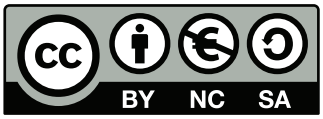
Programmieren für Kinder mit

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



Inhaltsverzeichnis

Hinweise für Lehrpersonen

0.1 Vorwort

Das Kinderlabor bietet seit vielen Jahren Unterrichtsmaterial zum Programmieren mit Scratch an:

https://kinderlabor.ch/informatik-fuer-kinder/programmieren-mit-scratch/.

Das Material ist aus Scratch-Kursen des Kinderlabors an Schulen hervorgegangen und erfreut sich im deutschsprachigen Raum grosser Beliebtheit. Dieses Buch ist eine für Scratch 3.0 aktualisierte und erweiterte Ausgabe dieses Materials, mit vielen zusätzlichen Innformationen für Lehrpersonen.
Je nach Alter und Fähigkeiten der Kinder eignet sich das Material für einen Klassen- oder Halbklassenkurs über ein Semester bis zu einem Jahr, bei zwei Lektionen pro Woche. Wir empfehlen das Material für Kinder der Klassenstufen 5 und 6. Es kann aber auch bei älteren Schülerinnen und Schülern sowie im Rahmen der Begabtenförderung erfolgreich eingesetzt werden. Ein regulärer Einsatz in den Klassen 3 und 4 empfiehlt sich nicht; das Material ist keine Anleitung zum Entwickeln von Spielen mit Scratch, sondern eine Anleitung zum "richtigen" Programmieren, auch unter Benutzung altersgerechter mathematischer Konzepte, wobei der spielerische Aspekt nicht zu kurz kommt.

Seit der Ersterstellung der Unterlagen vor ungefähr 10 Jahren hat Scratch sich gewandelt. Damals (Scratch 1.4) musste die Programmierumgebung noch separat heruntergeladen werden. Seit Scratch 2.0 ist die Programmierumgebung in die Scratch-Webseite integriert, was das Teilen und Verändern von Projekten wesentlich vereinfacht hat. Scratch 3.0 schliesslich kann sogar auf mobilen Geräten benutzt werden, erforderlich ist nur noch ein Browser und eine Internetverbindung.
Die Programmiersprache Scratch selbst hat sich in dieser Zeit ebenfalls verändert, allerdings auf den ersten Blick nicht dramatisch. Alles, was in den bisherigen Unterlagen des Kinderlabors behandelt wird, funktioniert auch in Scratch 3.0 noch gleich oder sehr ähnlich, weshalb diese Unterlagen auch immer noch oft heruntergeladen und benutzt werden. Seit Scratch 2.0 das Konzept der eigenen Blöcke eingeführt hat, ist es aber in Scratch auch möglich, neue Befehle zu definieren und diese in eigenen Programmen zu benutzen. Diese Möglichkeit der Spracherweiterung ist ganz essenziell für eine gute Programmiersprache, denn nur auf diese Weise können grössere Programme sinnvoll und lesbar strukturiert werden. Durch Einführung der eigenen Blöcke hat Scratch seinen bisherigen Nachteil gegenüber der klassischen Kinder-Programmiersprache LOGO wettgemacht.

Nun war es Zeit für eine Aktualisierung der Unterlagen. Bereits in Scratch 1.4 konnte mit Hilfe von Nachrichten indirekt eine primitive Form der Spracherweiterung realisiert werden, allerdings sind die eigenen Blöcke hier die wesentlich elegantere und "richtige"

Lösung. Diese Lösung sollte in allen aktuellen Scratch-Unterrichtsmaterialien eingeführt werden.

Bei der Aktualisierung der Materialien haben wir auch den Umfang deutlich erweitert. Neu werden Variablen, Operatoren und Nachrichten behandelt, und die erst seit Scratch 2.0 mögliche Strukturierung von Programmen mit Hilfe von eigenen Blöcke wird eingeführt. Auch zu den bisherigen Kapiteln gibt es viele zusätzliche Erklärungen und Aufgaben.
Die Unterlagen lagen bisher in Form von Folien vor, die auf engem Raum alles wesentliche in kurzer und knapper Form darstellen. Diese Folien waren so sehr kompakt, jedoch konnten sie nicht gut gewartet und erweitert werden. Für Kinder und Lehrpersonen war es auch nicht immer leicht, sie zu lesen. Das konnten wir nun verbessern: Die neuen Unterlagen sind in Form eines Buchs aufgebaut, wobei wir die erprobten Symbole für Programme, Erklärungen, Begriffe und Hinweise beibehalten haben.
Das Material für Schülerinnen und Schüler (Lektionen und Aufgaben) liegt neu in einem Dokument vor, wodurch sich das "Springen" zwischen zwei Dokumenten erübrigt. Neu ist auch diese Version für Lehrpersonen, die zusätzlich zu den Lösungen auch viele diadaktische Hinweise, Tipps und Tricks enthält.

Diese Neuauflage wurde im Herbstsemester 2019 an der ETH Zürich von Katharina Daun als Begleitmaterial zu ihrer Masterarbeit *Teaching programming with Scratch in grades 5 and 6* geschrieben und zusammen mit Bernd Gärtner im Sommer 2020 fertiggestellt.

Wir freuen uns sehr über Kommentare und Rückmeldungen unter bg@kinderlabor.ch!
Nun wünschen wir Ihnen viele interessante Programmier-Stunden und spannende Aha-Erlebnisse - legen Sie los und tauchen Sie ein in unser aktualisiertes Unterrichtsmaterial mit Scratch!

Bernd Gärtner, Juli 2020

0.2 Einleitung

Schön, dass Sie sich für unser Handbuch zur Programmierung mit Scratch interessieren. Da Informatikunterricht in der Schweiz ein relativ neues Thema ist, möchten wir Ihnen in diesem Kapitel einige Hinweise zur Verwendung dieses Handbuches geben. Einige dieser Tipps sind Ihnen vielleicht schon lange bekannt, doch hoffentlich sind auch ein paar neue hilfreiche Informationen mit dabei. Es werden auch im Laufe des Lehrmittels weitere Tipps und Tricks zur Durchführung gegeben. Diese sind jeweils mit einer Umrandung und farblich gekennzeichnet:

Es wird ebenso angemerkt, wenn ein Thema eher den Medien-Aspekt von Scratch anspricht. Scratch kann zum Beispiel durchaus auch benutzt werden, um eine Präsentation zu erstellen oder das Thema Copyright zu behandeln. Wenn man diese zwei Bereiche aber lieber klar trennen will, lässt man die jeweiligen Teile aus.

0.3 Informatik ohne Computer?

Die meisten Übungen aus diesem Handbuch sind für die Umsetzung in Scratch konzipiert. Es gibt aber auch einige Übungen, bei denen die Lernenden miteinander diskutieren sollen und Aufgaben, die mit Stift und Papier gelöst werden sollen. Ausserdem lassen sich besonders die Anfangsaufgaben auch gut mit Scratch-Blöcken aus Papier lösen. Dadurch sind die Lernenden zu Beginn nicht zu stark von den vielen verschiedenen Kostümen und Klängen abgelenkt. Welche Aufgaben sich dazu eignen ist auch in den einzelnen Kapiteln nochmals beschrieben.
Ein guter Mix aus *unplugged* Aufgaben (also ohne Computereinsatz) und Aufgaben mit Computereinsatz spricht verschiedene Lerntypen an und hilft auch besonders Kindern, die noch grossen Respekt vor der Informatik haben oder noch nicht so geschickt mit dem Computer umgehen können. Ausserdem kann man so gut etwas Abwechslung in den Unterricht bringen.
Es ist ausserdem hilfreich, im Unterricht Theorie und Praxis klar zu trennen - wenn möglich auch räumlich. Der Theorie-Teil kann zum Beispiel im vorderen Teil des Klassenzimmers beginnen und beim Übergang zur Praxis setzen sich die Lernenden an die Computer im hinteren Teil des Klassenzimmers. Wenn die Lernenden einmal am Computer sitzen, ist es schwierig, ihre Aufmerksamkeit zurückzugewinnen, um etwas erklären zu können. Daher sollte man darauf verzichten, die Klasse zu unterbrechen und die Frage klarzustellen, auch wenn sie mehrfach auftaucht. Stattdessen kann eine der Strategien angewandt werden, die unten vorgestellt werden.

Die Einführungsaufgaben zu jedem Abschnitt können meistens von der Lehrperson vorgezeigt werden. Die Lernenden können dabei vor der Durchführung erraten, was bei dem Einführungsbeispiel passieren wird. Durch die Theorie-Aufgaben wird das neue Wissen erst gefestigt und allfällige Missverständnisse können aufgeklärt werden, bevor die Lernenden beginnen, selbst am Computer zu programmieren.

0.4 Unterrichts-Strategien

Der Umgang mit dem Computer und mit Scratch ist neu und ungewohnt, und die Kinder haben oft viele Fragen. Es lohnt sich daher, sich eine Strategie zurechtzulegen, damit alle Kinder gut betreut sind und niemand zu lange auf eine Antwort warten muss.
Die folgenden Strategien sind aus verschiedenen Texten zusammengetragen und sollen als Hilfe dienen, damit die Lehrperson nicht überfordert wird mit Anfragen. [1]

Hände auf den Rücken! Zu Beginn reizt es bestimmt, den Fehler eines Kindes mit ein paar Klicks zu korrigieren, damit man Zeit hat für alle anderen. Dadurch werden aber die grundlegenden Probleme nicht verstanden und treten daher immer wieder auf. Die Lehrperson stellt daher nur gezielt Fragen, die das Kind zur Lösung leiten sollen.

Hilfe zur Selbstreflexion Wenn Lernende um Hilfe bitten, sollen sie jeweils die folgenden Fragen beantworten:

1. Was machst du gerade?
2. Wieso machst du das? / Was willst du erreichen?
3. Hilft das?

Durch die wiederholte Fragestellung werden die Lernenden dazu angeregt, sich diese Fragen selbst zu stellen und ihr Verhalten bei der Problemlösung zu reflektieren.

FAQ-Liste Häufig gestellte Fragen werden mit den zugehörigen Lösungen in einer Liste gesammelt und an einer Tafel aufgehängt.

Gegenseitige Hilfe Die Lernenden schreiben ihr Problem am Ende der Stunde auf und die Lehrperson sammelt die Probleme ein. Zu Beginn der nächsten Stunde

[1] "Informatikunterricht planen und durchführen" - Buch von Werner Hartmann, Michael Näf und Raimond Reichert
"Issues of structure and agency in computational creation in computational creation, in and out of school" - Doktorarbeit von Karen A. Brennan
"Programming; What it is and how to teach it" - Vortrag (verfügbar auf Youtube) von Amy J. Ko

suchen sich die Lernenden ein Problem eines Klassenmitgliedes aus und versuchen, das Problem zu verstehen, zu reproduzieren und allenfalls sogar zu lösen. Mit dieser Methode lernen die Lernenden, ein Problem klar zu formulieren, sich in andere hineinzuversetzen und (Programmier-) Fehler zu finden.

Support-Warteschlange Die Lernenden schreiben ihr Problem auf und hängen es an eine Liste an der Wandtafel, die von der Lehrperson nach und nach abgearbeitet wird.

Frage Drei Die Lernenden sollen zuerst drei Klassenmitglieder um Hilfe bitten, bevor die Lehrperson gefragt wird.

Mut zur Lücke Man findet nicht zu jedem Problem immer sofort eine Lösung. Das ist vollkommen in Ordnung! Seien Sie neugierig, mit den Lernenden gemeinsam neue Ansätze zu entdecken.

0.5 Gruppenarbeit

Das Bild der Informatik als Einzelarbeit ist weit verbreitet, obwohl das Gegenteil der Fall ist. Daher sollte wenn möglich auch im Unterricht Gruppenarbeit eingebaut werden. Die Arbeit an einem gemeinsamen Projekt ist eine sehr wichtige Unterrichtsmethode in der Informatik, da man so gut verschiedene Kompetenzen fördern kann, die in der Informatik wichtig sind.
In Scratch kann Gruppenarbeit auf verschiedene Arten umgesetzt werden. Die Lernenden können zum Beispiel in Zweiergruppen zusammen an einem Computer arbeiten. Nach dem Prinzip der *Paarprogrammierung* programmiert ein Kind, während das andere Kind genau aufpasst, mitdenkt, und auf allfällige Probleme aufmerksam macht. Es sollte dabei darauf geachtet werden, dass die Rollen regelmässig gewechselt werden. Die Lernenden können aber auch die Arbeit zuerst untereinander aufteilen, wobei jedes Kind das Programmieren von einem (oder auch mehreren) Objekten übernimmt. Nach der Programmierung werden diese Objekte dann heruntergeladen und im Gruppenprojekt zusammengefügt. Kinder die schneller fertig sind, können ihre Klassenmitglieder im Stil der Paarprogrammierung unterstützen. Eine weitere Möglichkeit ist es, Projekte zu *remixen*. Bei dieser Variante arbeiten die Gruppenmitglieder nicht zeitgleich an einem Projekt, sondern nacheinander. Dies funktioniert vor allem auch mit mehreren Projekten. Wenn ein Kind mit dem ersten Teil des Projektes fertig ist, kann das nächste Gruppenmitglied das Projekt remixen und erweitern.

0.6 Universelle Aufgaben

An dieser Stelle möchten wir ein paar Aufgaben vorstellen, die sich nach dem ersten Kapitel grundsätzlich jederzeit einfügen lassen. Zu diesen Aufgaben gibt es keine Lö-

sungsvorschläge, da die Aufgaben sich sehr individuell bearbeiten lassen.

Aufgabe 0.1

Stelle einem Klassenmitglied eine Aufgabe, die es mit Scratch lösen soll. Formuliere diese Aufgabe schriftlich. Erstelle selbst eine Lösung für diese Aufgabe mit Scratch, aber zeige sie deinem Gegenüber nicht. Tauscht nun eure Aufgaben und versucht jeweils die Aufgabe des anderen zu lösen. Wenn dein Gegenüber deine Aufgabe gelöst hat, stellt euch gegenseitig die Lösungen vor und vergleicht sie hinsichtlich der fehlerfreien Lösung der Aufgabe, der Benutzerfreundlichkeit, der Effizienz und der Eleganz der Skripte.

Aufgabe 0.2

Wähle ein Projekt eines Klassenmitgliedes aus, das dich inspiriert und erstelle einen Remix davon.

Diese Aufgabe eignet sich, um das Thema Copyright zu besprechen. Projekte, die man auf Scratch veröffentlicht, kann grundsätzlich jeder "remixen", also das Programm kopieren und erweitern oder verändern. Somit eignet sich diese Aufgabe auch besonders, wenn man die Lernenden Klänge oder Bilder aus dem Internet benutzen lässt, um eigene Objekte in Scratch zu erstellen. Sie erfahren so aus erster Hand, wie es ist, wenn andere ihre Werke benutzen und verändern.

Aufgabe 0.3

Erstelle eine animierte Festtagskarte (z.B. für Halloween, Ramadan, Ostern, Neujahr, Geburtstag...) für deine Familie oder Freunde. Wenn du beim Erstellen der Karte etwas Spannendes herausgefunden hast oder Mühe hattest, etwas zu programmieren, teile es mit der Klasse. Erkläre, was das Problem war (und wie du es gelöst hast).

Diese Aufgabe kann auch gut als Gruppenarbeit durchgeführt werden, bei der sich die Lernenden die Arbeit untereinander aufteilen. Dies bietet eine gute Gelegenheit, etwas "Projekt-Luft" zu schnuppern, ohne allzu viel Zeit zu brauchen.

Aufgabe 0.4

Erstelle ein Tutorial mit Scratch! Das kann eine Bastelanleitung sein, ein Kochre-

zept oder auch eine Anleitung, wie man etwas in Scratch programmiert.

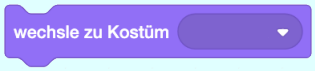
Aufgabe 0.5

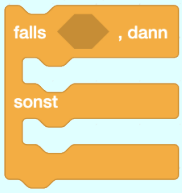
Wo findest du Hilfe, wenn du beim Programmieren nicht weiterkommst? Vergleiche verschiedene Informationsquellen: Welche findest du am hilfreichsten? Sind die Informationen glaubwürdig? Wie aktuell sind die Informationen? Sind sie vielleicht veraltet?
Mögliche Informationsquellen sind das Scratch Wiki (https://de.scratch-wiki.info/wiki/Hauptseite), das Scratch Forum (https://scratch.mit.edu/discuss/13/), Videos auf Youtube, Tutorials vom Scratch-Team, Tutorials von anderen Scratchern, Bücher, die Lehrperson, Eltern, Klassenmitglieder und viele mehr.

Diese Aufgabe bezieht sich auf Medienkompetenzen. Wenn man sich aufs Programmieren konzentrieren will, kann man sie weglassen.

0.7 Zusätzliches Material

Unter https://scratchblocks.github.io/#?style=scratch3&lang=de&script= kann man Skripte als Bild-Dateien speichern. Die Befehle entsprechen dabei grösstenteils dem Text, der auf dem Block steht. So wird zum Beispiel “gehe ()er Schritt” zu 

, “wechsle zu Kostüm [v]” wird zu 

, und einen 

Befehl erreicht man mit

```
falls <>, dann

sonst

end
```

Unter https://en.scratch-wiki.info/wiki/Block_Plugin/Syntax#Custom_Blocks gibt es dazu noch mehr Informationen (auf Englisch).
Die Lösungen zu den Aufgaben sind auch online verfügbar:
Kapitel 3: https://scratch.mit.edu/studios/26899341/
Kapitel 4: https://scratch.mit.edu/studios/27202421/
Kapitel 5: https://scratch.mit.edu/studios/27168352/

Kapitel 6: https://scratch.mit.edu/studios/27225638/
Kapitel 7: https://scratch.mit.edu/studios/27230131/
Kapitel 8: https://scratch.mit.edu/studios/27222532/

0.8 Was wird nicht behandelt?

Dieses Lernmaterial geht auf die folgenden Themen nicht genauer ein, obwohl sie mit Scratch behandelt werden könnten:

1. Listen (Arrays)
2. Klonen
3. Rekursion

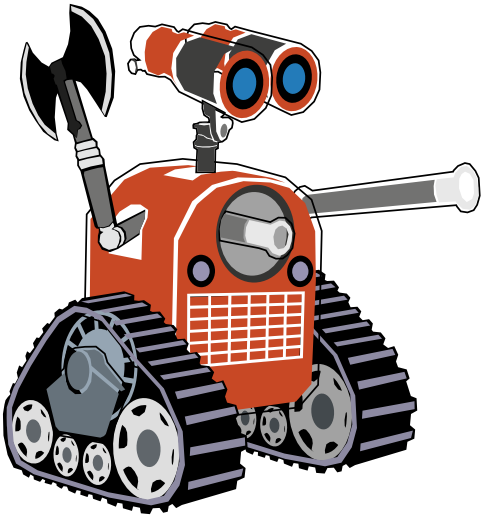
Kapitel 1

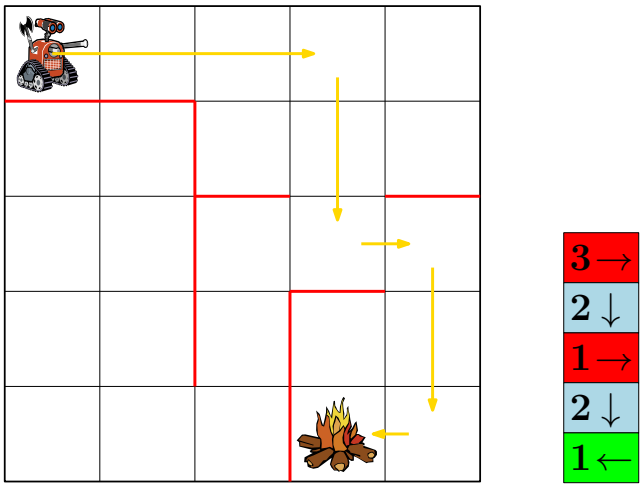
Programmieren

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

Es gibt verschiedene Möglichkeiten, wie man diese Einführung in die Welt der Programmierung gestalten kann. Eine Möglichkeit besteht darin, die Lernenden die Regeln der Aufgabe anhand der Beispiele unter https://scratch.mit.edu/projects/16532746 selbst entdecken zu lassen - Was ist das Ziel des Spiels, was bedeuten die roten Linien, wie steuert man den Roboter? Man kann die Einführung aber auch unplugged durchführen, indem man das Gitter mit Malerkrepp auf dem Boden darstellt. Die Mauern können z.B. mit Stiften hingelegt werden, so lassen sich die Mauern auch leicht wieder entfernen, wenn der Roboter die Axt einsetzt. Das Feuer wird als Bild ausgedruckt und hingelegt. Die Befehle für die folgenden Aufgaben können entweder auf Papier ausgedruckt und untereinander gelegt werden oder mit Lego-Steinen aufeinandergestapelt werden. Wichtig ist, dass die Befehle von oben nach unten ausgeführt werden, denn so funktioniert das später auch in Scratch. Ausserdem muss man aufpassen, dass alle Kinder auf einer Seite des Feldes stehen können, sonst kann die Ausführung von links/rechts und oben/unten verwirrend sein. Man kann die Aufgaben aber auch einfach am Beamer zeigen und mit der ganzen Klass besprechen.





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.

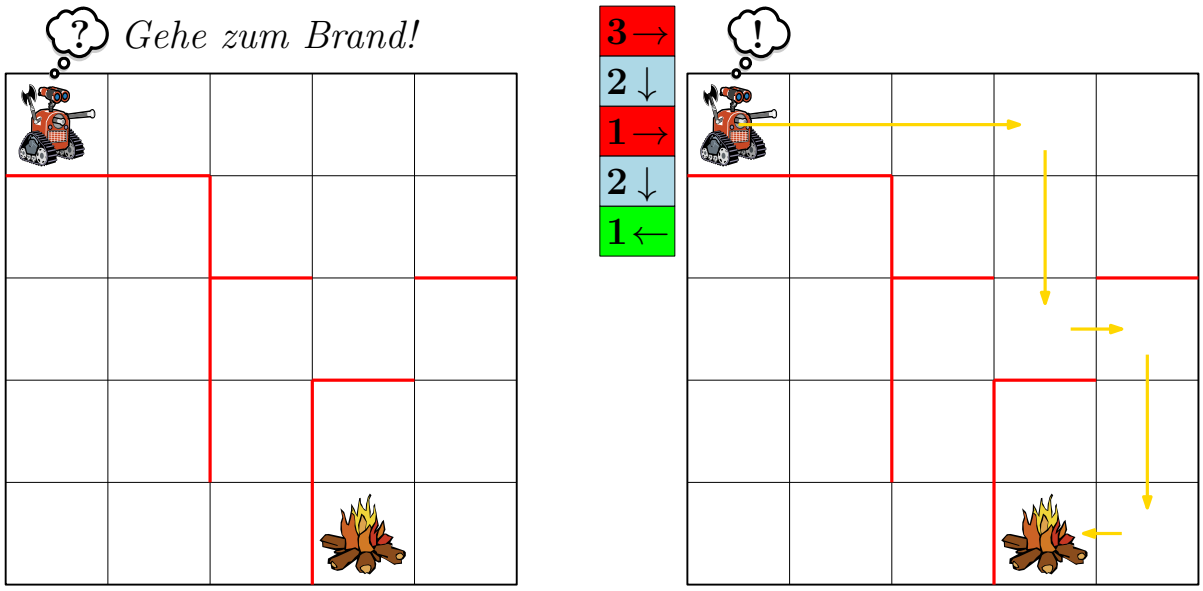


Abbildung 1.1: Computer verstehen unsere Sprache nicht (links), deshalb müssen wir ihre Sprache lernen, wenn wir wollen, dass sie etwas für uns tun (rechts). Die Sprache des Löschroboters sprechen wir mit bunten Blöcken, die von oben nach unten gelesen werden. Kannst du herausfinden, was die bunten Blöcke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

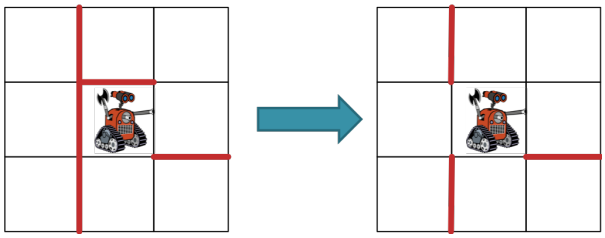
In vielen Haushaltsgeräten ist ein Computer versteckt, den du programmieren kannst. Nenne ein paar Beispiele!

