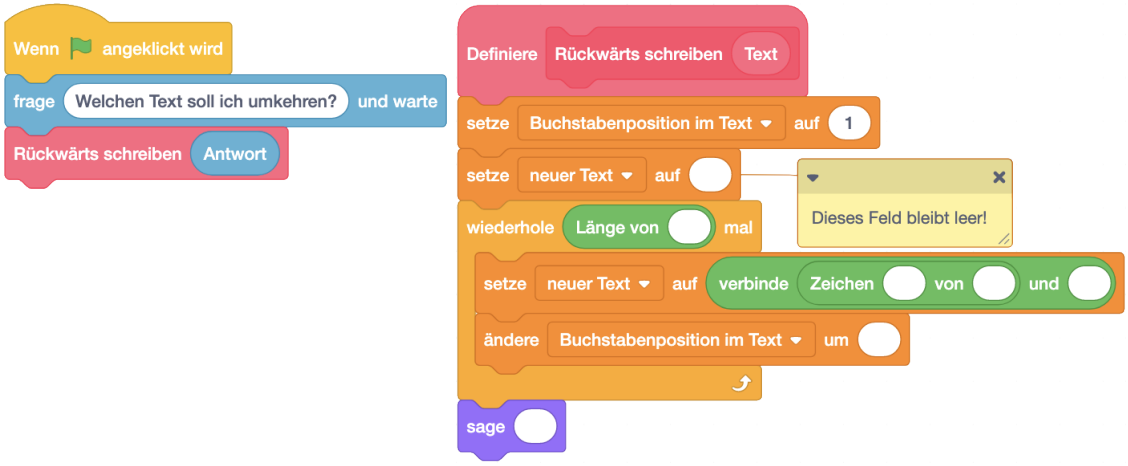
5. 

6. 

Aufgabe 7.15

Ergänze das folgende Programm so, dass du mit deinem eigenen Block Text ver-

schlüsseln kannst! Man soll Text eingeben können (Frage-und-Warte Befehl), den Scratch dann rückwärts sagt.



Der 

Befehl verbindet zwei Texte miteinander (beachte das Leerzeichen am Ende von "apple"):


.

Der 

Wert-Block antwortet mit dem jeweiligen Zeichen des Textes:
.

Der 

Wert-Block antwortet mit der Länge des Textes:

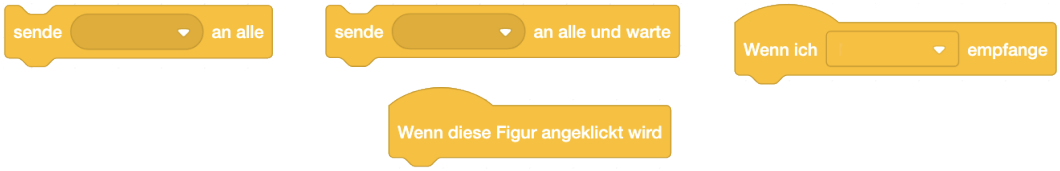

.

Zusatzaufgabe: Erweitere die Verschlüsselung des Textes! Du kannst zum Beispiel jedes 'a' im Text durch ein 'e' ersetzen und umgekehrt. Achte darauf, dass du den Text danach immer noch entschlüsseln kannst!

Kapitel 8

Kommunikation

Hier lernst du, wie Objekte miteinander kommunizieren können. So kannst du auf andere Objekte reagieren.
Die Blöcke, die du dabei kennenlernst:



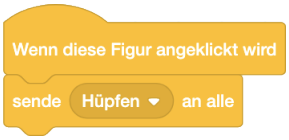
Die Begriffe, die hier neu eingeführt werden:

- Nachricht

8.1 Nachrichten senden und empfangen

Erstelle ein Programm mit Scratch und einem Knopf-Objekt. Erstelle im Programmierbereich des Knopf-Objekts das folgende Skript:
Wähle dazu im Menü des 

Befehls "Neue Nachricht" aus und gib "Hüpfen" als Nachrichtentext ein.



Für Scratch erstellst du dieses Skript:

Was passiert, wenn du den Knopf anklickst?