1.3 Programmiersprachen

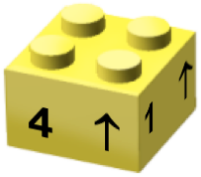
Eine *Programmiersprache* ist eine Sprache, die der Computer versteht. Sie besteht aus Befehlen, die du im Umgang mit dem Computer benutzen kannst.

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.

1→ 2→ 3→ 4→	Gehe 1, 2, 3, 4 Kästchen nach rechts
1↓ 2↓ 3↓ 4↓	Gehe 1, 2, 3, 4 Kästchen nach unten
1← 2← 3← 4←	Gehe 1, 2, 3, 4 Kästchen nach links
1↑ 2↑ 3↑ 4↑	Gehe 1, 2, 3, 4 Kästchen nach oben
A	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.

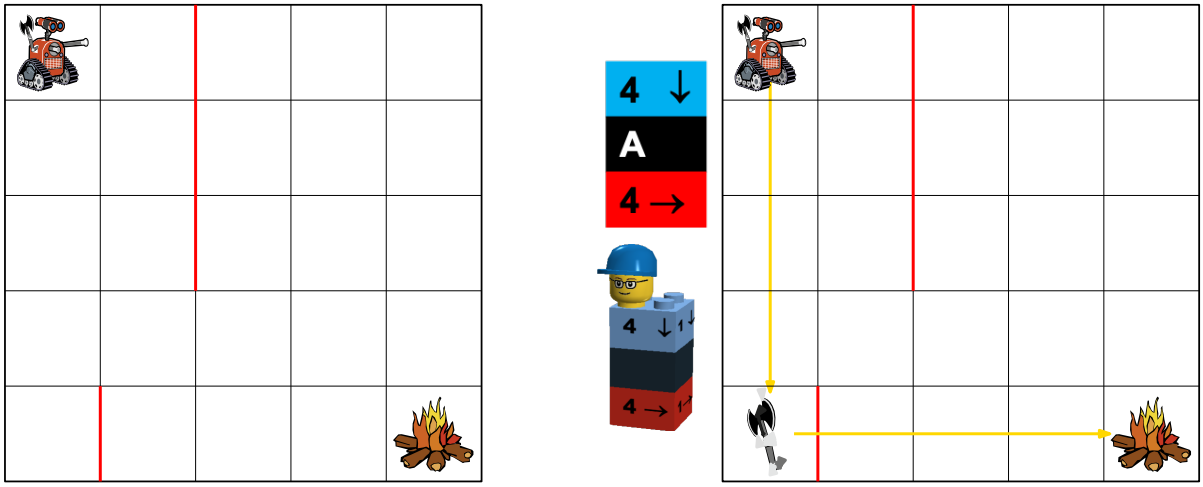


Abbildung 1.2: Links: Die Ausgangssituation; Mitte: das Programm – Befehle werden von oben nach unten gelesen; Rechts: Ausführung des Programms durch den Löschroboter.

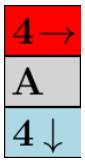
Nicht jedes Programm ist korrekt. Es gibt Programme, die den Löschroboter nicht zum Brand bringen, aber auch Programme, bei denen er kaputt geht (wenn er über

den Rand des Gitters oder gegen eine Wand fährt).

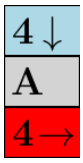
Aufgabe 1.6

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)

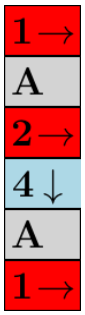


b)



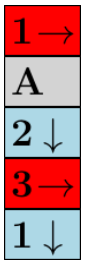
c)

d)



e)

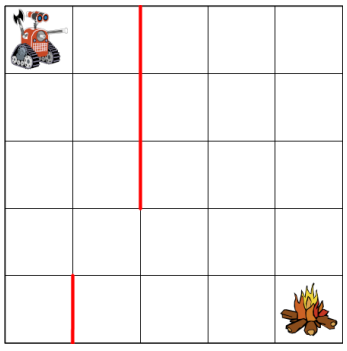
f)



Hier können die Lernenden zuerst abstimmen, ob der Roboter ans Ziel kommt, kaputt geht oder nicht ans Ziel kommt. Dann darf ein Kind Roboter spielen und ein anderes liest die Befehle vor, die das Roboter-Kind dann ausführt.

Aufgabe 1.7

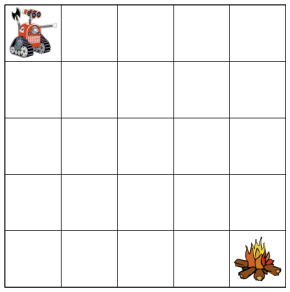
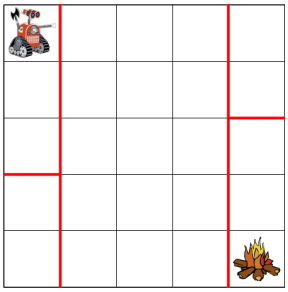
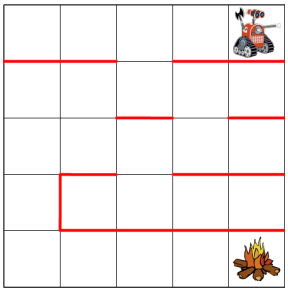
Schreibe alle Programme auf, die den Löschroboter in der Ausgangssituation unten mit drei Befehlen zum Brand steuern!



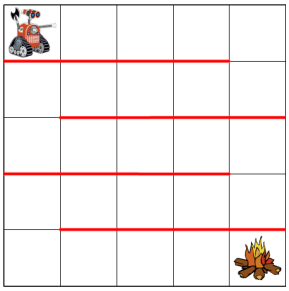
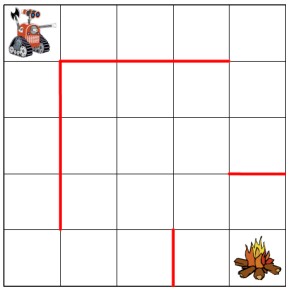
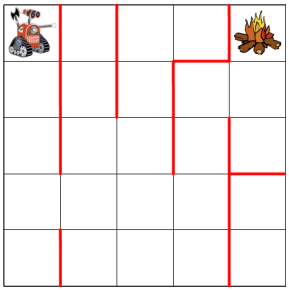
Aufgabe 1.8

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!

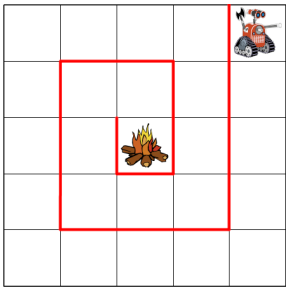
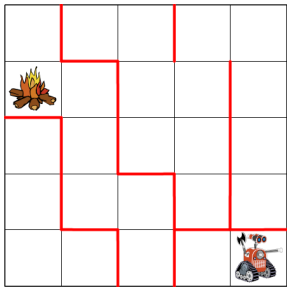
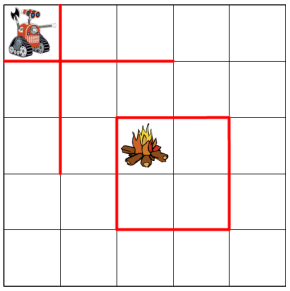
a) b) c)

d) e) f)

g) h) i)

Kapitel 2

Erste Schritte mit Scratch

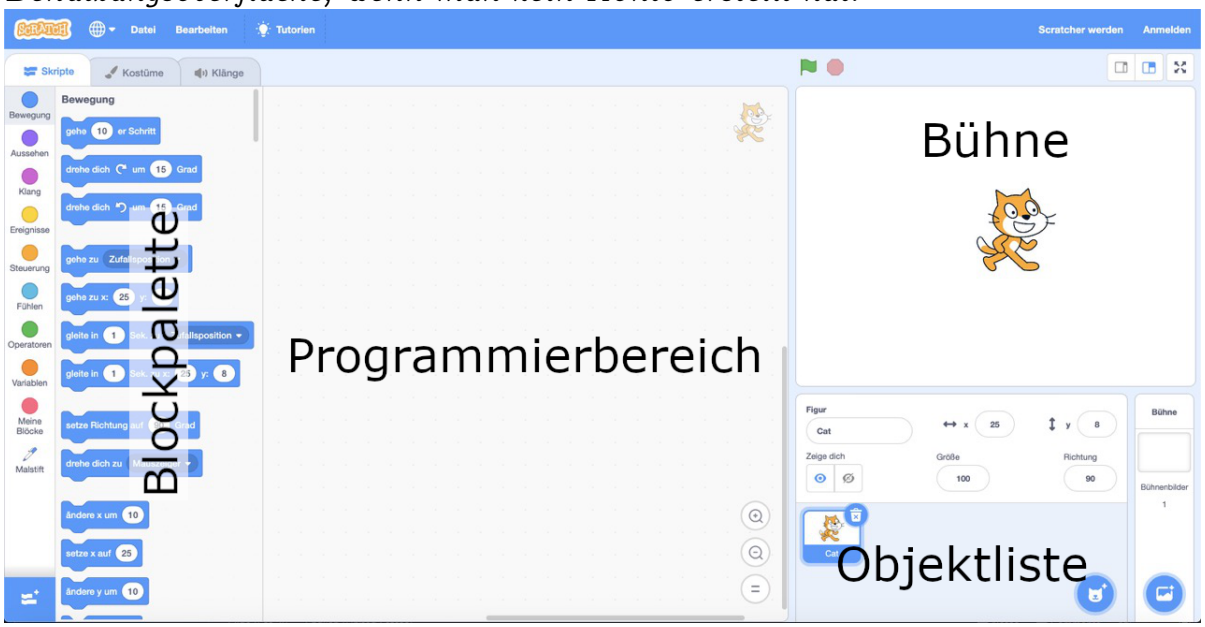
Scratch ist eine Programmiersprache, die es dir ermöglicht, deine eigenen interaktiven Geschichten, Animationen, Spiele, Musik und Kunstwerke zu erstellen und sie als *Scratch-Projekte* anderen über das Internet mitzuteilen. Bei Scratch programmierst du mit Blöcken, die du wie Legosteine stapelst, um 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.

Als Lehrperson kann man auch einen Account anfordern, mit dem sich Accounts für die Lernenden erstellen lassen und die Projekte verwalten lassen. Mehr dazu hier: https://scratch.mit.edu/educators/faq
Ein Account hat Vor- und Nachteile. Einerseits kann der Social Media Aspekt von Scratch die Kinder motivieren und sie können sich mit Gleichgesinnten über ihre Erfahrungen austauschen. Andererseits muss man dafür auch die negativen Seiten von Social Media in Kauf nehmen (Gefahr von Cyberbullying, Verbreitung von Fehlinformationen, ...). Man kann dies aber natürlich auch als Anlass nehmen, den Lernenden diese Themen sichtbar zu machen, und Verhaltensregeln zu diskutieren. Wenn ein Account erstellt wird, sollten auf jeden Fall Sicherheitsregeln zur Wahl des Passwortes und des Nutzernamens diskutiert werden. Gute Übungen und Beispiele zu diesen Themen finden sich in Kapitel 1I-L von connected 01 und den Unterrichtsarrangements "Meine persönlichen Daten" und "Unsere Netiquette" in inform@21. Im Allgemeinen ist es wichtig zu diskutieren, welche Informationen man im Netz veröffentlicht.
Scratch ist auch als offline-Version verfügbar: https://scratch.mit.edu/download/. Diese Version kann man auch ohne Internetverbindung nutzen. Das kann hilfreich sein, wenn die Lernenden zu viel Zeit damit verbringen, andere Projekte auf Scratch auszuprobieren oder im Internet zu surfen.

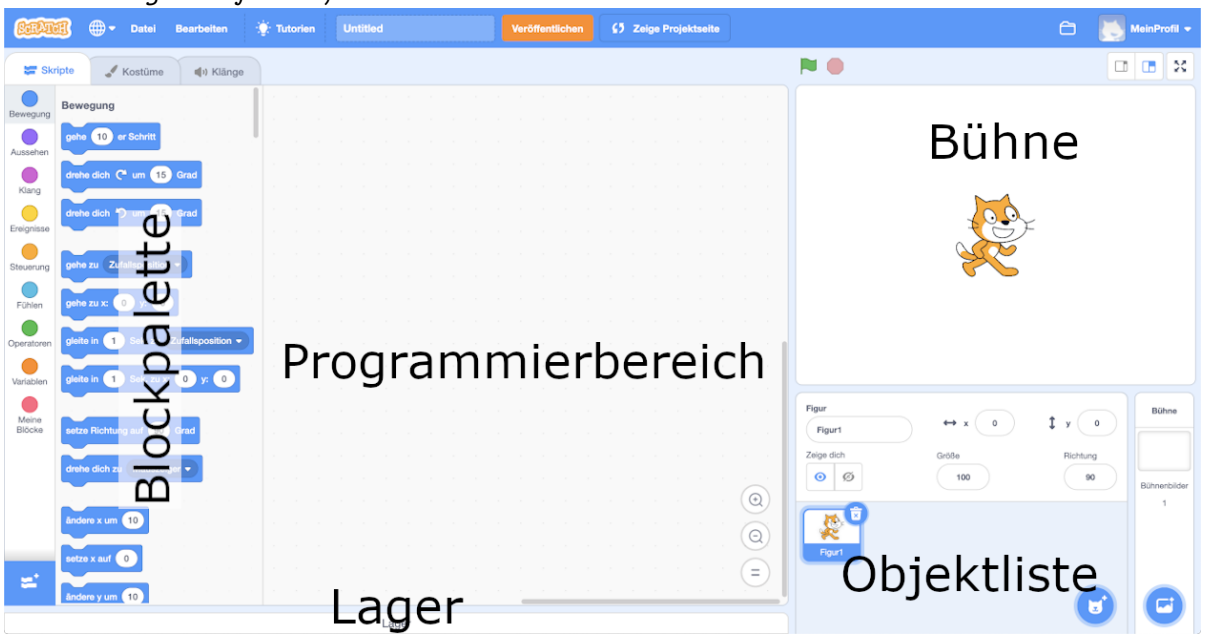
2.1 Die Benutzungsoberfläche

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 han-

delnden 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*.

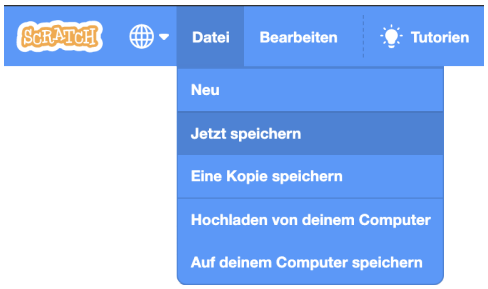
Objektliste Um ein Objekt zu programmieren, wählst du es hier aus. Am Anfang gibt es nur ein Objekt, die Katze Scratch.

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

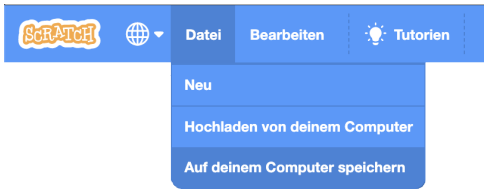
Mit einem Klick auf die Weltkugel oben links kannst du die Sprache ändern:

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!

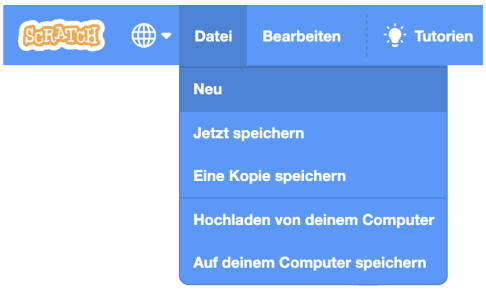


Wenn du keinen Account hast, kannst du deine Projekte auf dem Computer als Datei speichern.

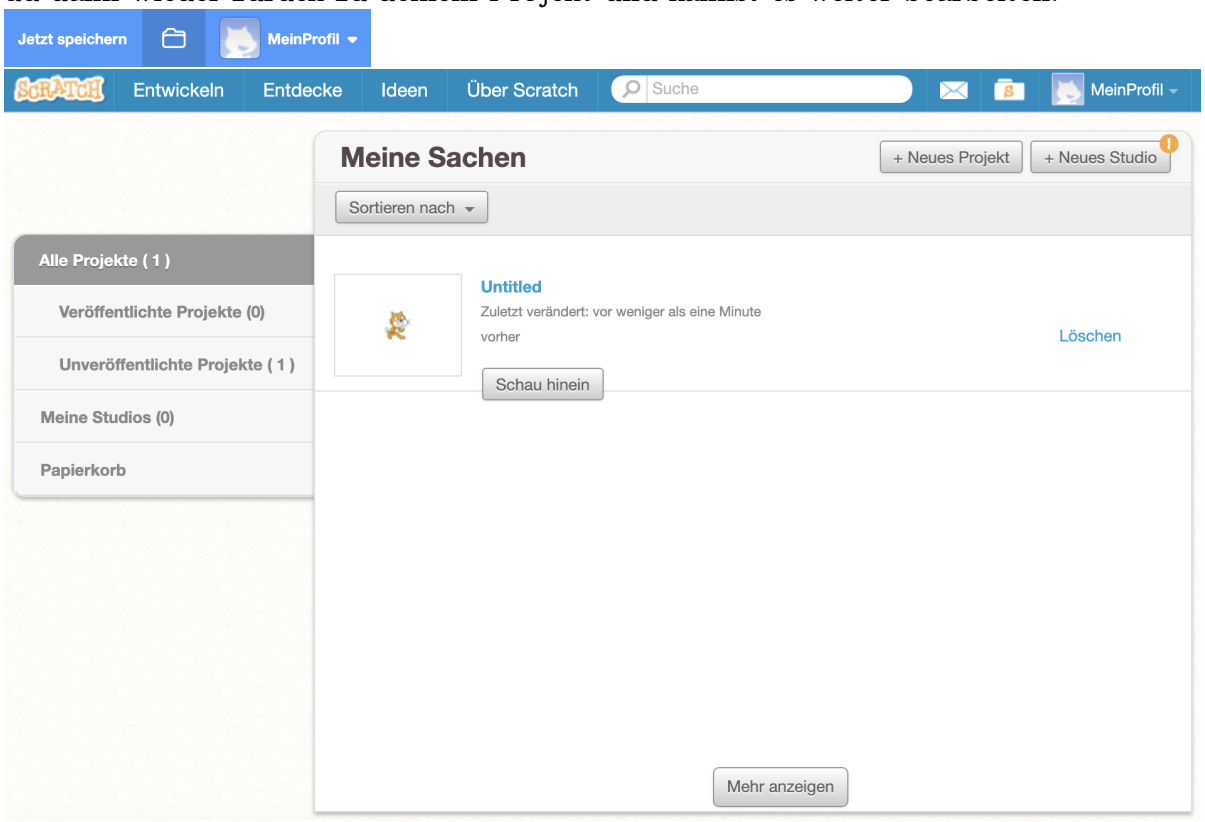


Lege am besten eine Ordnerstruktur an, damit du deine Projekte gut wiederfindest. Wenn du ein Projekt weiter bearbeiten willst, klicke auf "Hochladen von deinem Computer" und wähle dann die Datei aus.

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.



Wenn man einen Account erstellt hat, kann man die Projekte auch in Studios sortieren. Dazu müssen sie aber vorher veröffentlicht werden.
Mit Scratch lassen sich viele verschiedene Dateitypen entdecken, und somit diese Medienkompetenz zumindest teilweise abdecken:

- Kostüme für die Bühne oder für ein Objekt können als Vektorgrafik (.svg) oder als Rastergrafik (.png) gespeichert werden. Hochladen kann man auch Bilder in anderen Bildformaten wie zum Beispiel .jpg.
- Klänge haben die Endung .wav doch es können auch .mp3 Dateien hochgeladen werden.
- Listen werden als .txt Datei gespeichert, wobei die einzelnen Elemente der Liste durch Zeilenumbruch getrennt werden. Listen werden aber in diesem Skript nicht behandelt.
- Objekte (also ein Paket von Kostümen, Klängen und Skripten) haben die Extension .sprite3.
- Programme (Paket von Objekten inklusive Bühnenobjekt) haben die Extension .sb3.

Auch die nächste Aufgabe bezieht sich auf Medienkompetenzen. Wenn man sich aufs Programmieren konzentrieren will, kann man sie weglassen.

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

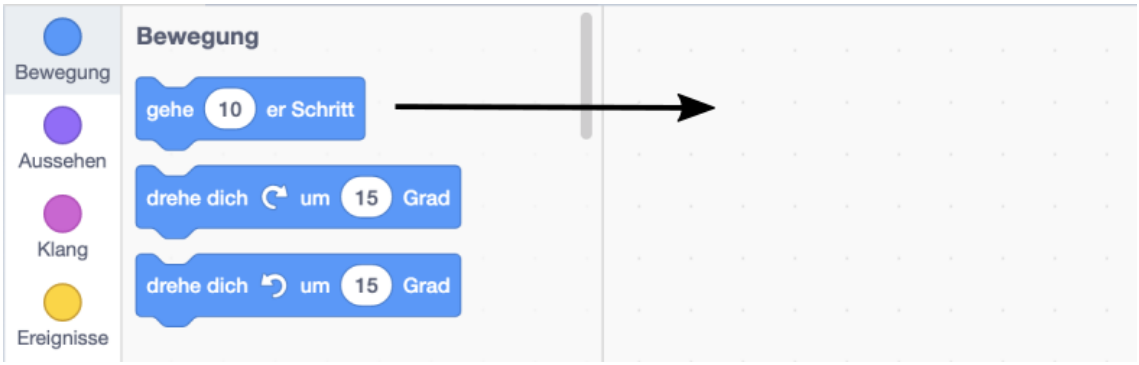
Dein Erstes Projekt

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:

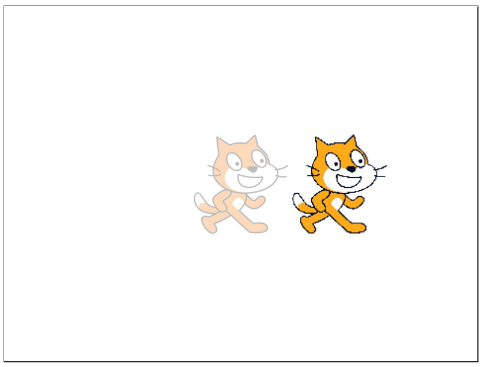
- (Scratch-)Befehl
- Parameter
- Skript (Befehlsfolge)
- Attribut (Merkmal)
- Hut (Ereignisbehandlung)
- Ansichten: Entwicklungsansicht und Präsentationsmodus
- Versionierung, Versionsnummer
- Studio

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.