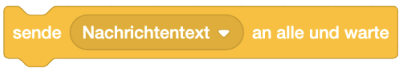
Wenn du den Knopf anklickst hüpft Scratch kurz nach oben. Wenn der Knopf mehrmals kurz nacheinander angeklickt wird, bleibt Scratch oben.

Der Befehl 

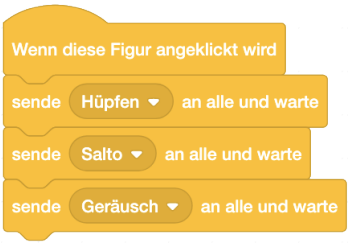
sendet allen (den Objekten und der Bühne) eine *Nachricht* mit dem Nachrichtentext. Die Nachricht startet bei allen Empfängern die Skripte (falls vorhanden) unter dem Hut 

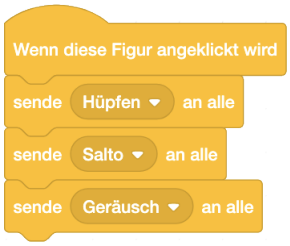
mit dem gleichen Nachrichtentext.

Was ist der Unterschied zwischen dem Befehl 

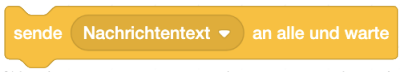
und dem Befehl 

? Probiere es mit dem Programm unten aus:





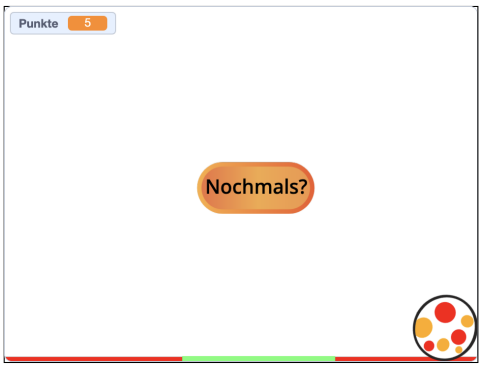
Der Befehl 

wartet die Reaktion der Empfänger auf die Nachricht nicht ab; direkt nach dem Senden geht es mit dem nächsten Befehl im Skript weiter. Der Befehl 

hält sein Skript an, bis *alle* Empfänger mit ihren Skripten unter den zugehörigen Hüten fertig sind. Achtung: Wenn das Empfängerskript einen Wiederhole-Fortlaufend Befehl enthält, kann es sein, dass das Skript des Senders nie fertig ausgeführt wird!

Aufgabe 8.1

Ergänze dein Pong-Spiel um eine "Nochmals" Taste. Die Taste soll erst erscheinen, wenn der Ball den Boden berührt. Wenn die Taste angeklickt wird, soll das Spiel wieder von vorne beginnen. Die Taste soll dann wieder verschwinden.



Aufgabe 8.2

Komponiere ein Musikstück! Scratch soll dabei anderen Instrumenten eine Nachricht schicken, wenn sie einsetzen sollen. Die Instrumente sollen sich bewegen, wenn sie Musik spielen!

Aufgabe 8.3

Such dir drei Figuren aus. Diese Figuren stellen sich nacheinander vor. Scratch moderiert: Wenn die erste Figur fertig ist, kommt die nächste dran.

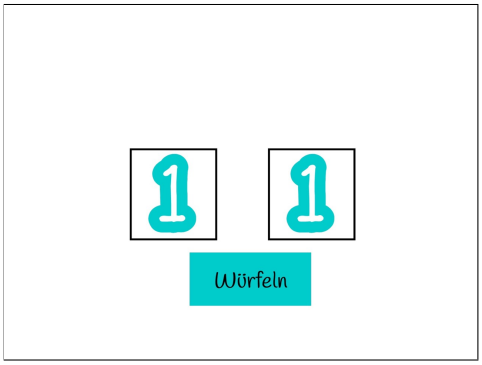
Aufgabe 8.4

Verändere das Einstiegsbeispiel so dass Scratch von überall auf der Bühne hüpfen kann.
Zusatz: Wie kannst du erreichen, dass Scratch nicht immer weiter hochfliegt, wenn der Knopf mehrmals hintereinander angeklickt wird? Tipp: Verwende eine Variable in der du speicherst ob Scratch gerade hüpft!