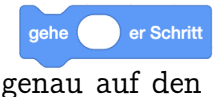
Aufgabe 3.1

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 

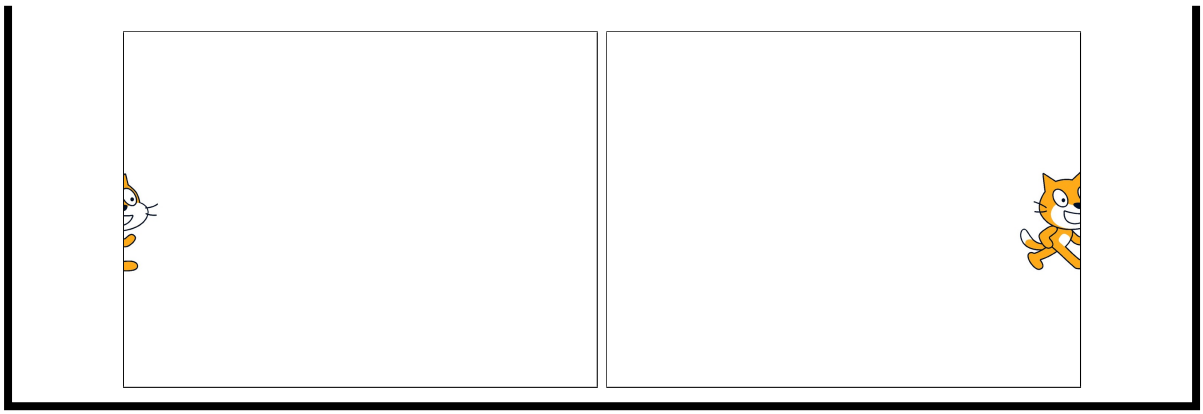
benutzen!

Wenn negative Zahlen noch nicht offiziell im Unterricht eingeführt worden sind, kann man bei dieser Aufgabe das Prinzip kurz erklären.

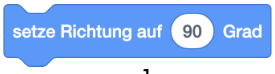
Aufgabe 3.2

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?



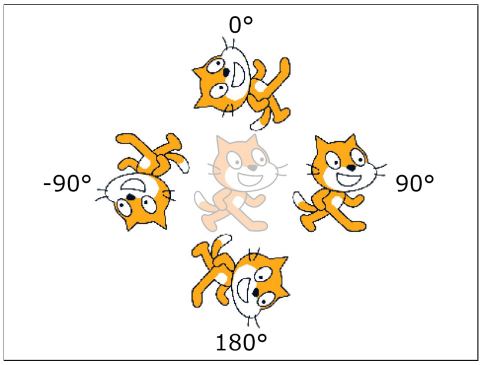
3.2 Die Richtung wählen

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:

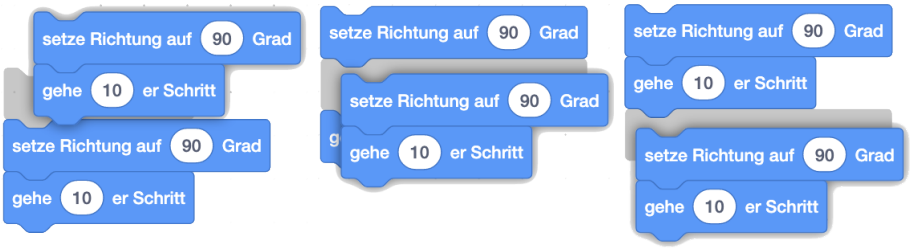
Klicke den Stapel ein paarmal an und beobachte dabei wieder die Bühne!

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.

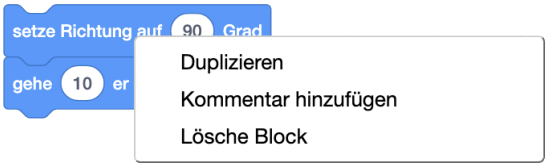


Ein Stapel von Befehlen ist ein *Skript*. Bei jedem Anklicken des Stapels führt dein Objekt alle Befehle des Skripts einmal aus (von oben nach unten).

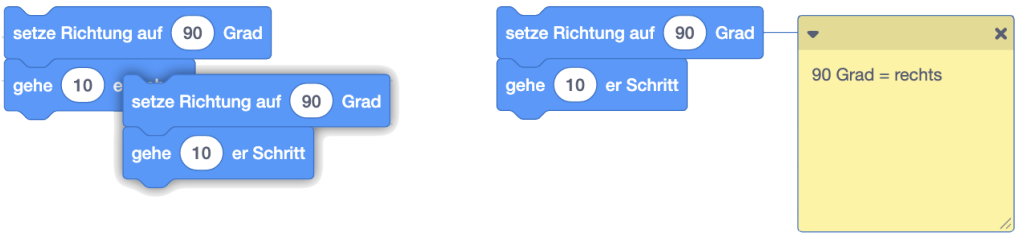
Du kannst Stapel von oben, in der Mitte oder von unten an ein bestehendes Skript anbauen:



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.



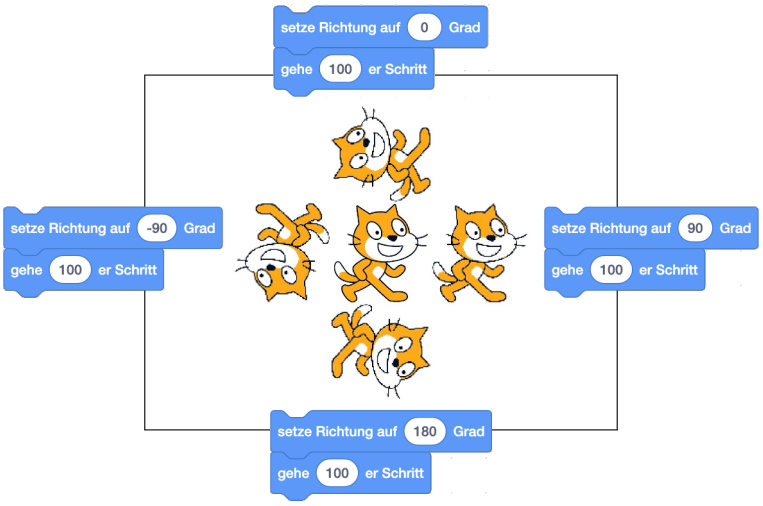
Mit Rechtsklick auf den Block kannst du ausserdem einen Block duplizieren oder einen Kommentar hinzufügen.



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)

b)

c)

d)

e)

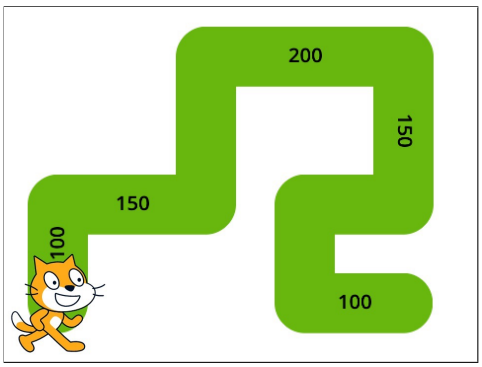
f)

Der zweite Teil dieser Aufgabe lässt sich gut im Gruppenverband umsetzen: Auf der Wandtafel wird die Startposition und -richtung eingezeichnet. Die Lernenden erhalten bei jeder Teilaufgabe eine Papierfigur von Scratch, die sie mit einem Magnet an die richtige Stelle hängen. Wenn sich nicht alle einig sind oder ein Kind unsicher wirkt, zeigt jemand die einzelnen Schritte langsam vor. Wenn die Aufgaben zu einfach sind, können sich die Lernenden gegenseitig ähnliche Aufgaben stellen oder die Startposition und -richtung wird verändert.

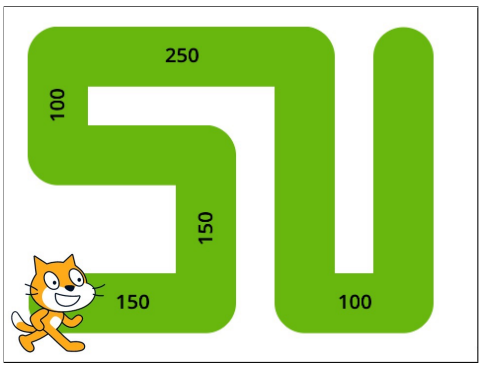
Aufgabe 3.4

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?

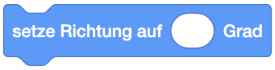
1.



2.



Aufgabe 3.5

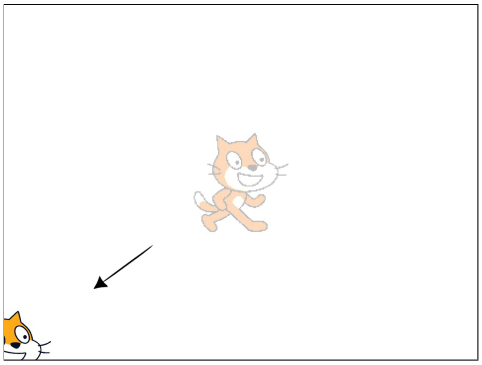
Wieviele Schritte misst die Bühne von ganz unten bis ganz oben? Tipp: Drehe Scratch mit dem 

-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!



Diese Aufgabe können die Lernenden auch auf Papier lösen. Zuerst sollten sie sich einen Plan machen, auf welchem Weg Scratch in die linke untere Ecke kommt.

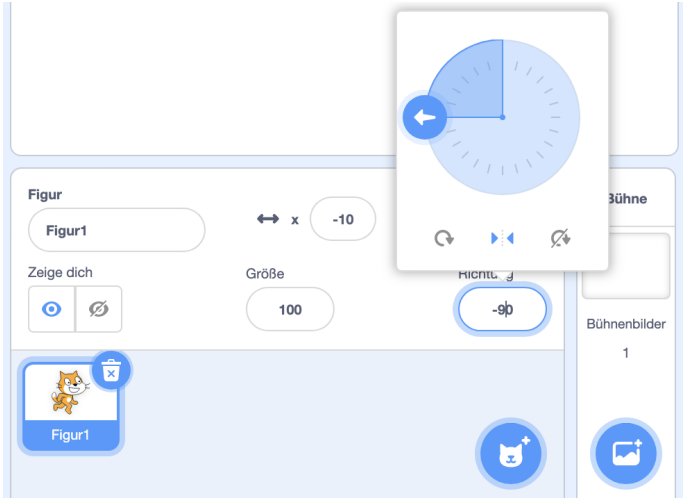
3.3 Ein Objekt animieren

Baue dir im Programmierbereich diese beiden Skripte zusammen:

Die violetten Blöcke findest du in der Blockpalette unter „Aussehen“, die blauen

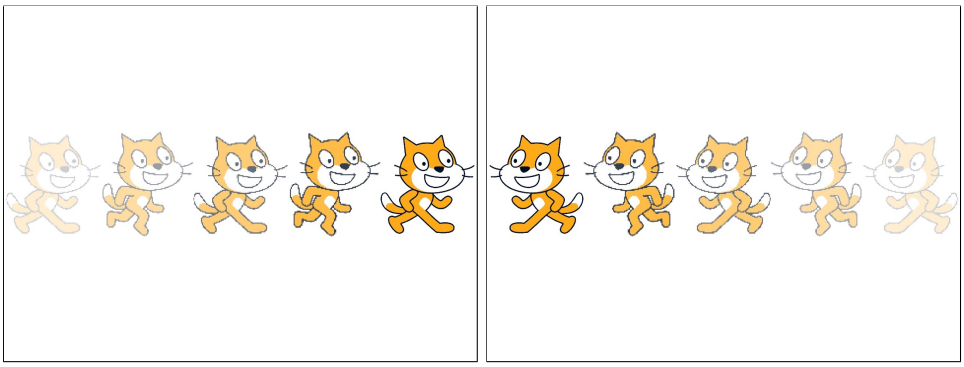
bei „Bewegung“.
Setze ausserdem den Drehtyp auf links-rechts, indem du entweder auf den Block 

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



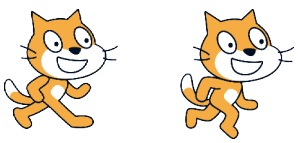
Klicke beide Skripte jeweils ein paarmal an. Was passiert dabei auf der Bühne?

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

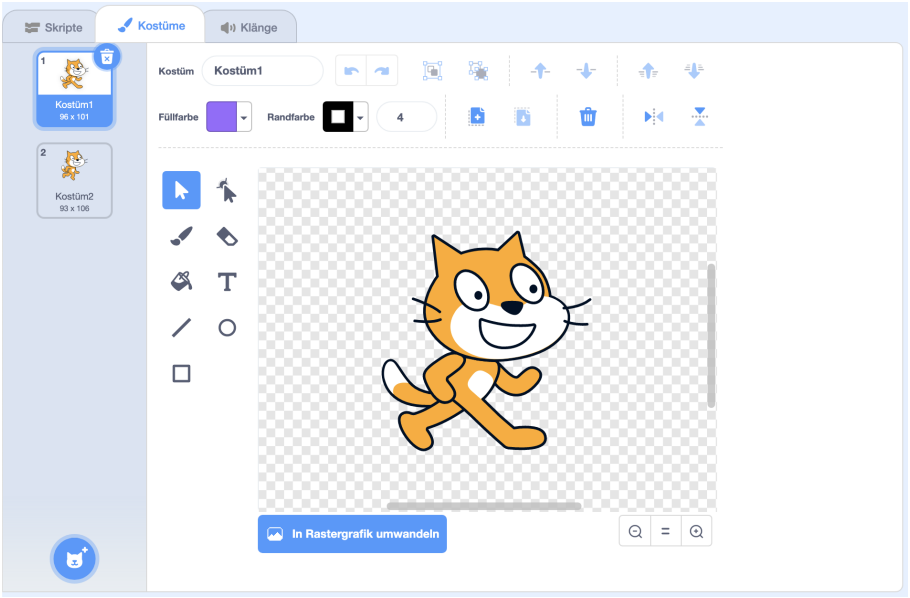


Erreicht wird das durch Kostümwechsel. Scratch hat am Anfang zwei Kostüme zur Auswahl:

Kostüm 1: Kostüm 2:



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

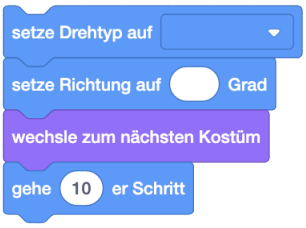


Das Kostüm ist ein *Attribut (Merkmal)* des Objekts. Andere Attribute sind Drehtyp, Standort, Richtung und Verhalten des Malstifts (siehe Kapitel 3.5). Alle Attribute zusammen beschreiben den aktuellen Zustand des Objekts, der beim Speichern des Projekts mitgespeichert wird.

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.

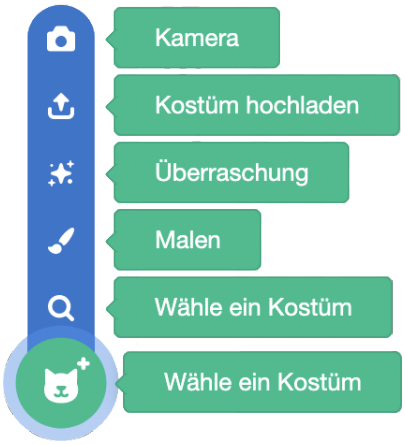


Dann probiere aus, ob deine Vermutung stimmt!

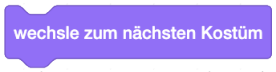
Aufgabe 3.8

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.

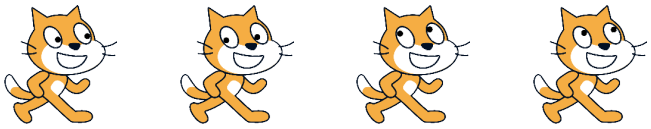


Beim Hochladen von Bildern aus dem Internet sollte auf Copyright geachtet werden. Auf https://pixabay.com/de/ gibt es beispielsweise lizenzfreie Bilder.
Die Aufnahme von Bildern mit der Webcam kann die Kinder begeistern, da sie sich dadurch selbst in ein Spiel oder eine Animation einbauen können. Es sollte aber vorher besprochen werden, dass diese Bilder dann auf ewig im Internet sind.

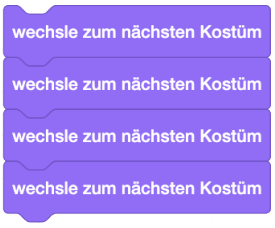
Um die Animation zu sehen, klicke dich mit dem Befehl 

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.

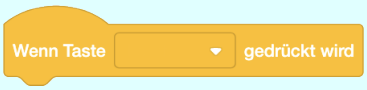
Wenn du eine der Pfeiltasten gedrückt hältst, läuft Scratch flüssig in die zugehörige Richtung.

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.

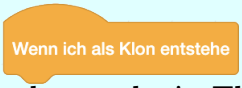
Es gibt leider zu Scratch keine offizielle Dokumentation die genau beschreibt, welchen Effekt ein Block hat. Daher können wir hier nicht eindeutig beschreiben, welcher Hut welcher Regel folgt - denn Scratch kann sich von Version zu Version auch etwas ändern. Scratch versucht jedoch, dass auch alte Projekte die zum Beispiel mit Scratch 2.0 erstellt wurden grösstenteils weiterhin funktionieren. Zur Zeit folgen die Hüte den Regeln folgendermassen:

Das Skript wird abgebrochen und beginnt von vorne:

Das Skript ignoriert das Ereignis und wird zu Ende ausgeführt:



Ein Spezialfall ist der  - Hut. Das Skript wird hier nur ausgeführt, wenn der Wahrheitswert der Bedingung von falsch zu wahr wechselt. Das Skript wird ganz ausgeführt, auch wenn der Wahrheitswert sich währenddessen ändert. Dieser Block wird in unserem Lernmaterial nicht näher vorgestellt.

Ein weiterer Spezialfall ist der  -Hut. Dieses Ereignis kann nur einmal auftreten. Klonen ist aber auch ein Thema, auf das wir in diesem Lernmaterial nicht genauer eingehen.

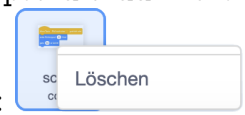
Aufgabe 3.9

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

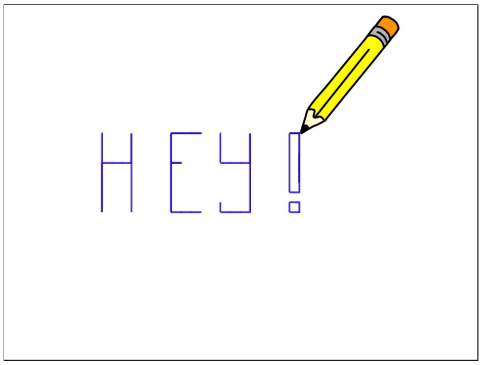
Male für Scratch ein Stift-Kostüm oder wähle ein bereits vorhandenes Stiftkostüm

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:

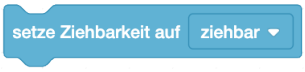
Die Malstift-Blöcke kannst du hinzufügen, wenn du unten in der Blockpalette auf  klickst und dann “Malstift” auswählst. Wechsle durch einen Klick auf das Symbol  oben rechts über der Bühne in den *Präsentationsmodus*. Bewege den Stift mit den Pfeiltasten herum und beobachte, was nach dem Drücken von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

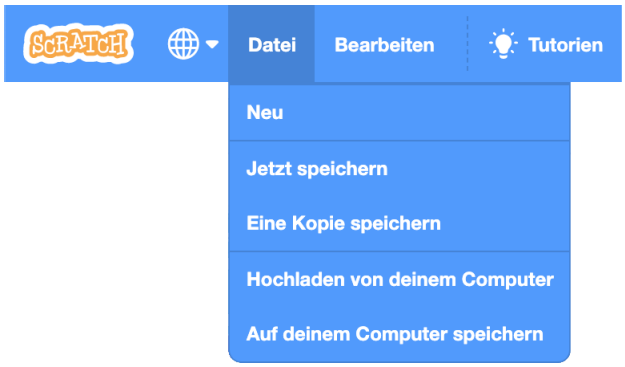
Es gibt verschiedene *Ansichten* deines Programms. Die Ansicht, die wir bis jetzt verwendet haben, heisst *Entwicklungsansicht*. In der Entwicklungsansicht kannst du neue Blöcke zu deinem Programm hinzufügen, sie durch Anklicken ausführen und die Objekte auf der Bühne an die richtige Stelle ziehen. Diese 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.



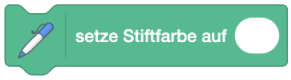
Auf diese Weise kannst du sicherstellen, dass du immer eine funktionierende Version deines Projektes hast, auch wenn sich in der neuen Version irgendwo ein Fehler einschleicht. Denk daran, die Kopie umzubenennen, damit du weisst, was du in welcher Version hinzugefügt hast.

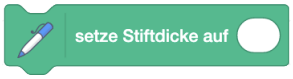
Häufig gibt man in der Benennung einer neuen Version eines Projektes auch die *Versionsnummer* an. Diese zählt mit, wieviele Versionen man bereits vorher erstellt hat. So kann man schnell vergleichen, welches die neuste Version ist. Die Dokumentation und Speicherung von Änderungen nennt man *Versionierung* 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.

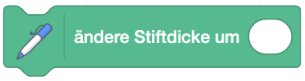
Bei grösseren Programmen setzt sich die Versionsnummer aus Hauptversionsnummer, Nebenversionsnummer und Revisionsnummer zusammen. Die Hauptversionsnummer wird erhöht, wenn eine grundlegend neue Version des Produktes entsteht. In manchen Fällen (wie z.B. Scratch oder Python) wird dann auch die ältere Version noch für eine Weile unterstützt. Die Nebenversionsnummer wird erhöht, wenn es kleinere Änderungen am Projekt gab (zum Beispiel eine neue Funktionalität). Die Revisionsnummer wird erhöht, wenn ein Fehler im Programm korrigiert wurde. Alle Nummern können dabei auch mehrstellig werden. Es ist Konvention, dass bei einer Erhöhung der Hauptversionsnummer die Nebenversionsnummer und Revisionsnummer auf 0 zurückgesetzt werden. Ebenso wird bei einer Erhöhung der Nebenversionsnummer die Revisionsnummer auf 0 zurückgesetzt.
Eine gute Versionsverwaltung ist nicht nur in der Informatik wichtig. Bei allen Dokumenten, die mehrmals überarbeitet werden, lohnt es sich, ein klares System für die Versionierung aufzustellen. Dieses in der Informatik gebräuchliche System ist für die Lernenden aber noch zu kompliziert und für die mit Scratch erstellten Programme auch nicht notwendig. Hier reicht eine simple Nummerierung.

Aufgabe 3.11

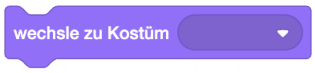
Erweitere das Malprogramm! Vorschläge dafür:

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



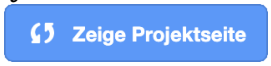
5. Programmiere einen „Radiergummi“: Auf Tastendruck soll der Malstift ein Radiergummi-Kostüm anziehen. Dann soll er radieren, d.h. mit Stiftfarbe weiss malen, wobei die Stiftdicke etwas grösser gewählt ist. Die passende Stiftdicke kannst du z.B. durch Ausprobieren herausbekommen.

3.7 Ein Projekt veröffentlichen

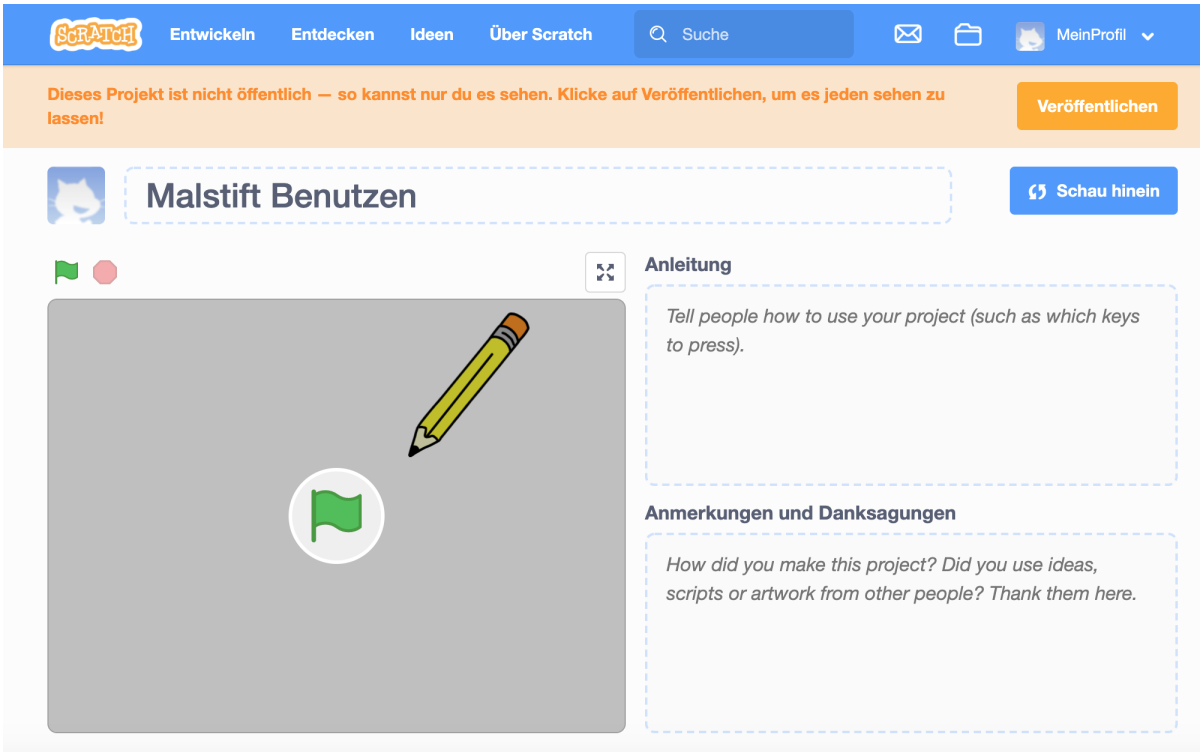
Dieses Unterkapitel kann ausgelassen werden, wenn die Lernenden kein Benutzerkonto erstellt haben. Die genaue Beschreibung eines Programms und dessen Bedienung kann aber trotzdem thematisiert werden.
Ausserdem ist es an dieser Stelle ist sinnvoll zu besprechen, was auf Scratch veröffentlicht werden darf. Die Lernenden dürfen selber Vermutungen anstellen und recherchieren dann, ob sie spezifische Verhaltensregeln für Scratch finden (siehe https://scratch.mit.edu/terms_of_use und https://scratch.mit.edu/community_guidelines). Welche Massnahmen können die Lernenden ergreifen, wenn sie auf unangemessene Beiträge stossen?

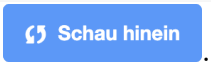
Wenn du mit einem Programm (zum Beispiel dem Malprogramm aus dem vorherigen Kapitel) zufrieden bist, kannst du es veröffentlichen. Zum Veröffentlichen brauchst du ein Benutzerkonto. Erst wenn du dein Projekt veröffentlichst, können es andere anschauen. Frage aber vor dem Veröffentlichen deine Eltern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

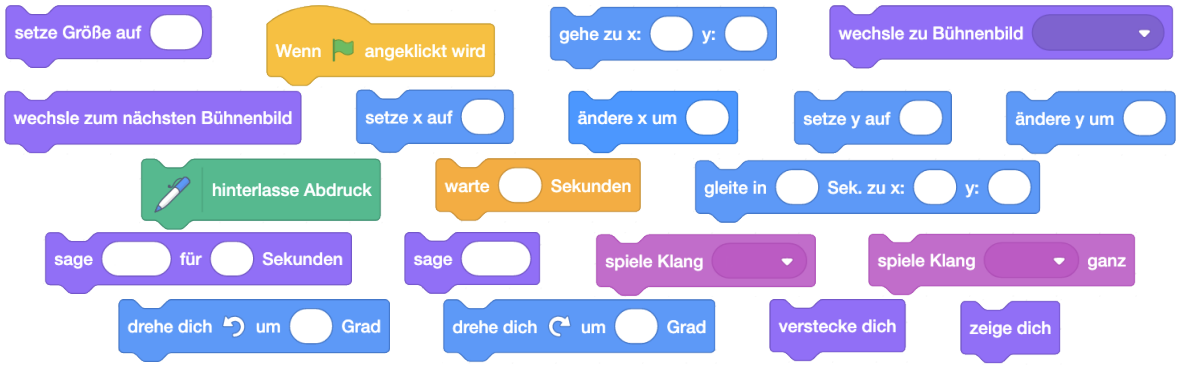
Es kann sich auch lohnen, pro Kapitel ein Studio zu erstellen. Leider ist es in Scratch nicht möglich, ein Projekt direkt im Studio zu erstellen, so wie man das von Ordnern kennt. Stattdessen erstellt man das Projekt und fügt es dann dem Studio hinzu. Studios lassen sich auch nur erstellen, wenn man ein Benutzerkonto erstellt hat. Ausserdem können nur veröffentlichte Projekte zu einem Studio hinzugefügt werden. Wenn die Lernenden keine Projekte veröffentlichen dürfen, kann dieser Hinweis also auch übersprungen werden. Studios können aber hilfreich sein, um Lösungen aller Lernenden zu einer Aufgabe zu sammeln. Alternativ lassen sich Projekte aber auch auf dem Computer in der gewohnten Ordnerstruktur speichern (und dann zum Beispiel per E-Mail oder USB-Stick miteinander ausgetauscht werden). Dies ist besonders relevant, wenn die Offline-Version von Scratch verwendet wird.

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:



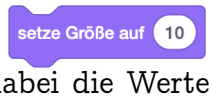
Die Begriffe, die hier neu eingeführt werden:

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

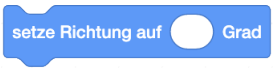
4.1 Die Position wählen

Wenn die Lernenden das Konzept des Koordinatensystems bereits gut kennen, kann dieser Teil abgekürzt werden. Die Einführungsaufgabe eignet sich gut, um die Lernenden selbst das Konzept entdecken zu lassen, zum Beispiel auch in Zweier-Gruppen. Die zugehörigen Aufgaben können auch gut an der Wandtafel mit einem eingezeichneten Koordinatensystem und ausgeschnittenen Befehlsblöcken durch-

geführt werden (siehe Kapitel 0.7). In den Aufgaben 4.4, 4.5 und 4.6 sammelt man zuerst verschiedene Pläne, mit denen sich die jeweilige Aufgabe lösen lässt. Dann fügt ein Kind den passenden Befehl zum Skript hinzu und ein anderes Kind bewegt Scratch entsprechend und zeichnet die hinterlassene Spur.

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

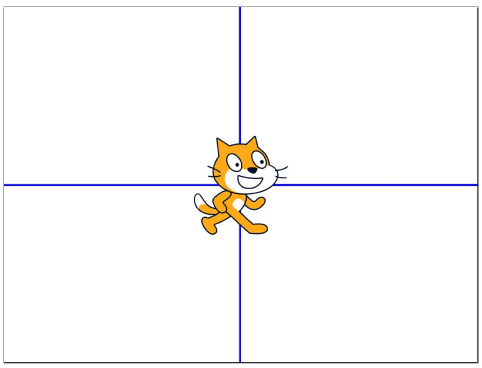
Minimaler Wert von x	-240
Maximaler Wert von x	240
Minimaler Wert von y	-180
Maximaler Wert von y	180
Werte in der Mitte der Bühne	x=0, y=0
Setze Richtung auf	Keine Auswirkung auf x und y
10er Schritt in Richtung 90 Grad (rechts)	x wird um 10 grösser
10er Schritt in Richtung -90 Grad (links)	x wird um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

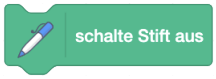
Was denkst du passiert, wenn man für Scratch das folgende Skript erstellt und dann auf die grüne Flagge über der Bühne klickt? Zeichne deine Vermutung auf und probiere dann aus, ob das stimmt!

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

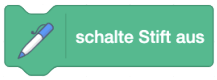


Aufgabe 4.1

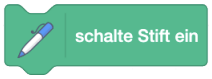
Welches Bild malt Scratch, wenn du aus diesem Skript den jeweils angegebenen Block entfernst?

a) Den ersten 

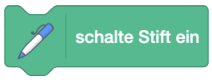
- Block

b) Den zweiten 

- Block

c) Den ersten 

- Block

d) Den zweiten 

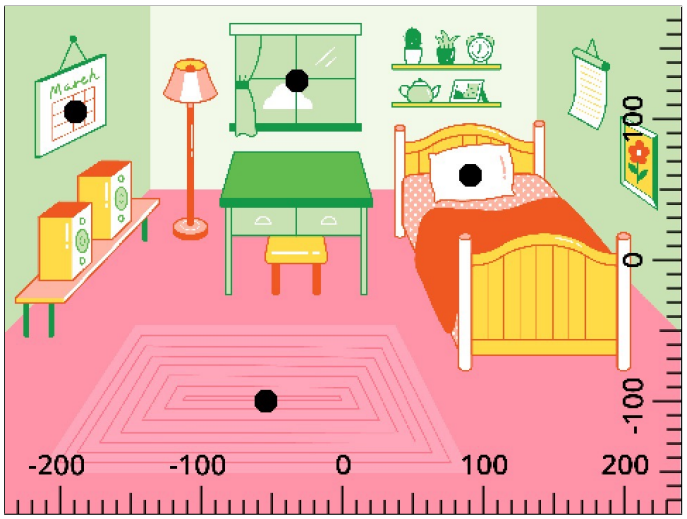
- Block

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.

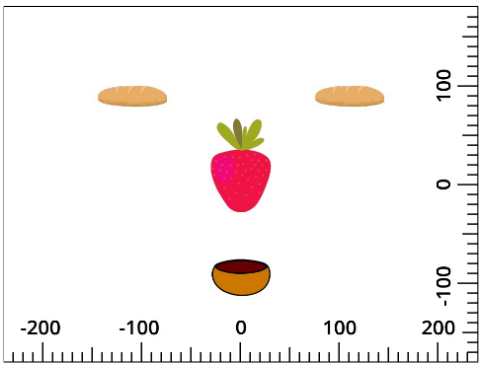
Kalender:
(,)
(,)
Fenster:
(,)
(,)
Kissen:
(,)
(,)
Teppich:
(,)
(,)



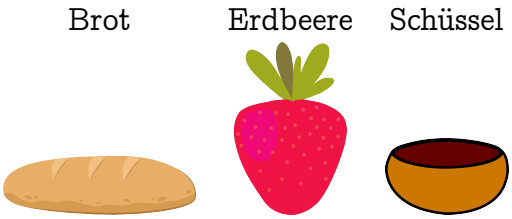
Aufgabe 4.3

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

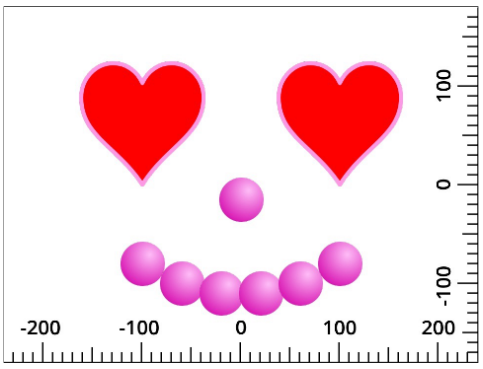
1.



Verwendete Kostüme:

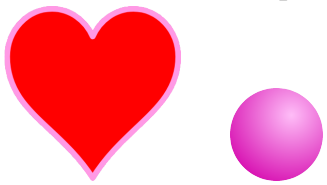


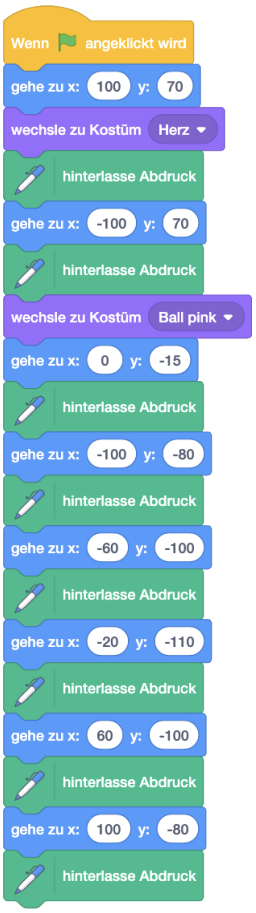
2.



Verwendete Kostüme:

Herz Ball pink

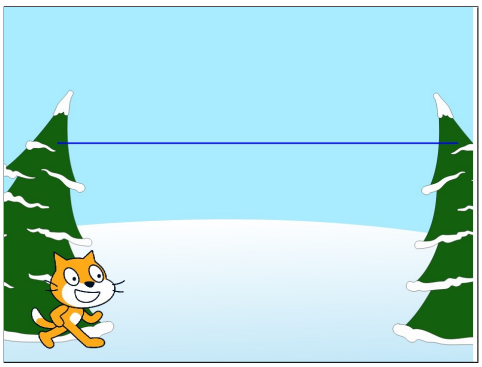




Zusatzaufgabe: Denk dir selber ein “Gesicht” aus und programmiere es!

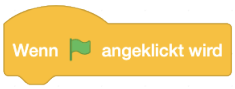
Aufgabe 4.4

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!

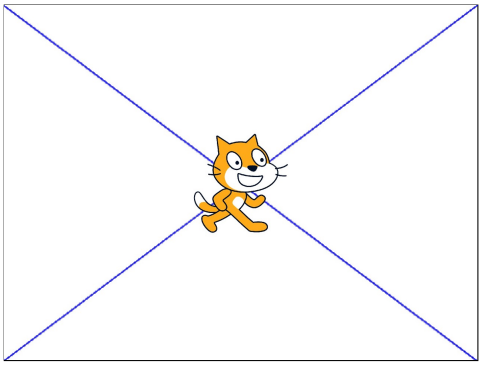


Aufgabe 4.5

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

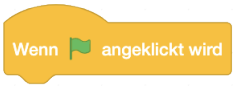


- Block.

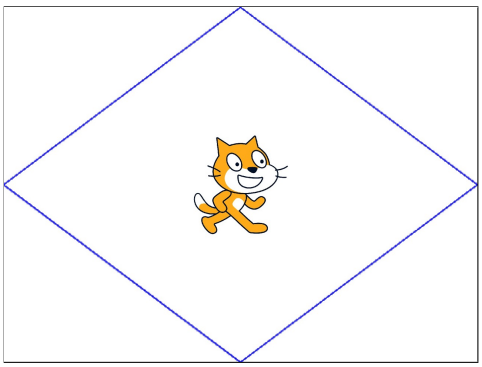


Aufgabe 4.6

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

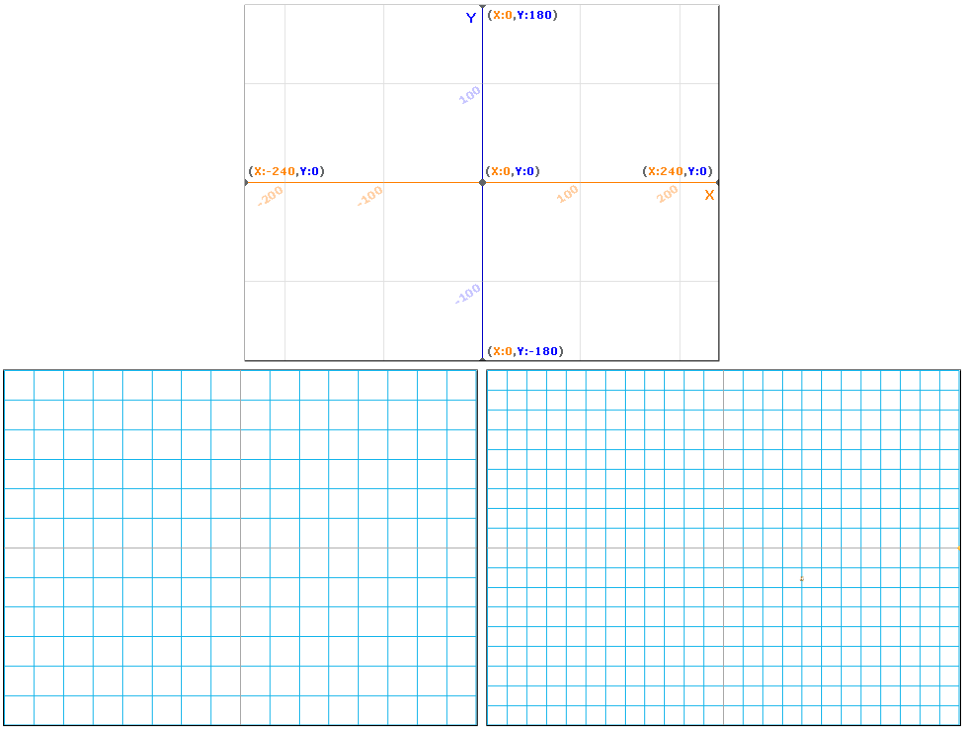


- Block.



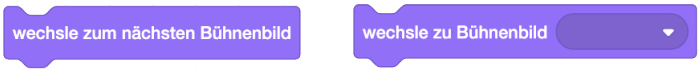
4.2 Den Hintergrund wechseln

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:

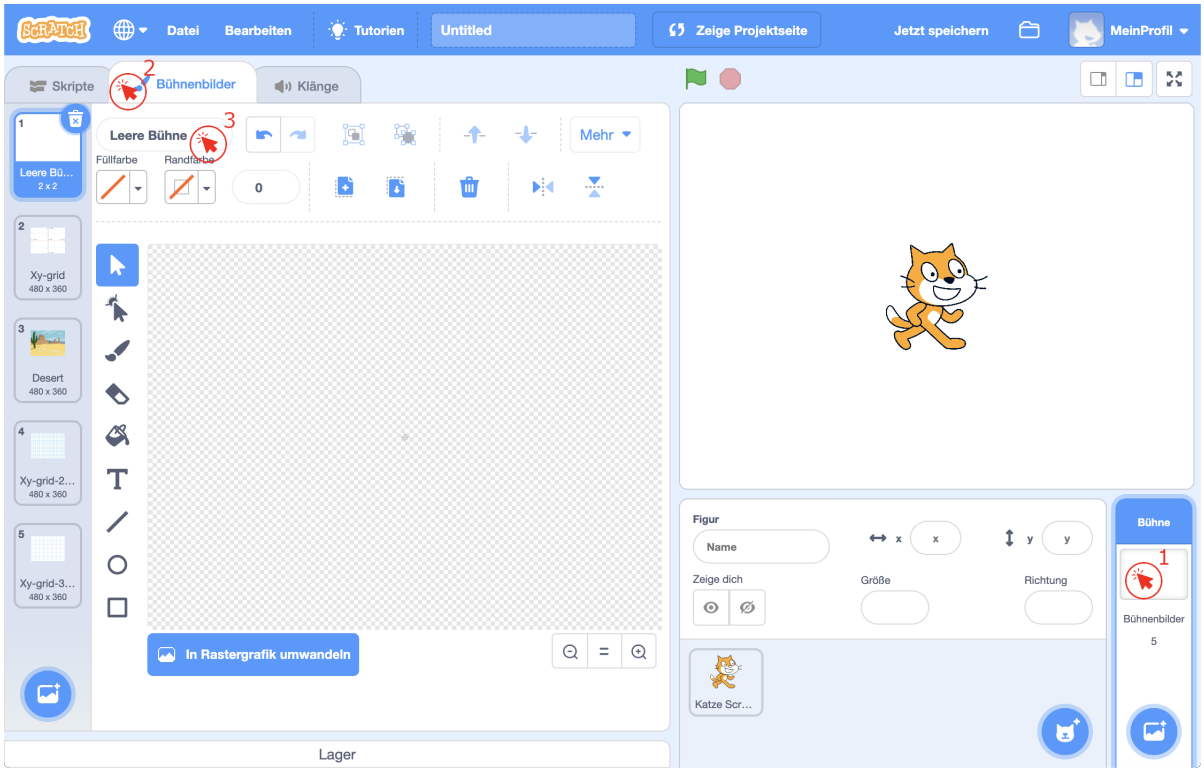


Der erste Hintergrund zeigt dir die Koordinaten in 100er Schritten und die x- und y-Achse sind angeschrieben. Der zweite Hintergrund ist in 30er Schritten gekachelt und der letzte in 20er Schritten.

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:



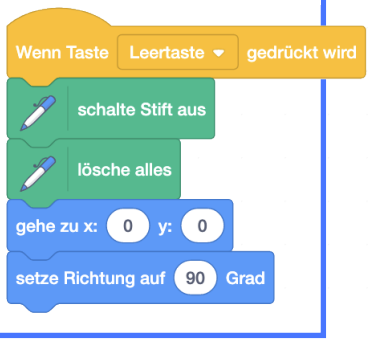
Auch bei Bühnenbildern ist es wieder wichtig, einen guten Namen zu wählen, damit man beim Wechseln genau weiss, welches Bild man auswählen soll. Hier ist zum Beispiel "Bühnenbild1" nicht sehr aussagekräftig - besser wäre vielleicht "Leere Bühne" oder "Alles Weiss". Den Namen der Bühnenbilder kannst du 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

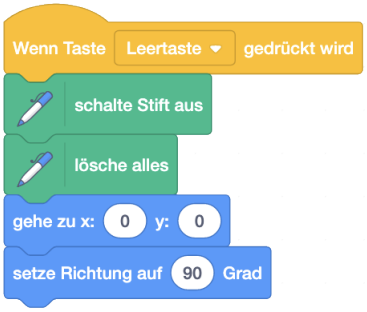
Block beginnen, sind auch Aufräumskripte oft unter diesem Block zu finden.

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

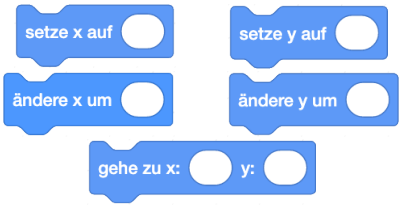
1. Ist die Reihenfolge der Befehle in diesem Aufräumskript egal? Wenn nicht, worauf musst du achten?

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

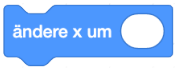
Probiere die folgenden Befehle mit verschiedenen Parametern aus. Welchen Effekt haben sie auf die x- und y-Position von Scratch? Ist das Resultat immer das gleiche?



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)

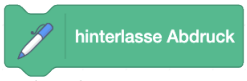
Aufgabe 4.8

Was ist der Unterschied zwischen dem 

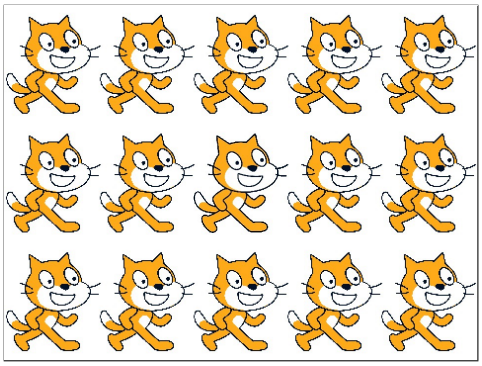
Befehl und dem 

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.