Aufgabe 8.5

Erstelle ein Programm mit drei Figuren: Zwei Würfel und eine Taste. Für die Würfel kannst du die Zahlen-Kostüme verwenden und sie mit einem Quadrat umrahmen. Wenn die Taste angeklickt wird (

), sollen die Würfel zu einem zufälligen Kostüm (1 bis 6) wechseln. Überlege dir eine Bedingung, unter der man gewinnt, zum Beispiel wenn beide Würfel die gleiche Zahl zeigen oder wenn die Summe ihrer Augenzahlen grösser ist als acht. Lasse dann einen Gewinn-Klang spielen und Ballons erscheinen (Hintergrund "Party")! Du kannst das Würfeln auch etwas realistischer gestalten, indem du den Würfel mehrmals zu einem zufälligen Kostüm wechseln lässt, bevor er "anhält".



Zusatz-Aufgaben:

1. Füge einen weiteren Würfel hinzu (Rechtsklick und Duplizieren) und ändere die Gewinn-Bedingung, so dass sie von allen drei Würfeln abhängt.

2. Beim Klicken der Taste soll abwechselnd einmal der linke, dann einmal der rechte Würfel "würfeln".

3. Herausforderung: Programmiere das Würfelspiel “Die böse Eins”! Es wird in Runden gespielt. Wenn man am Zug ist, darf man entscheiden, ob man würfeln will oder seinen Zug beendet. Beim Würfeln werden die erzielten Augenzahlen aus der Runde zusammengezählt, ausser es wird eine Eins gewürfelt - dann verliert man alle Punkte aus der Runde. Wenn man den Zug beendet, wird die jetzige Punktzahl gespeichert und die andere Person ist am Zug. Wer zuerst 100 Punkte hat, hat gewonnen.

Quellenangaben

Quellenverzeichnis

Dieses Handbuch ist eine Adaption von “Programmieren für Kinder mit Scratch” von Bernd Gärtner, siehe https://kinderlabor.ch/informatik-fuer-kinder/programmieren-mit-scr
Dieses Ursprungsmaterial wurde inspiriert durch das Lehrmittel “Einführung in die Programmierung mit LOGO: Lehrbuch für Unterricht und Selbststudium”.
Aufgabe 6.5 wurde inspiriert durch das Scratch-Projekt “If-Else Dragon - Commands Example” von “dill1233” (https://scratch.mit.edu/projects/1214042/).
Aufgabe 6.3 wurde inspiriert durch das Scratch-Projekt “If/else Maze Challenge [starter]” von “janeemac” (https://scratch.mit.edu/projects/143662830/).
Aufgabe 5.2 wurde inspiriert durch “Measuring student learning in introductory block-based programming: Examining misconceptions of loops, variables, and boolean logic” von Shuchi Grover und Satabdi Basu, erschienen 2017 in “Proceedings of the 2017 ACM SIGCSE technical symposium on computer science education”, Seiten 267–272.
Aufgaben 7.2 und 7.5 wurde inspiriert durch “Programming misconceptions for school students” von Alaaeddin Swidan, Felienne Hermans und Marileen Smit, erschienen 2018 in “Proceedings of the 2018 ACM Conference on International Computing Education Research”, Seiten 151 – 159.
Aufgabe 8.3 wurde inspiriert durch das Scratch-Projekt “The ’3 Favorite Things’ Pass-It-On Project!” von “karenb” (https://scratch.mit.edu/projects/236312698/).
Aufgabe 8.5 wurde inspiriert durch das Scratch-Projekt “Three of a Kind” von “karenb” (https://scratch.mit.edu/projects/237039971/).

Bildnachweis

Skripte und Bühnenbilder sind bearbeitete Screenshots von Scratch-Programmen. Scratch ist ein Projekt der „Lifelong Kindergarten Group“ des MIT Media Lab. Weitere Informationen gibt es unter scratch.mit.edu, LLK.media.mit.edu und media.mit.edu.
Icons für verschiedene Aufgabentypen (Seite 4 und folgende):
Laptop, Papier und Stift, Sprechblase, Lupe: janjf93, pixabay.com