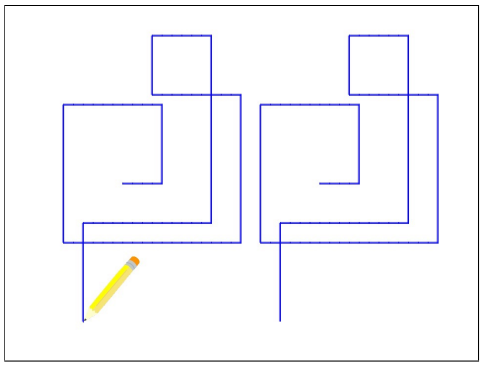
Beachte, dass die Stempelbilder von Scratch alle in die gleiche Richtung schauen!

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.

Da diese Aufgabe eher schwierig ist, kann es sich lohnen, sie vorher mit der Klasse zu besprechen oder die Lernenden in Gruppen arbeiten zu lassen. Zuerst lässt man die Lernenden selbst Ideen für einen Ablauf sammeln. Dieser kann dann zuerst unplugged an der Wandtafel getestet werden. Auch hier sollen die Lernenden möglichst selbstständig herausfinden, wie man die Korrektheit systematisch testen kann (für alle vier Pfeiltasten kontrollieren). Zusätzlich kann man hier noch diskutieren, ob die Aufgabe klar gestellt ist. Was soll passieren, wenn man versucht, direkt auf der rechten Seite zu zeichnen? Was passiert wirklich mit dem erstellten Programm?

4.5 Mehrere Objekte gleichzeitig steuern

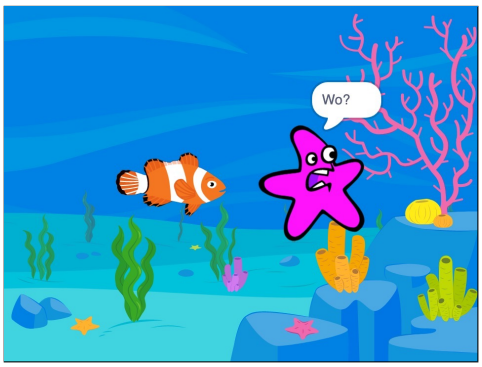
Gib Scratch ein Fisch-Kostüm und erstelle dieses Skript:

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.

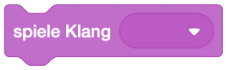
Wenn mehrere Skripte gleichzeitig ausgeführt werden, ist es oft schwierig, den Fehler in einem fehlerhaften Programm zu finden. Nutze das Lager, um einzelne Skripte zwischenzeitlich aus deinem Programm zu entfernen und dein Programm zu testen:

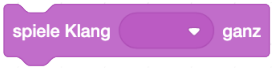
1. Öffne das Lager, indem du es anklickst.
2. Ziehe das Skript ins Lager.
3. Lösche das Skript aus dem Programmierbereich, indem du es in die Blockpalette ziehst.
4. Teste dein Programm: Funktionieren alle anderen Skripte wie erwartet oder besteht der Fehler immer noch? Wenn die übrigen Skripte funktionieren, 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.

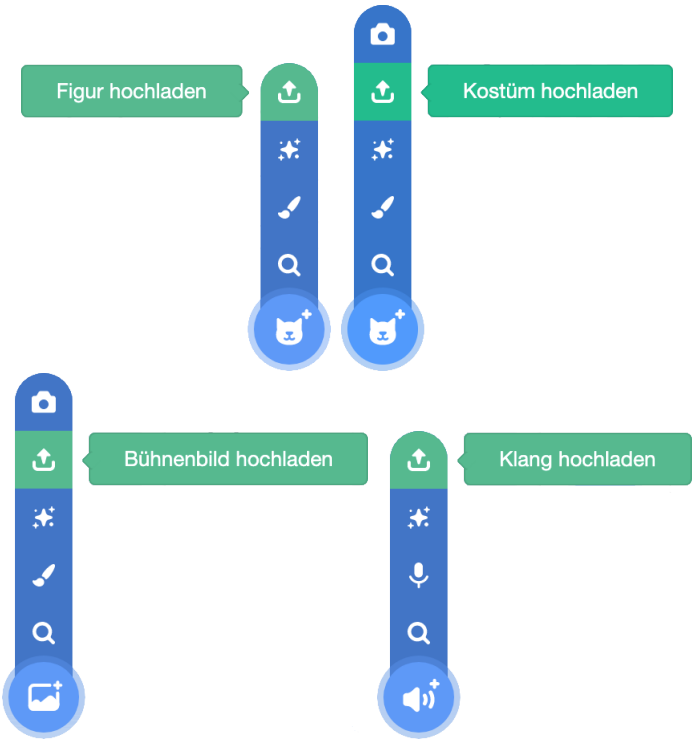
Damit die Lernenden nicht zu lange in der Figuren-Bibliothek von Scratch verbringen oder Kostüme zeichnen, kann man hier ein Zeitlimit setzen. Ausserdem kann man die Anzahl Figuren beschränken. Am Ende können alle Lernenden ihr Projekt kurz vorstellen.

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.

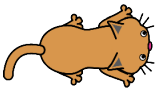
Mit Rechtsklick auf ein Objekt kannst du es auch exportieren. Dadurch speicherst du das Objekt mit all seinen Kostümen, Klängen und Skripten auf dem Computer. So kannst du es in einem anderen Projekt wiederverwenden. Wenn du nur bestimmte Kostüme oder Klänge des Objekts brauchst, kannst du diese auch 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":



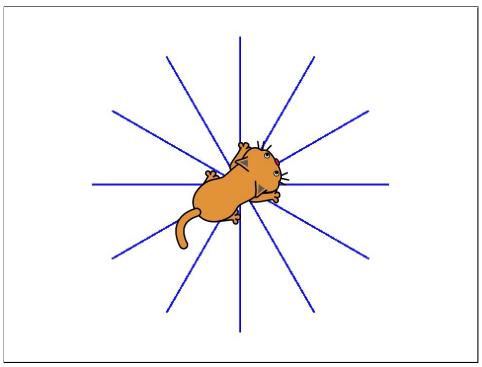
Achte auf die verschiedenen Dateiformate, die du hier verwenden kannst - welche kommen dir bekannt vor und welche sind neu?

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?

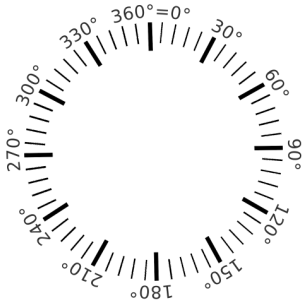


Bei den Pfeiltasten dreht sich Scratch ein Stück nach rechts oder nach links. Die Leertaste „feuert“ eine Linie in die aktuelle Richtung ab.



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

Richtungen:

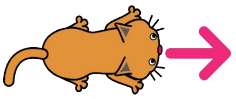


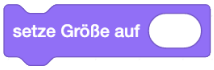
Drehwinkel:

=

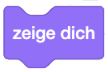
“+ 5 Minuten”

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 

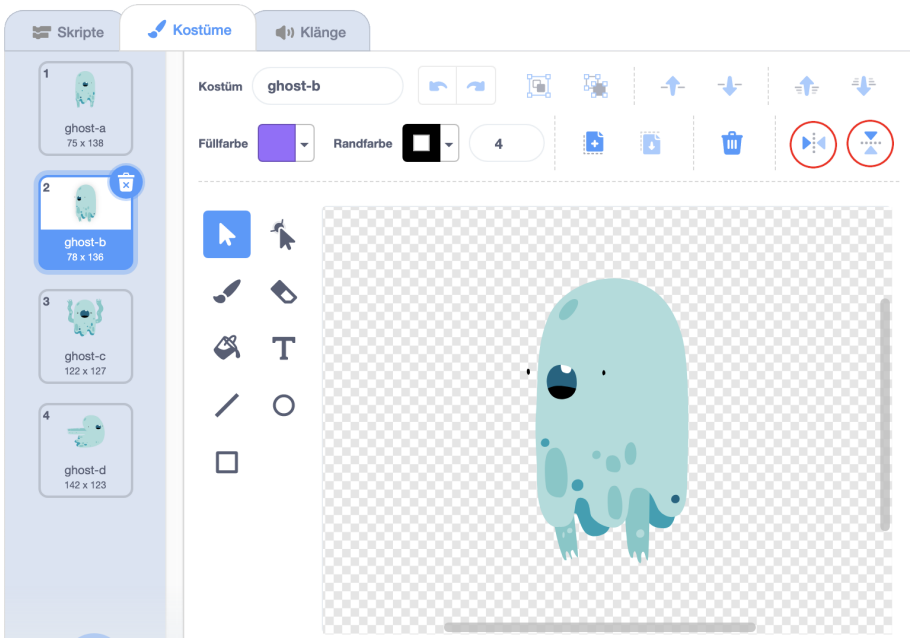
verkleinert worden oder es versteckt sich mit 

, 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:



Die folgenden zwei Aufgaben lassen sich gut im Klassenverband als kurzes Quiz durchführen.

Aufgabe 4.14

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

5 Minuten?

30°

10 Minuten?

1 Minute?

30 Minuten?

20 Minuten?

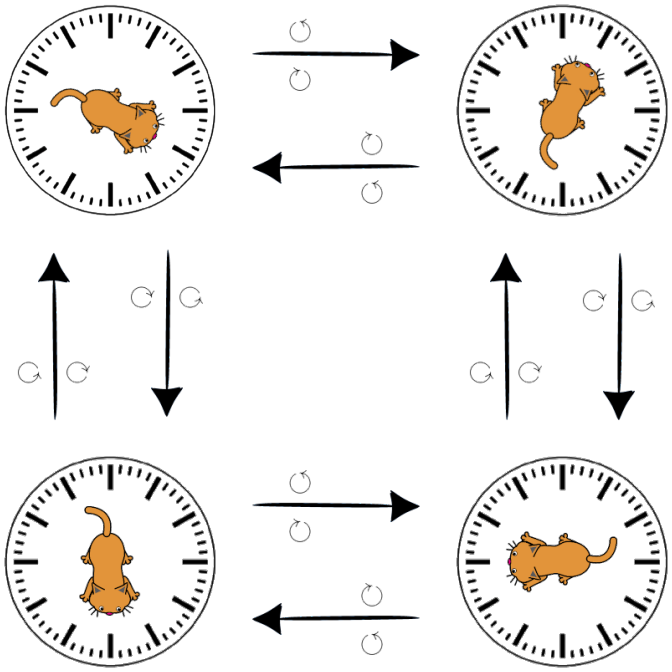
15 Minuten?

7.5 Minuten?

60 Minuten?

Aufgabe 4.15

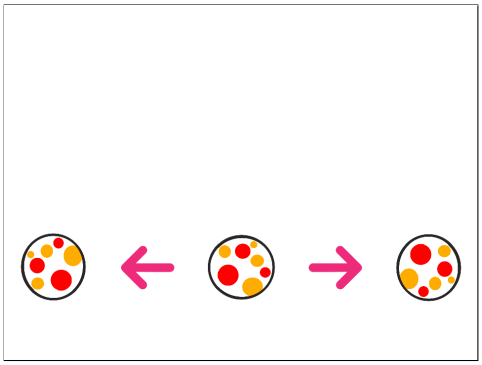
Trage jeweils den Winkel ein, um welchen Scratch sich drehen muss, um von einem Bild zum nächsten zu kommen. Schreibe dabei jeweils den Drehwinkel für die Drehung im Uhrzeigersinn (↻) und gegen den Uhrzeigersinn (↺) auf.



Erkennst du einen Zusammenhang zwischen den Winkeln?

Aufgabe 4.16

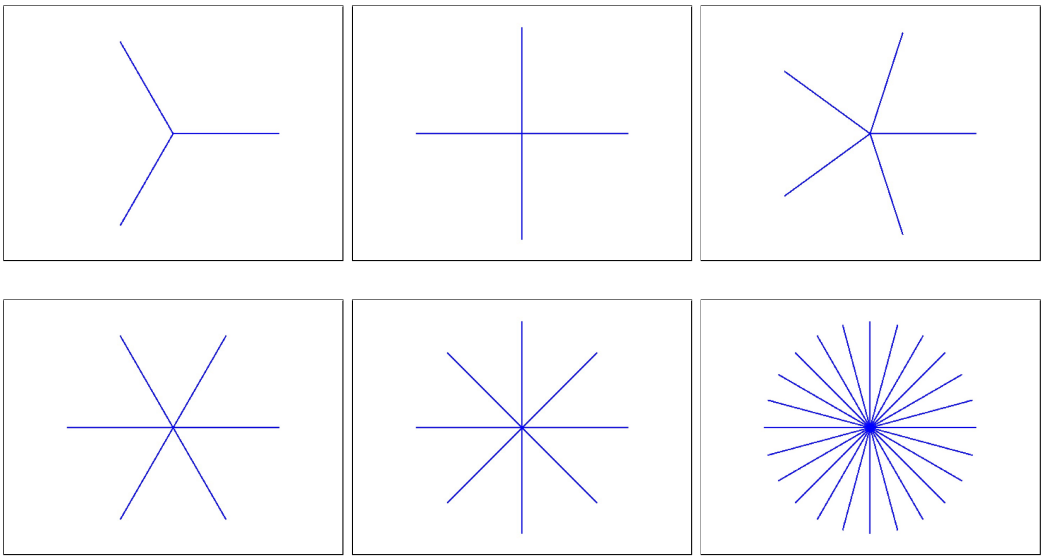
Lade für Scratch das Beachball-Kostüm und erstelle zwei Skripte, mit denen du den Ball mit den Pfeiltasten ← und → auf der Bühne nach links und nach rechts rollen kannst. Der Ball muss sich dabei drehen!



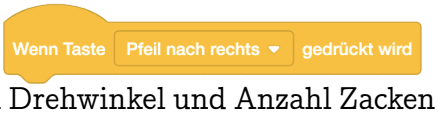
Aufgabe 4.17

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?

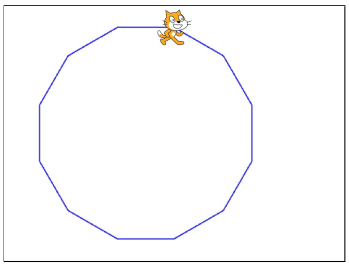
4.7 Vielecke malen

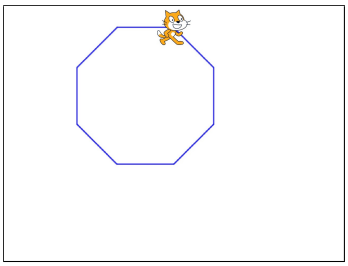
Gib Scratch diese zwei Skripte und probiere im unteren Skript die Winkel 30°, 45°, 60°, 72°, 90°, 120°. Wie oft musst du jeweils die Taste „Pfeil nach rechts“ drücken, bis Scratch wieder am Ausgangspunkt ist, und wie sehen die Bilder aus?

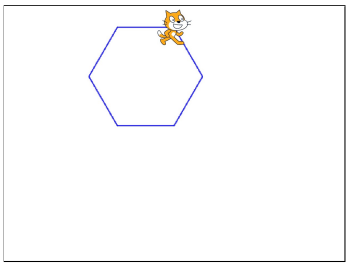
Das Skript sieht sehr ähnlich aus wie die Skripte zu Aufgabe 4.17, es fehlt nur der zweite 

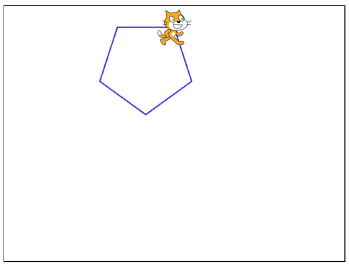
Befehl mit dem man wieder rückwärts läuft. Dieses Einführungsbeispiel ist also auch ein gutes Beispiel um zu zeigen, dass kleine Änderungen im Programm grosse Auswirkungen haben können. Man muss das Programm genau durchlesen, um zu verstehen was passiert, statt einfach zu raten.

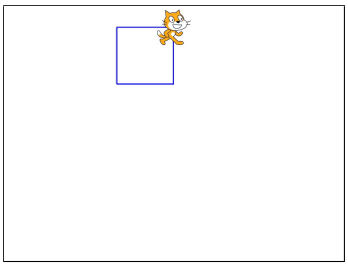
Scratch malt Vielecke. Für jede Ecke musst du einmal → drücken.

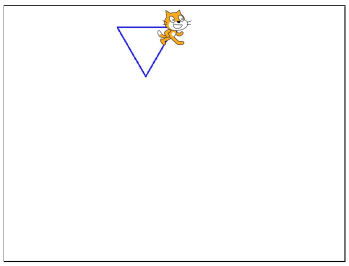

30° / Zwölfeck


45° / Achteck

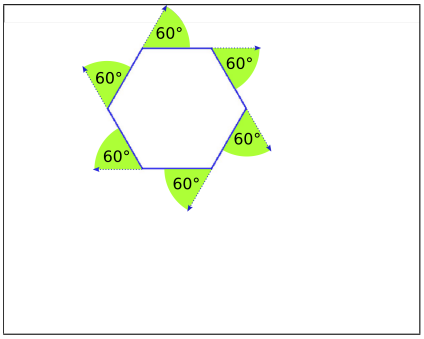

60° / Sechseck


72° / Fünfeck


90° / Viereck

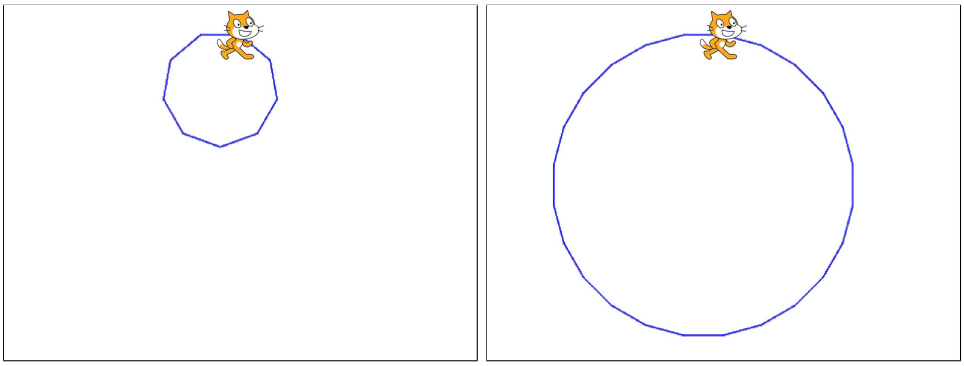

120° / Dreieck

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?

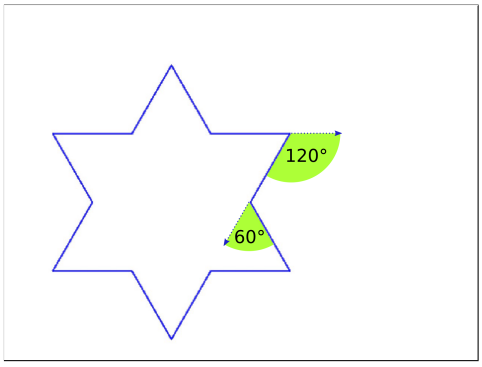


Tipp: Der Kreis ist in Wirklichkeit kein Kreis, sondern ein 24-Eck! Du kannst für beide Bilder das gleiche "Grundskript" nehmen und musst jeweils nur den Drehwinkel ändern.

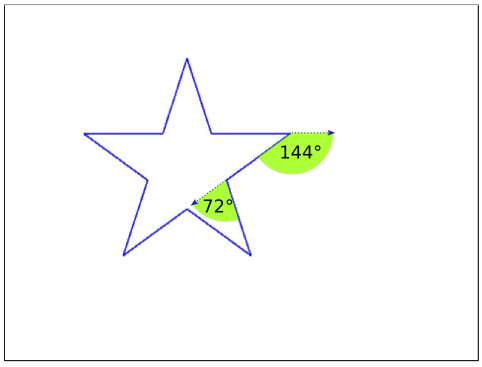
Aufgabe 4.19

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

a)



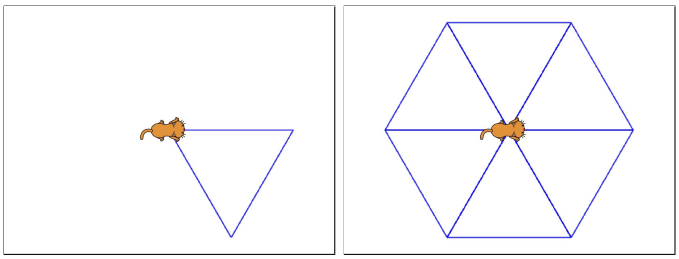
b)



Tipp: ein Tastendruck = *zwei* Linien!

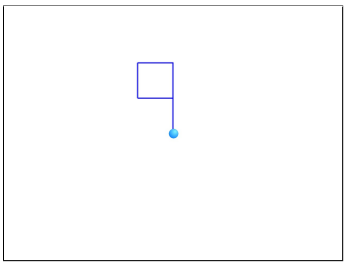
Aufgabe 4.20

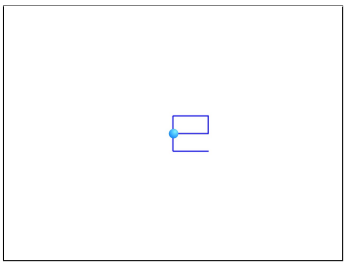
Mit den beiden Skripten rechts malt Scratch (nach dem Aufräumen durch Klick auf die grüne Flagge) auf Druck der Leertaste immer wieder das gleiche Dreieck (Bild unten links). Wie musst du das Skript für die Leertaste ergänzen, damit durch wiederholtes Drücken der Leertaste das Bild unten rechts entsteht?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

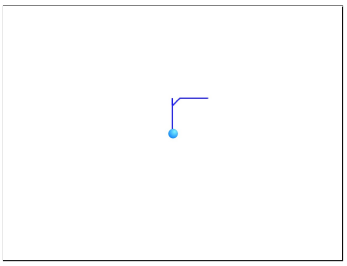


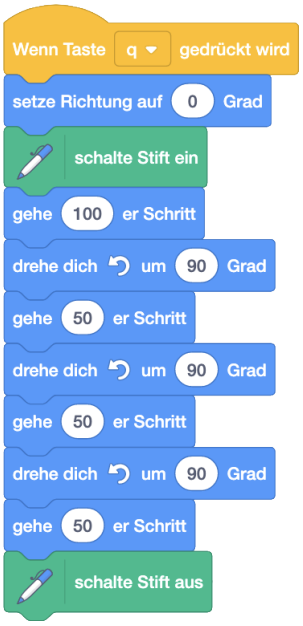
Aufgabe 4.21

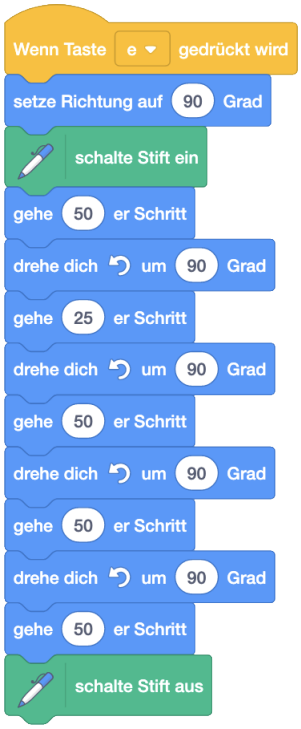
Erstelle ein Skript, mit dem du deine Initialen (Anfangsbuchstaben deines Vor- und Nachnamens) malen kannst (gross oder klein)! Unten sind drei Beispiele. Der Punkt zeigt jeweils, wo die Zeichnung beginnt.

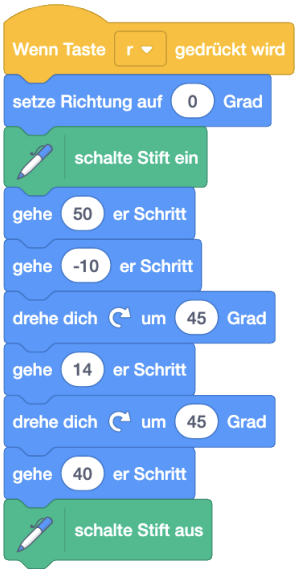












Hier kann man die Lernenden auch zuerst herausfinden lassen, was mit dem Skript gezeichnet wird.

Kapitel 5

Wiederholungen

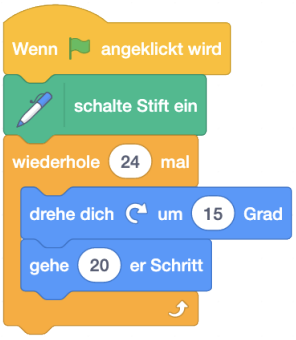
Hier erfährst du, wie du Befehle und Befehlsfolgen mehrfach ausführen kannst. Dadurch entstehen auf einfache Weise "komplizierte" Abläufe, die du aber gut planen musst. Du erfährst, wie du Objekte miteinander interagieren lassen kannst.
Die Blöcke, die du dabei neu kennenlernst:

Die Begriffe, die hier neu eingeführt werden:

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

5.1 Dinge mehrmals tun

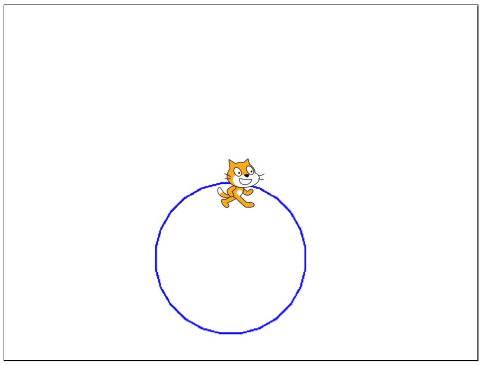
Erstelle dieses Skript und klicke auf die grüne Flagge über der Bühne. Was passiert?



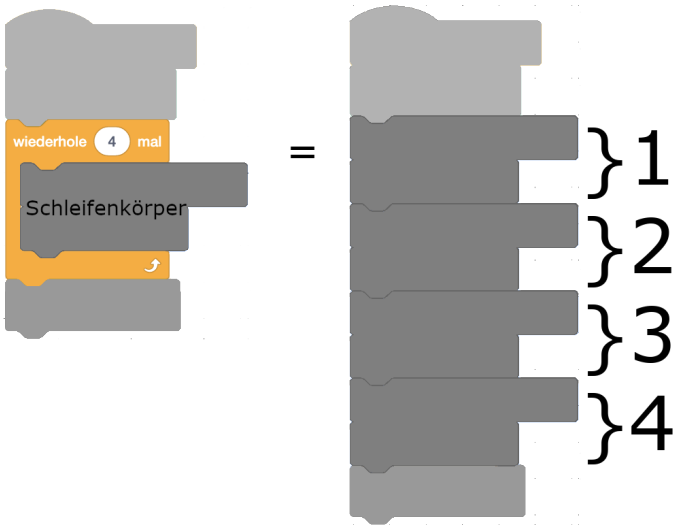
Wenn es dir zu schnell geht, kannst du den Block 

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!

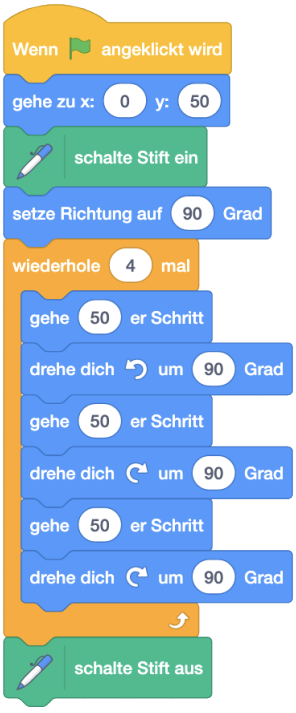
In dieser Aufgabe wird geprüft, ob die Lernenden das Konzept der Wiederholung in Scratch richtig verstanden haben. Es gibt einige verbreitete Fehlvorstellungen zu diesem Konzept. Wenn diese früh angesprochen werden, fällt das Programmieren in den folgenden Aufgaben leichter. Die Aufgabe kann auch im Klassenverband besprochen werden. Wichtig ist, dass die Lernenden jeweils nochmals in eigenen Worten erklären, wie sie auf die Lösung kommen.

Aufgabe 5.3

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

a)

b)

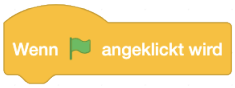


c)

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

Wenn die Lernenden schon etwas mehr Übung mit Winkeln haben, können hier auch Zeichnungen mit anderen Winkeln gezeigt werden. Die Lernenden können auch ausprobieren, sich gegenseitig Aufgaben zu stellen.

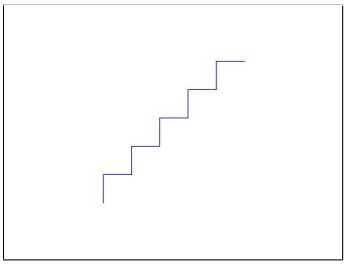
Aufgabe 5.4

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

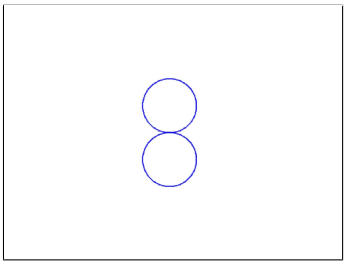
und dem Wiederhole-Mehrmals-Befehl.

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

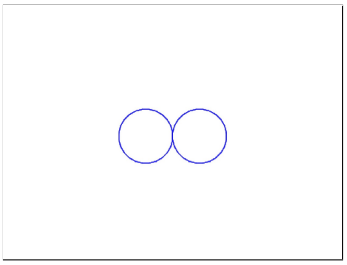
Treppe



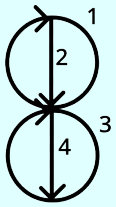
Acht



Liegende Acht



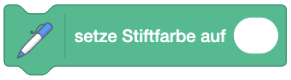
Es kann sein, dass Lernende hier den folgenden Plan vorschlagen:



Dieser ist sehr schwierig umzusetzen, da wir zuerst ausrechnen müssten, wie gross der Durchmesser des Kreises ist. Wenn die Lernenden nicht selbst auf diesen Plan kommen, kann man auch am Schluss nochmals fragen, was sie von diesem Plan halten, um zu diskutieren dass nicht alle Pläne gleich gut umsetzbar sind.
Schnellere Lernende können sich hier auch schon überlegen, wie man die Kreismuster aus Aufgabe 5.9 erstellen kann. Die Lösung mit verschachtelten Schleifen ist zwar nicht ganz ideal, kann aber hier das Verständnis für Schleifen vertiefen.

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?

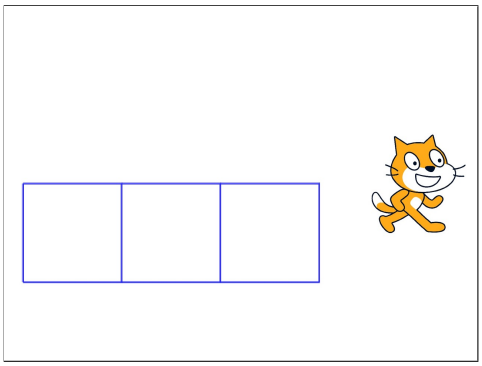
Zusatzaufgaben:

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

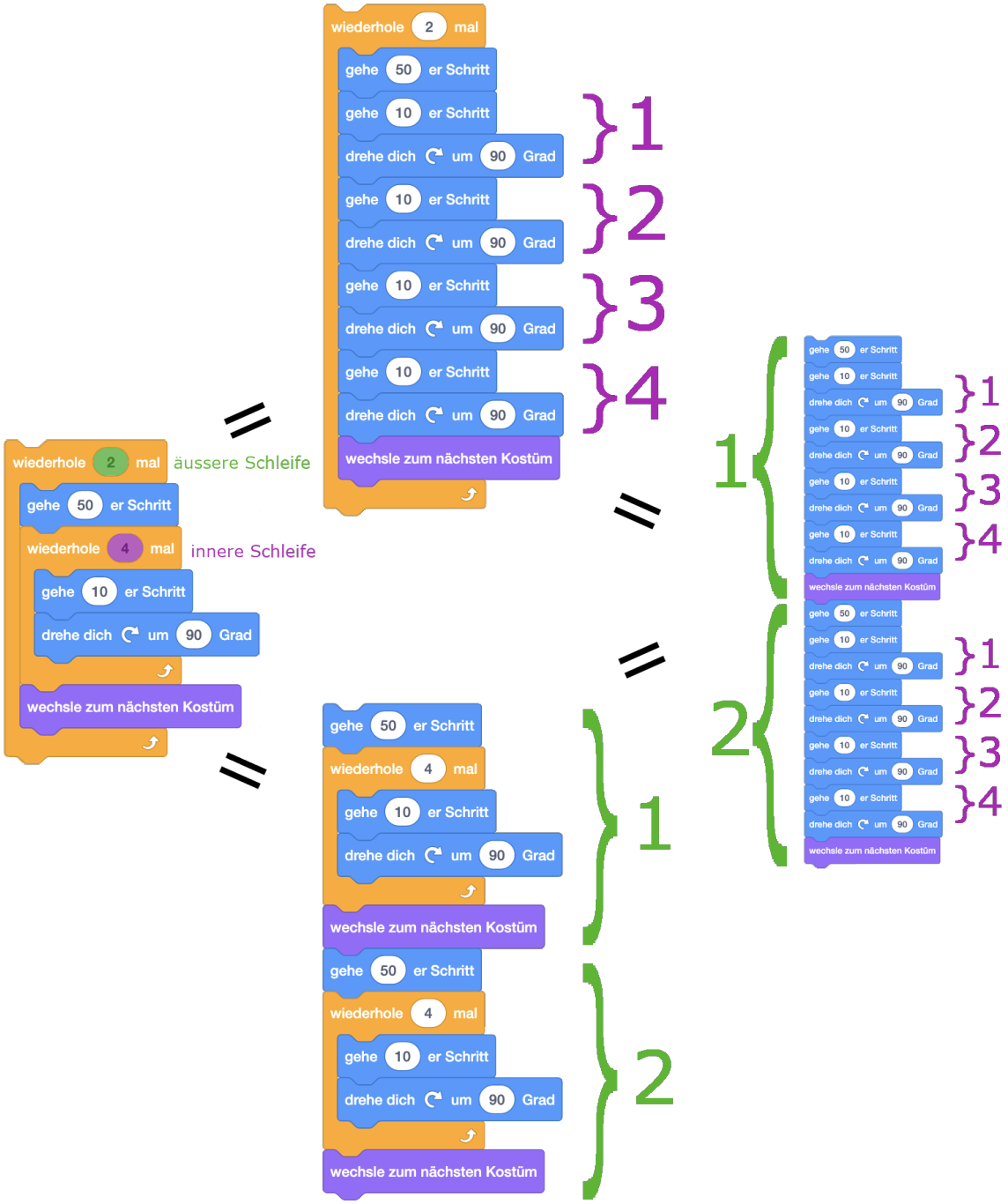
5.2 Verschachtelte Schleifen

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.

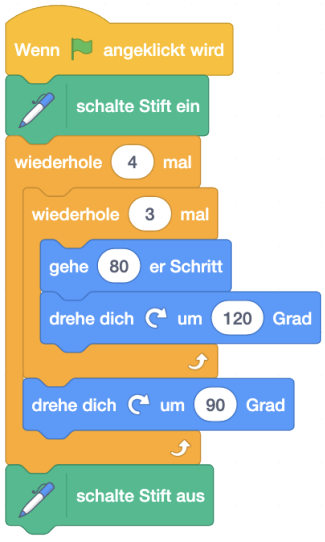
Dieses Konzept ist für Kinder oft noch schwierig zu verstehen. Wir beschränken uns daher hier darauf, dass die Lernenden fremde Programme mit zwei ineinander verschachtelten Schleifen lesen und verstehen können. Um verschachtelte Schleifen selbst implementieren zu können wird im nächsten Kapitel das Konzept des eigenen Blocks eingeführt, welches das Programm für die Kinder leichter lesbar macht und dem wichtigen Konzept der Modularität in der Informatik gerecht wird.
In der Programmierung werden verschachtelte Schleifen häufig benutzt, um zum Beispiel die Elemente einer Matrix (bzw. eines verschachtelten Arrays) durchzugehen.

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!

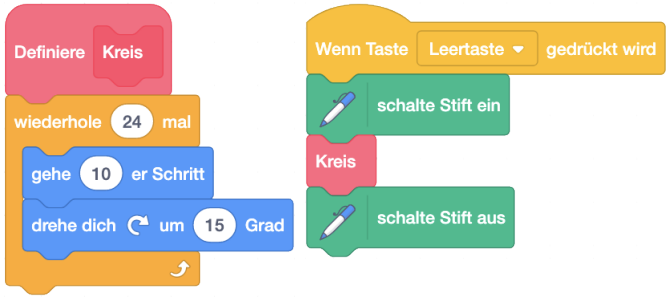
a)



b)

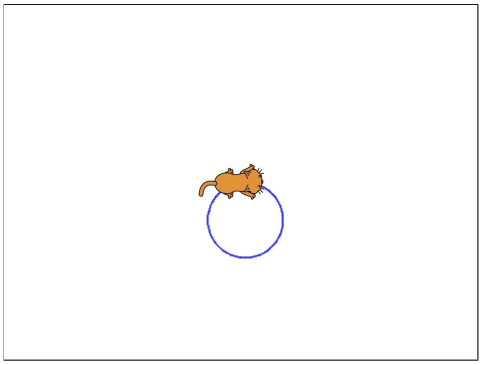
5.3 Eigene Blöcke

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:

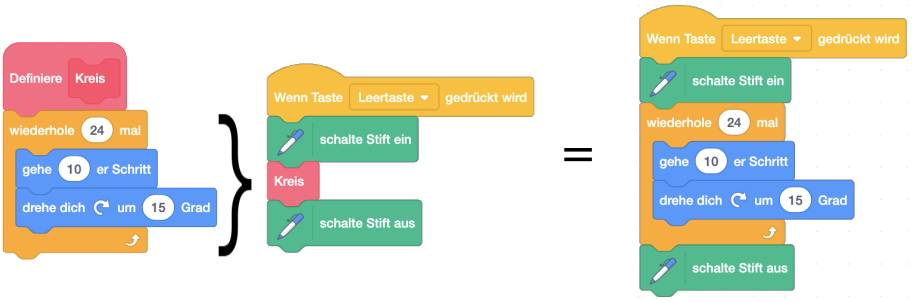


Was passiert, wenn du die Leertaste drückst?

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:



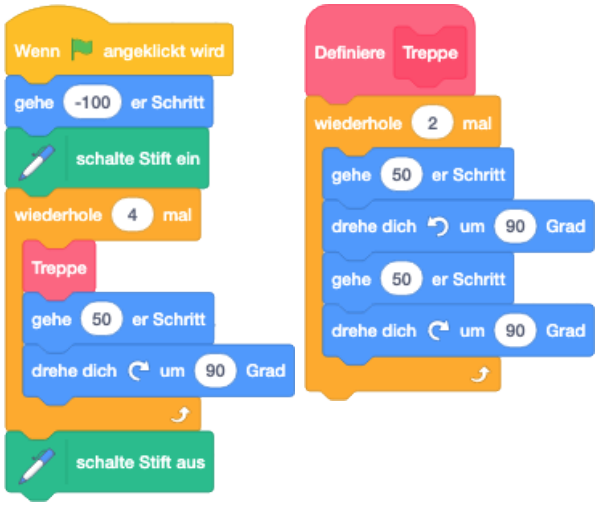
Unter dem "Definiere"-Hut schreibst du das Skript, das überall dort ausgeführt werden soll, wo du den eigenen Block einfügst. Dadurch kannst du im Programmierbereich Platz sparen und dein Programm wird besser lesbar, wenn du deinen Blöcken gute Namen gibst. Sobald du einen neuen Block definiert hast, 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)

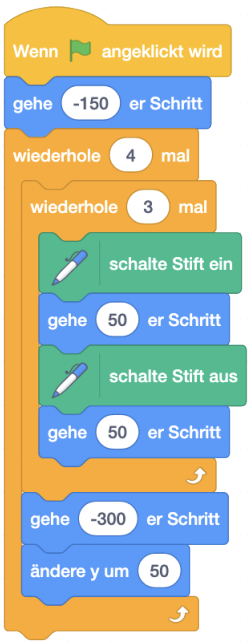


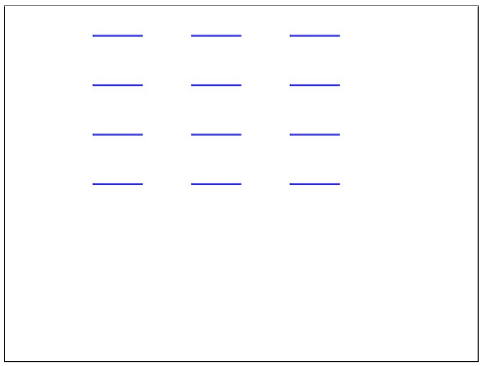
b)

Aufgabe 5.8

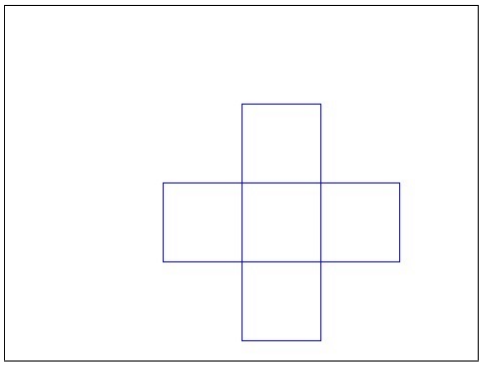
Scratch zeichnet mit den beiden Skripten unten die jeweiligen Bilder. Definiere eigene Blöcke, um das Programm lesbarer zu machen!

a)



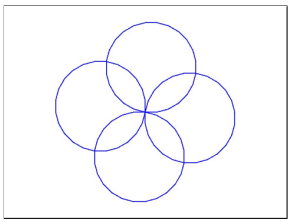


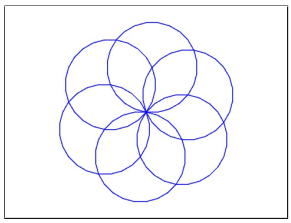
b)

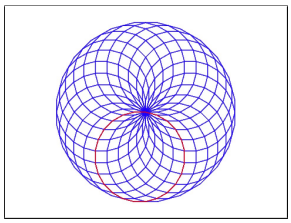


Aufgabe 5.9

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!

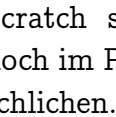
a) 
4 Kreise

b) 
6 Kreise

c) 
24 Kreise

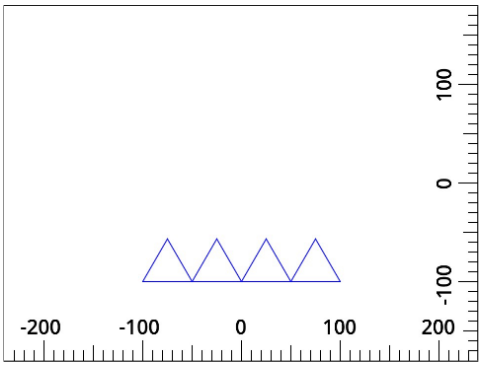
Ablauf: Wiederhole mehrmals diese beiden Schritte:

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

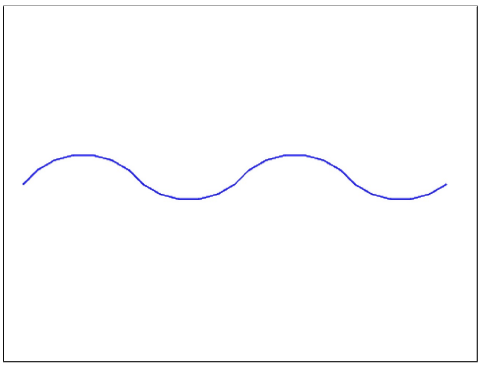
Lass verschiedene Figuren tanzen! Verwende dabei eigene Blöcke, um gewisse Teile des Tanzes zu definieren (zum Beispiel einen Salto machen oder nach rechts hüpfen).

Damit die Lernenden bei dieser Aufgabe nicht zu viel Zeit damit verbringen, die verschiedenen Kostüme und Klänge aus der Scratch-Bibliothek auszuprobieren, kann man hier ein Zeitlimit setzen. Ausserdem kann man die Anzahl Figuren im Endprojekt begrenzen. Am Ende dürfen alle ihre programmierte "Choreografie" der Klasse zeigen.

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 

.

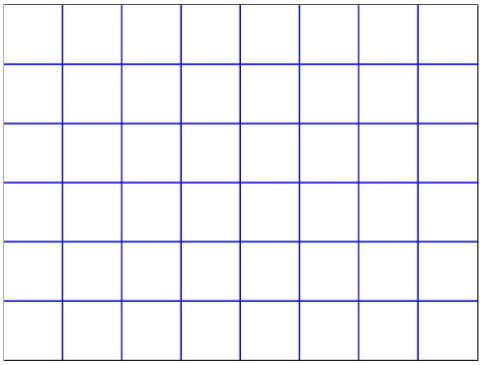


Ablauf:
Drehe dich zuerst um 45° ↻ und wiederhole dann mehrmals diese beiden Schritte:

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

Aufgabe 5.13

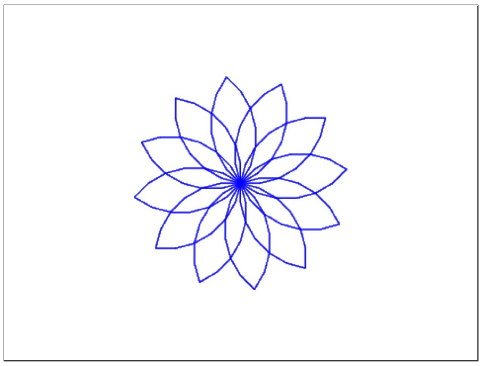
Erstelle ein Skript, das die Bühne mit diesem Gittermuster bemalt (8 Kästchen in der Breite, 6 in der Höhe. Die Gitterlinien haben also jeweils 60 Schritte Abstand voneinander). Nach aussen ist das Gitter offen, am Bühnenrand sollen also keine Linien gezeichnet werden! Versuche hier, selbst den Ablauf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

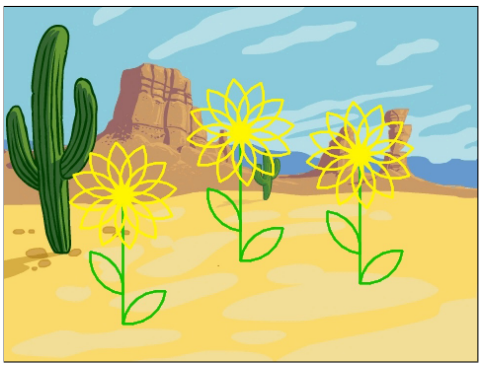
1. Male ein Blütenblatt
2. Drehe dich

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

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

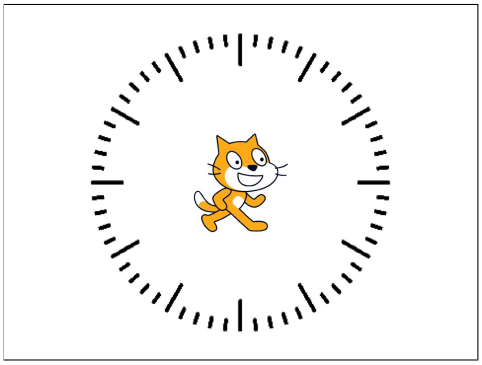
Den gleichen Ablauf kannst du verwenden, um Blumen in der Wüste wachsen zu lassen! Du musst dabei nur vorher jeweils noch zwei Blätter und einen Stiel

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



Aufgabe 5.15

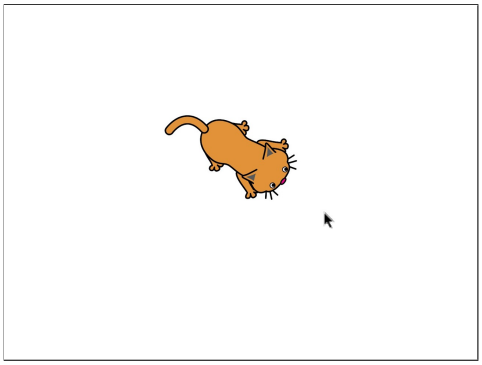
Male das Ziffernblatt einer Uhr! Stiftfarbe und Stiftdicke kannst du frei wählen. Versuche hier zuerst selbst, den Ablauf herauszufinden!



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: 

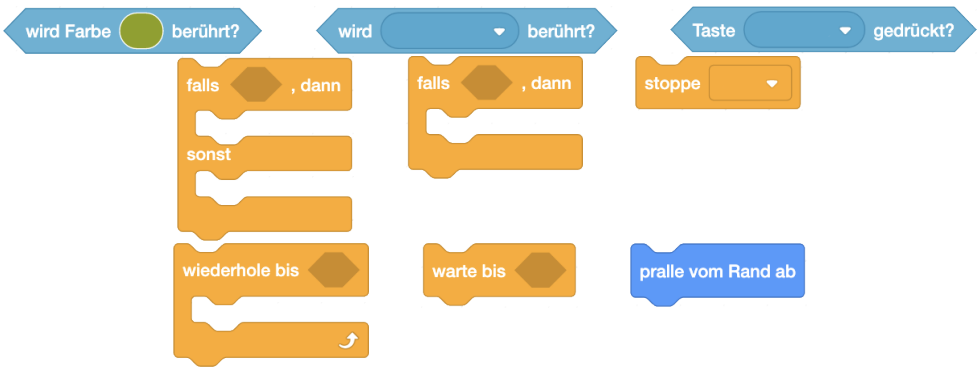
Lade eine Maus als zweites Objekt und erstelle ein Skript mit dem die Maus beim Klick auf die grüne Flagge selbst vor Scratch davonlaufen kann (zum Beispiel im Kreis)! Gib Scratch das "Verfolgungsskript" und wähle im Menü des Befehls statt Mauszeiger die Maus aus (nachdem du das Objekt sinnvoll umbenannt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

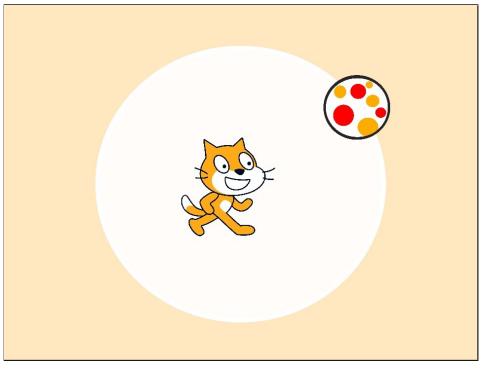


Die Begriffe, die hier neu eingeführt werden:

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

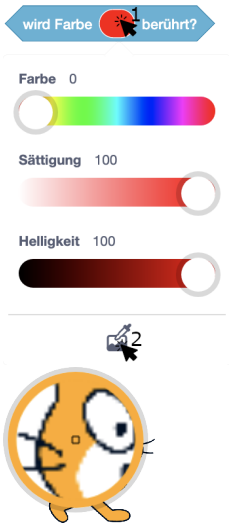
6.1 Bedingungen prüfen

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

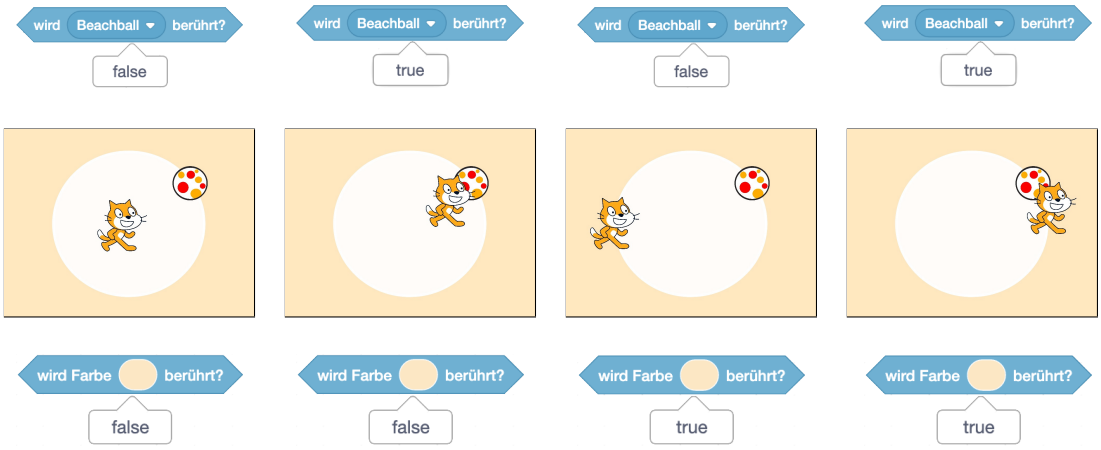


Ziehe diese drei Blöcke in den Programmierbereich von Scratch und klicke sie an. Versuche bei allen drei Blöcken, zwei verschiedene Antworten zu bekommen!

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.

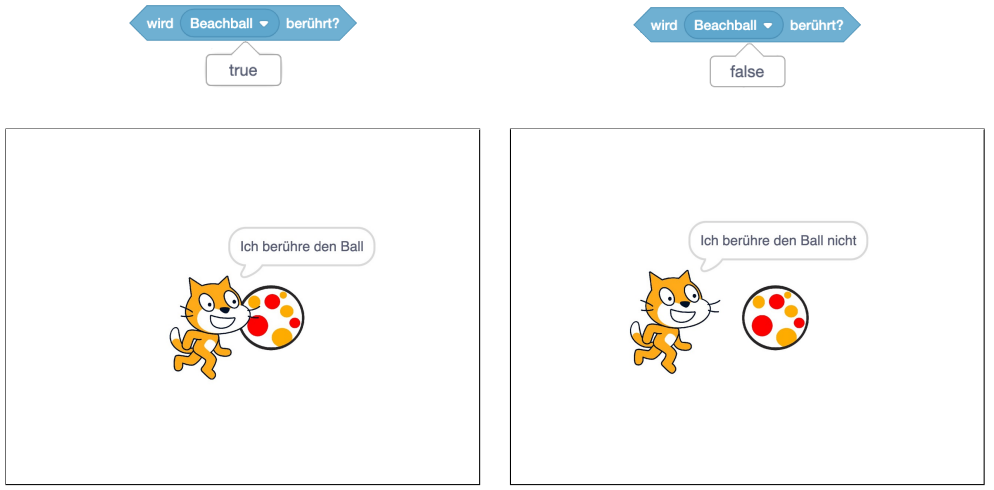


Jeder Block der Form [redacted] ist eine *Bedingung*. Der Text in der Bedingung ist eine Ja-Nein-Frage. Bei jedem Anklicken der Bedingung wird die Ja-Nein-Frage der Situation entsprechend beantwortet. Antwort “true” bedeutet “Ja, das stimmt, die Bedingung gilt!”. Antwort “false” heisst “Nein, das stimmt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

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

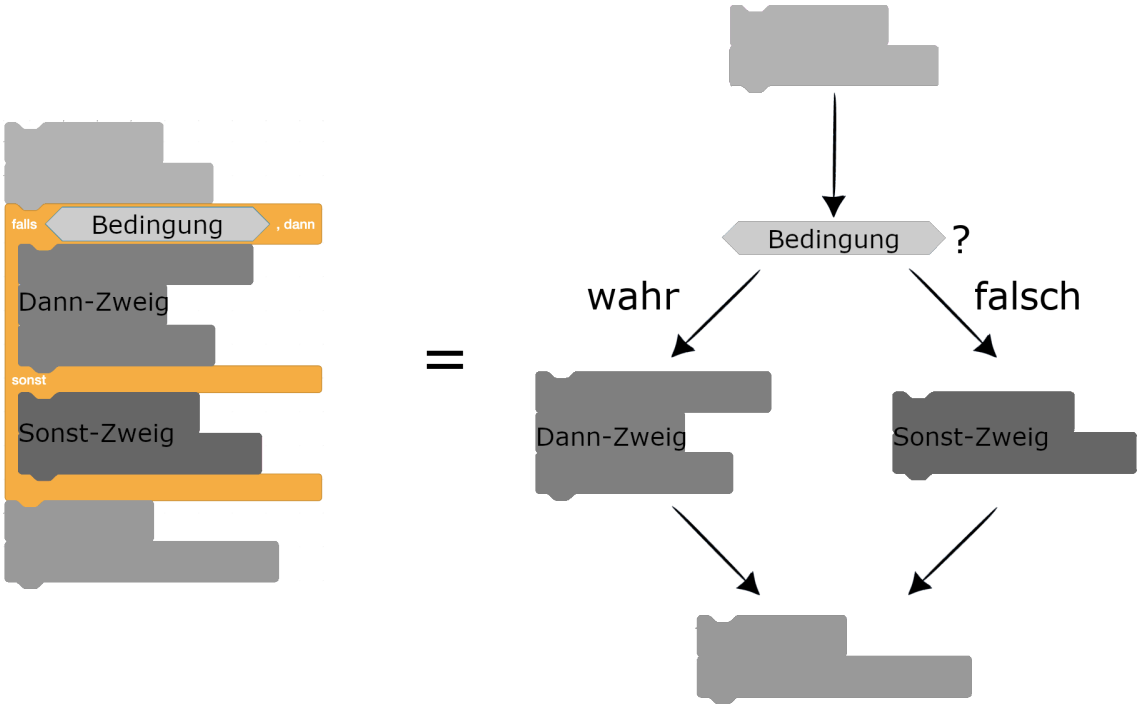
Lade wieder den Beachball als zusätzliches Objekt und erstelle dieses Skript für Scratch:
Ziehe dann Scratch auf der Bühne herum und drücke die Leertaste. Was passiert?

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

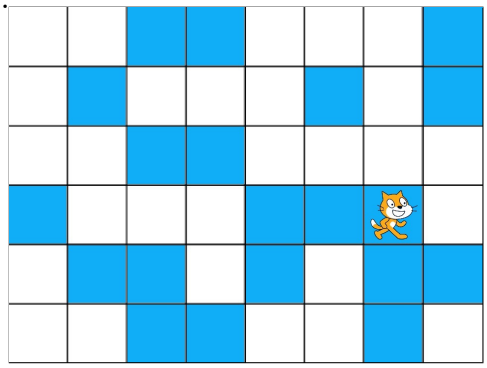
Zeichne das Flussdiagramm für den Wiederhole-Fortlaufend-Befehl:

Pfeile der entsprechenden Form haben die Lernenden noch nicht gesehen, daher sollte man die Lernenden hier experimentieren lassen.

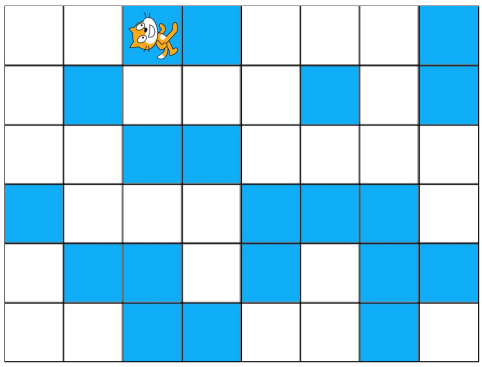
Aufgabe 6.3

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.

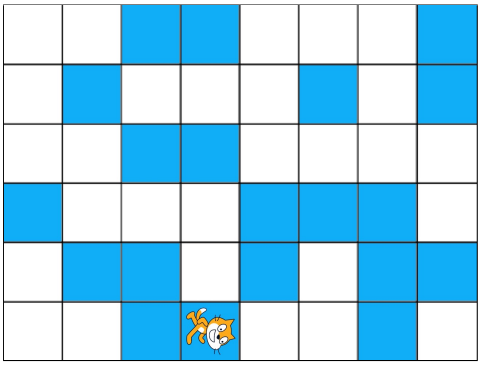
1.



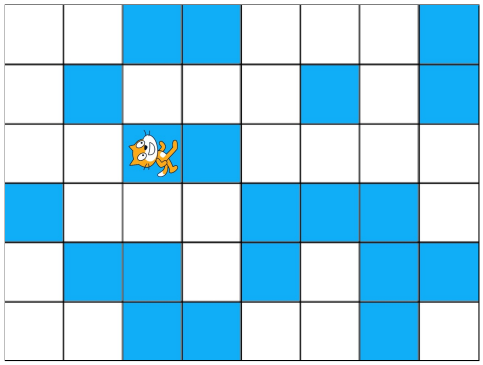
2.



3.

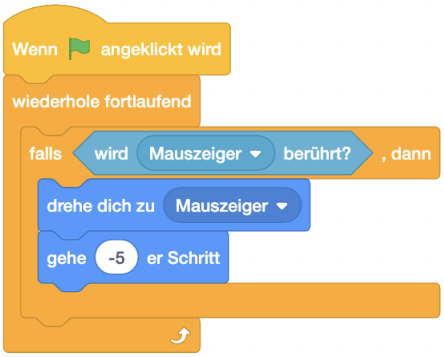


4.

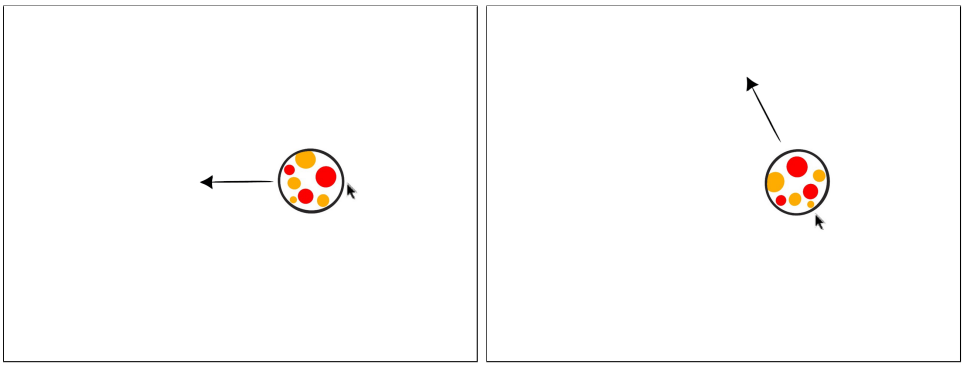


6.3 Dinge tun, falls...

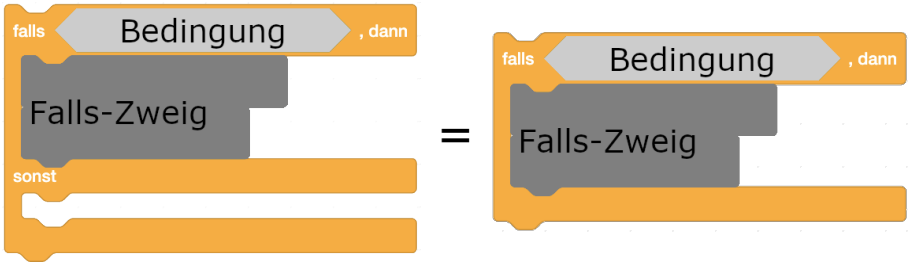
Erstelle für den Ball dieses Skript und klicke die grüne Flagge an! Was passiert, wenn du mit dem Mauszeiger den Ball berührst?



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.



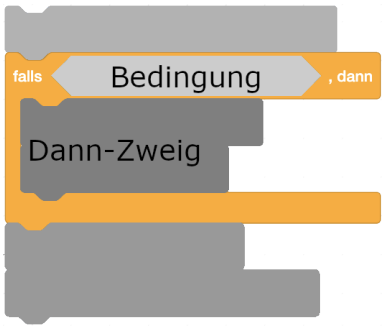
Den *Falls-Dann*-Befehl kannst du verwenden, wenn du nur etwas tun willst, wenn die Bedingung gilt. Wenn die Bedingung nicht gilt, wird die Befehlsfolge im Dann-Zweig nicht ausgeführt. In beiden Fällen geht es danach mit dem nächsten Befehl im Skript weiter. Im gezeigten Beispiel soll der Ball nichts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

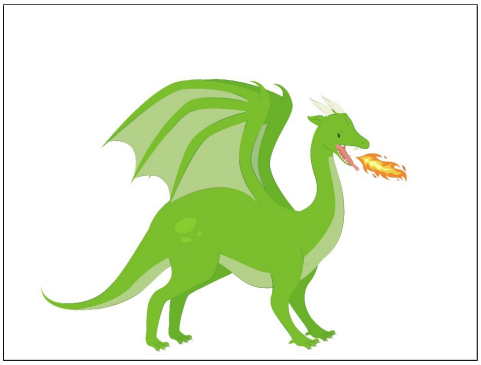
Aufgabe 6.4

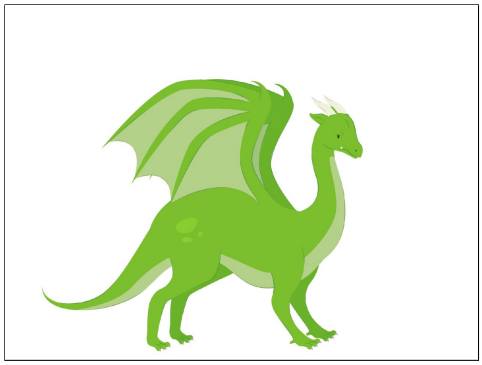
Zeichne das Flussdiagramm für den Falls-Dann-Befehl:

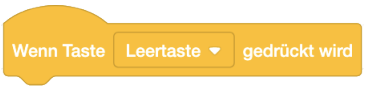


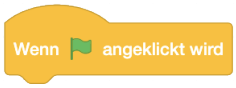
Aufgabe 6.5

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 

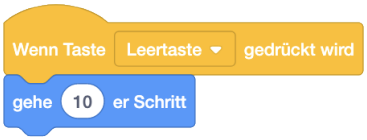
sondern mit 

!

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

Aufgabe 6.6

Erkläre die Unterschiede zwischen diesen beiden Skripten:

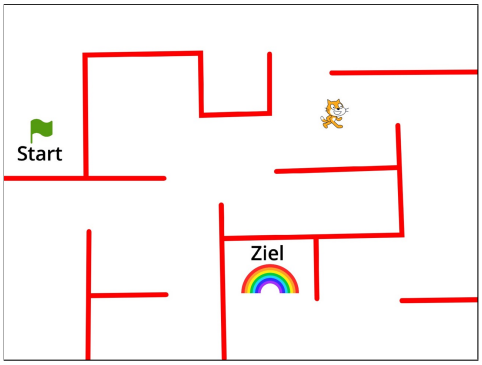


Was passiert, wenn die Leertaste gedrückt wird? Was passiert, wenn das Skript angeklickt wird?

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.



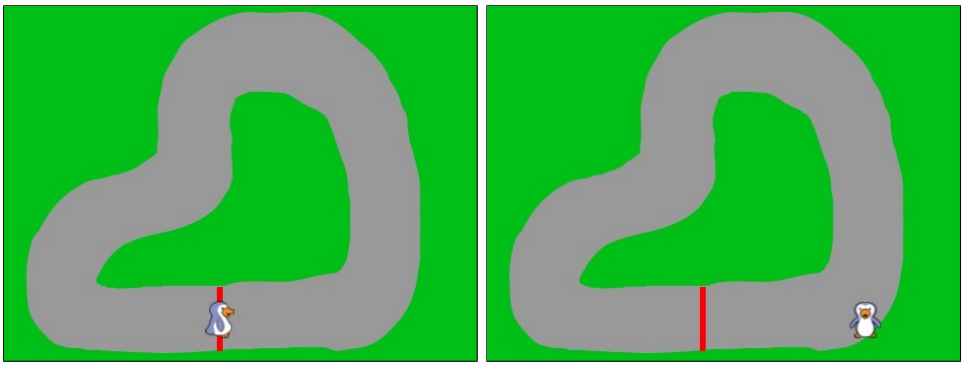
Tipps: Um das Spiel zu beenden, benutze den Befehl 

.

In dieser und der folgenden Aufgabe können die Kinder kreativ sein und ihren Projekten eine eigene Note verleihen. Sie dürfen die Objekte selber aussuchen, das Labyrinth und die Rennbahn selbst zeichnen und auswählen, wie ein Gewinn oder Fehlschlag visualisiert werden soll.
Das Laufskript ist in dieser Aufgabe vorgegeben, weil sich die Lernenden auf den Rest des Programms konzentrieren sollen. Mit einigen Tastaturen kann man mit diesem Skript durch gleichzeitiges Drücken von zwei Pfeiltasten ein Objekt auch diagonal laufen lassen.

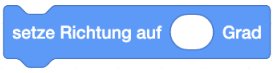
Aufgabe 6.8

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 

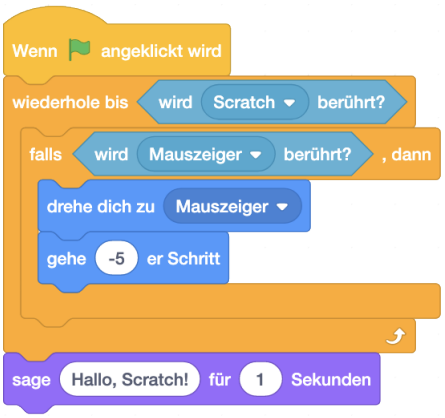
! Zu Beginn ("Grüne Flagge angeklickt") solltest du das Objekt mit Hilfe der Befehle

und


in eine sichere Startposition bringen.

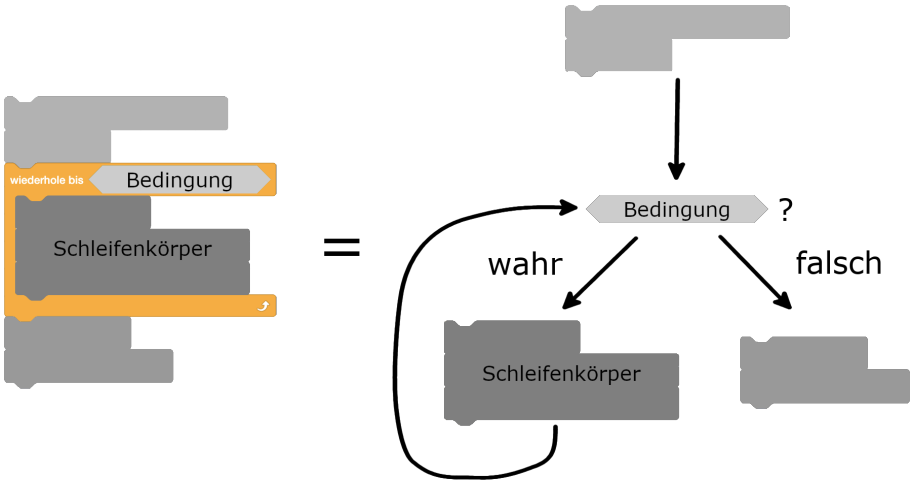
6.4 Dinge tun, bis...

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?

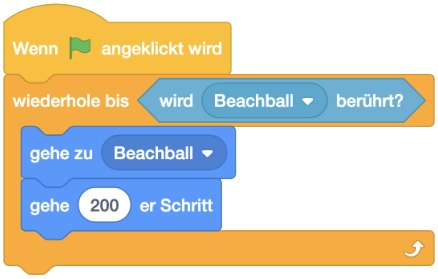


Scratch wartet auf den Ball, während der Ball sich vom Mauszeiger wegbewegt. Ist der Ball bei Scratch angekommen, begrüssen sich die beiden.

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.



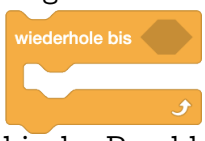
Innerhalb des Schleifenkörpers wird die Bedingung aber nicht überprüft. Es kann also durchaus vorkommen, dass die Bedingung zwischendurch erfüllt ist, aber die Wiederholung nicht aufhört, weil die Bedingung direkt vor der nächsten Wiederholung nicht mehr gilt.
Beispiel: In diesem Skript wird die Wiederholung 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):

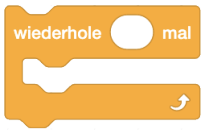


Nimm den *Warte-Bis Befehl*, wenn du *nichts* tun willst, bis die Bedingung erfüllt ist:

Aufgabe 6.9

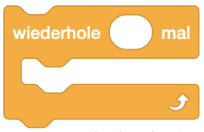
Verändere das Rennbahn-Spiel aus Aufgabe 6.8!

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 Anzahl 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.

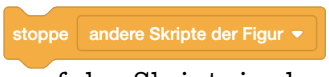
Lassen sich die beiden Erweiterungen gleichzeitig benutzen? Unterscheide dabei, ob die Rundenzahl zurückgesetzt werden soll, wenn der Rand berührt wird, oder nicht. Überlege dir auch, was passiert, wenn du das Objekt nach einer Randberührung wieder zurück auf die Start-/Ziellinie setzt. Programmiere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

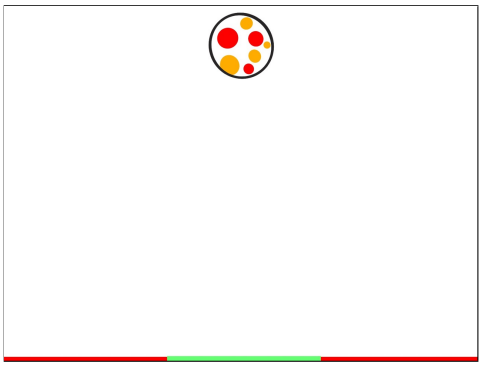
Tipp: Mit dem 

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.

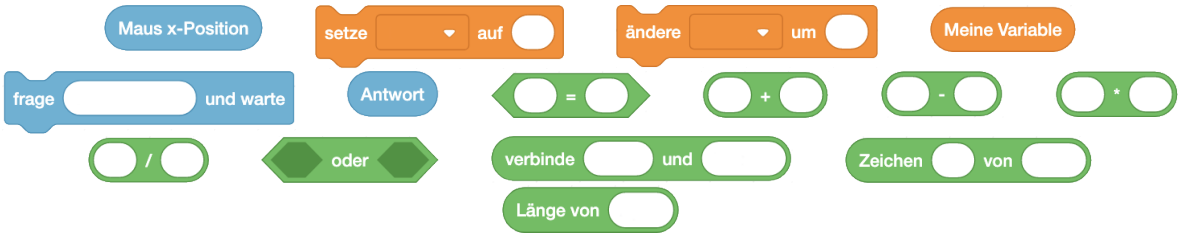
Statt der ausführlichen Beschreibung kann man den Lernenden das Spiel auch im Präsentationsmodus vorzeigen. Auf die noch nicht bekannten Befehle kann man aufmerksam machen, oder die Lernenden selbst ausprobieren und entdecken lassen. Die Schläger-Bewegung zu implementieren ist mit den bis jetzt bekannten Blöcken nicht ganz einfach. Im nächsten Kapitel werden wir dazu noch eine bessere Lösung finden. Wenn die Lernenden diese Lösung aber schon jetzt entdecken, ist das natürlich auch ok.
Aufgepasst wenn der Schläger selbst gezeichnet wird: Um das Kostüm selbst zu zeichnen kann man durch Halten der Shift-Taste eine horizontale Linie zeichnen. Dadurch hat aber das Bild eine Höhe von 0 Pixeln und die Linie wird nicht angezeigt. Mit dem Umwandeln in eine Rastergrafik kann dies behoben werden.

[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:

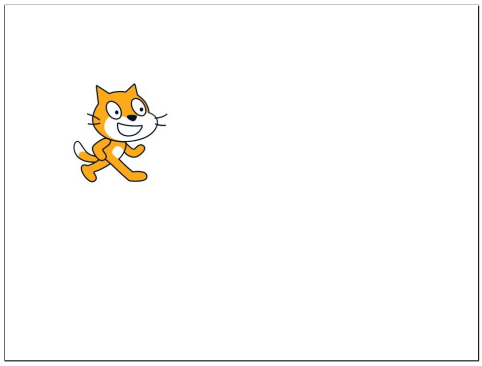
- Wert-Block
- Variable
- Operator
- Ausdruck
- formaler und tatsächlicher Parameter

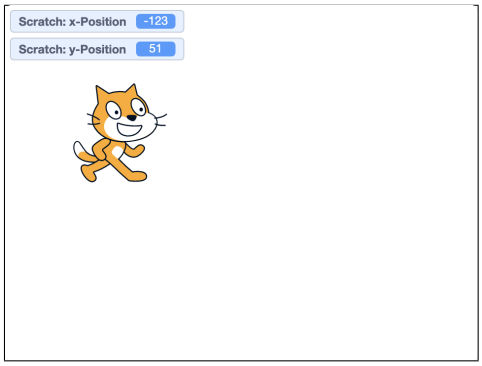
7.1 Wert-Blöcke

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 ( ) 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.



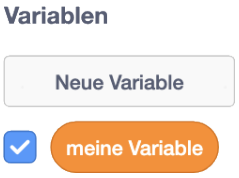


Aufgabe 7.1

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".



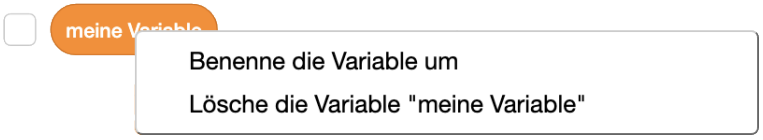
Bewege nun Scratch mit den Pfeiltasten ← und → über die Bühne und beobachte den Wert für "meine Variable", der oben links in der Bühne angezeigt wird.

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:

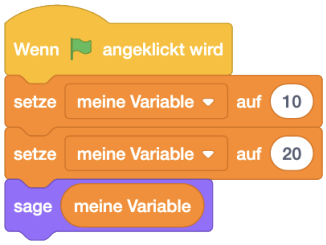


Was wäre ein besserer Name für die Variable im Einführungsbeispiel oben?

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?

In dieser Aufgabe werden einige häufige Fehlannahmen überprüft. Die Lernenden können so das Konzept der Variable festigen und bei Unklarheiten Fragen stellen, bevor sie beginnen zu programmieren.

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?

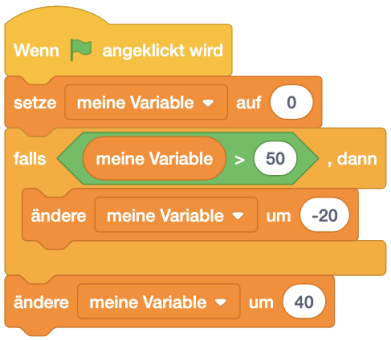
Scratch sagt nach 1000 Schritten “Ich habe mein Trainingsziel erreicht!”.

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

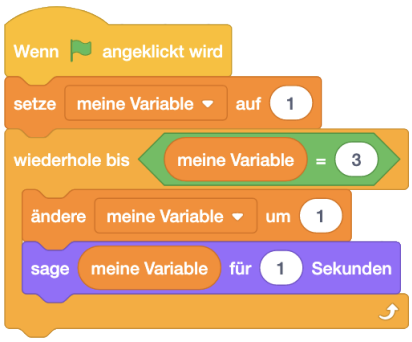
Schreibe für die folgenden Skripte auf, welche Werte die verwendeten Variablen am Ende des Skriptes haben. Stelle dazu am besten eine Tabelle auf mit allen Variablen und ihren Werten. Wenn der Wert einer Variable verändert wird, streichst du den alten Wert durch und schreibst den neuen Wert auf. Schreibe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

1.



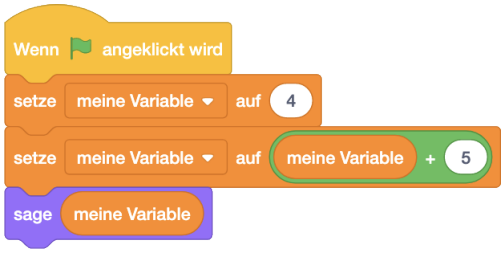
2.

3.



4.

5.

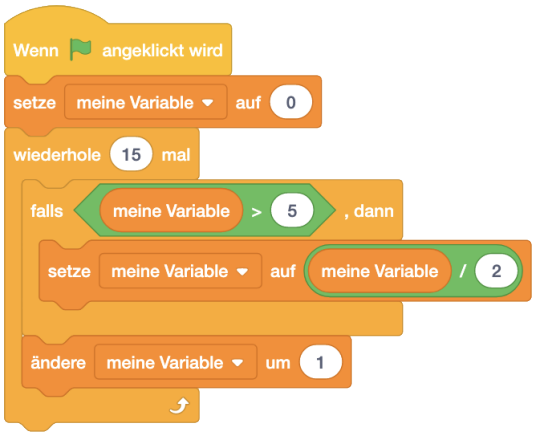


Auch in dieser Aufgabe werden wieder einige häufige Fehlannahmen überprüft.

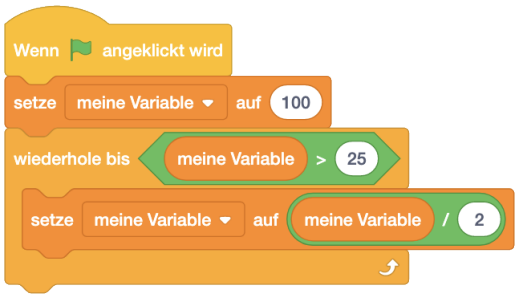
Aufgabe 7.6

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

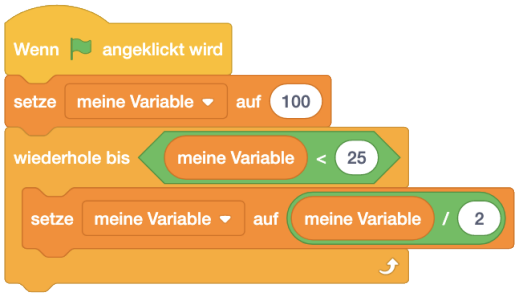
1.



2.



3.



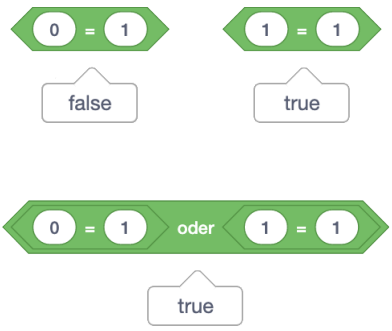
Aufgabe 7.7

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:



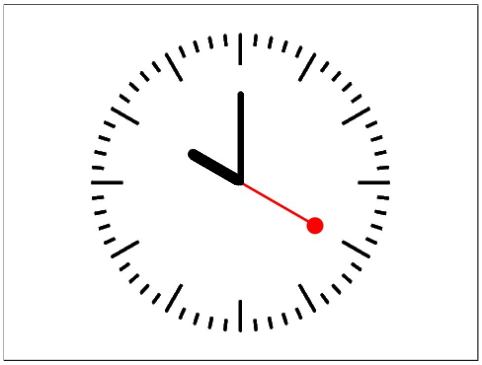
Aufgabe 7.8

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)!

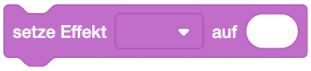


Zusatz:

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 

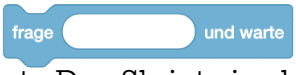
   verändern.

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

Wenn das Thema Proportionalität noch nicht behandelt wurde, sollte bei dieser Aufgabe zuerst erklärt werden, wie man von Stunden / Minuten / Sekunden zu Grad umrechnet.
Bei den Zusatzaufgaben dürfen sich die Lernenden auch andere Ereignisse ausdenken, die zu bestimmten Zeiten passieren, und diese animieren.

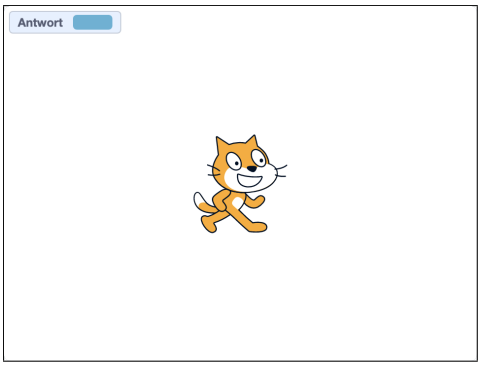
Aufgabe 7.10

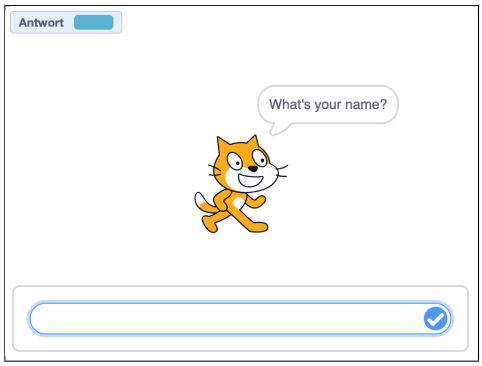
Stell dir vor, die Macher von Scratch würden den Wiederhole-Mehrmals Befehl entfernen, weil er kaputt war. Kannst du den Befehl durch andere Befehle ersetzen? Verwende dazu Variablen und Operatoren. Zeige deine Umsetzung am folgenden Beispiel:

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!

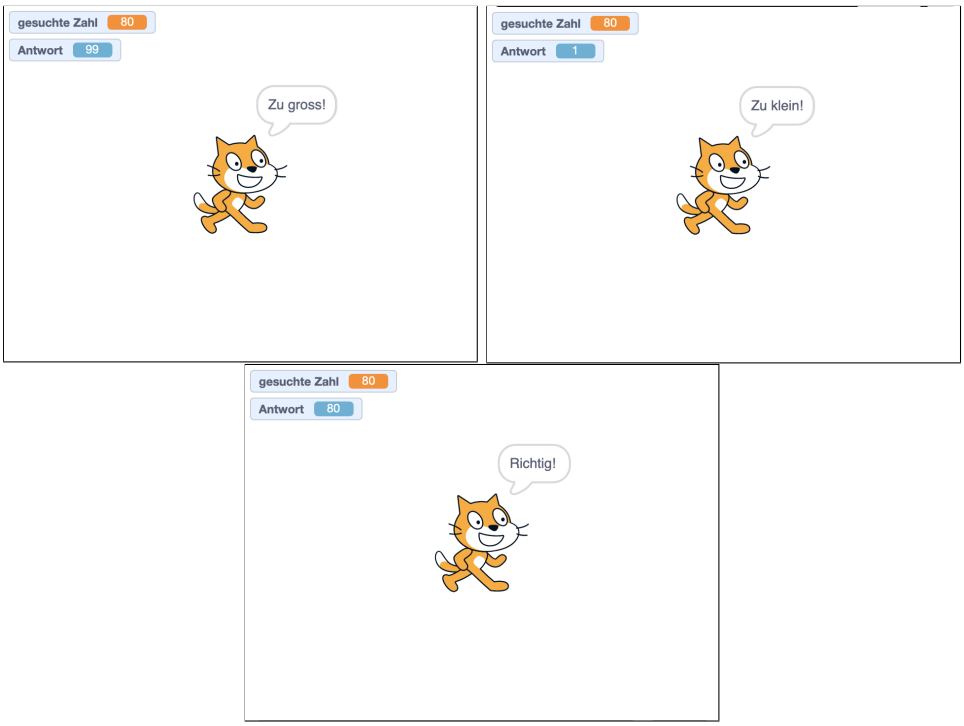





Antwort
What's your name?
Scratch
Antwort Scratch

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.

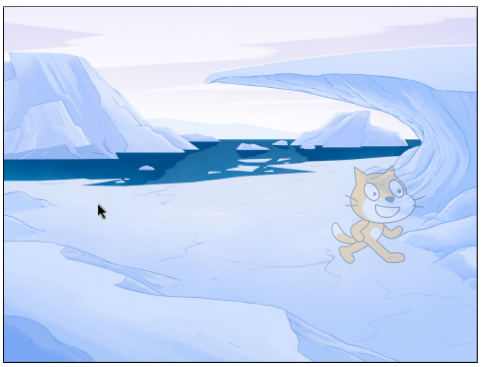
Aufgabe 7.12

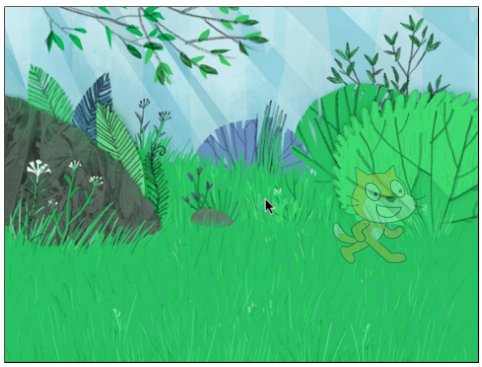
Erstelle ein Programm, in dem der Benutzer Scratch suchen muss. Scratch versteckt sich an einer zufälligen Position auf der Bühne. Bei jedem Mausklick des Benutzers soll Scratch überprüfen, wie weit der Mausklick von Scratch entfernt war und dies in einer Variable speichern, die auf der Bühne sichtbar 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

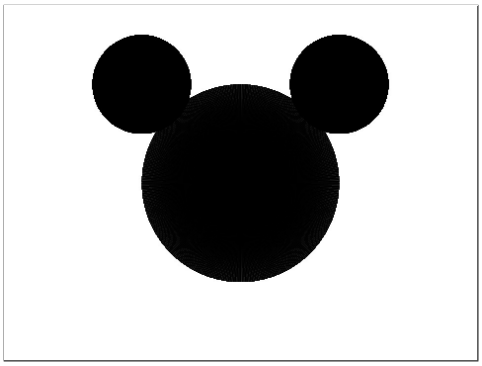
Erstelle das folgende Programm. Erstelle dazu einen eigenen Block und füge ein Eingabefeld für Zahl oder Text hinzu. Benenne dieses Eingabefeld dann wie in dem gezeigten Programm. Den Wertblock des Parameters kannst du einfach aus dem Definitionsblock herausziehen, um ihn in der Definition zu verwenden.

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.

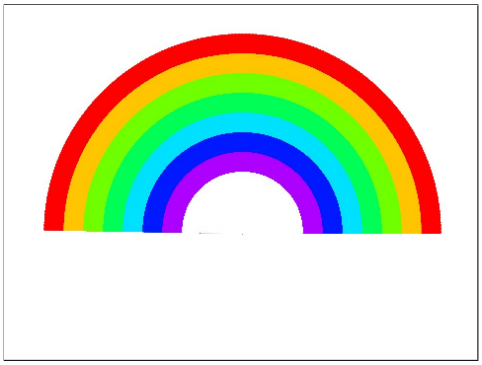
Scratch zeichnet den Umriss einer weltbekannten Maus mit einem grossen und zwei kleineren Kreisen:



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?

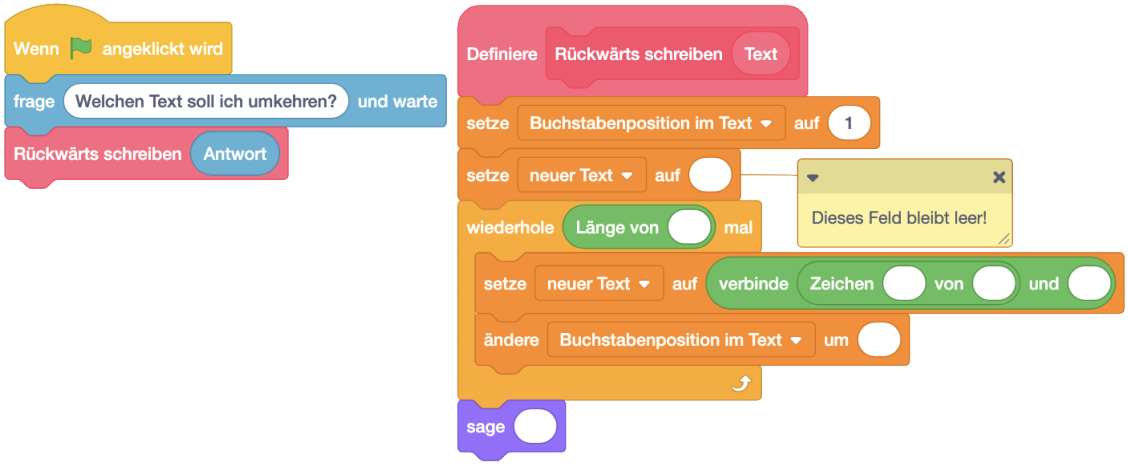
5. 

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:


.

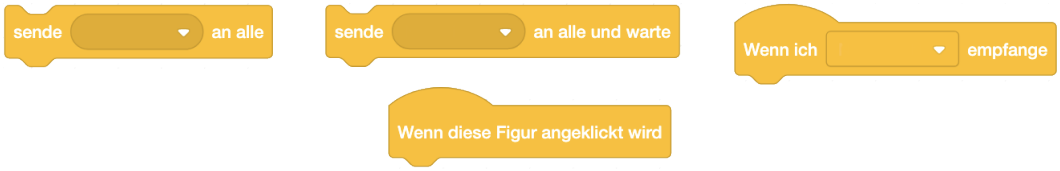
Zusatzaufgabe: Erweitere die Verschlüsselung des Textes! Du kannst zum Beispiel jedes 'a' im Text durch ein 'e' ersetzen und umgekehrt. Achte darauf, dass du den Text danach immer noch entschlüsseln kannst!

Scratch ist nicht ideal, um Ver- und Entschlüsselung zu programmieren, da es etwas umständlich ist, einen Text zu bearbeiten. Ausserdem haben die Methoden in Scratch keinen Rückgabewert, daher kann man das Resultat nur mit dem Sage-Block sichtbar machen. Daher ist in dieser Aufgabe schon ein Code-Skelett vorgegeben, das nur noch mit den richtigen Blöcken ausgefüllt werden muss. Die fehlenden Blöcke können auch gezeigt werden, damit die Lernenden nur noch zuordnen müssen.

Kapitel 8

Kommunikation

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



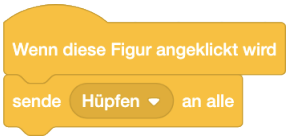
Die Begriffe, die hier neu eingeführt werden:

- 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?

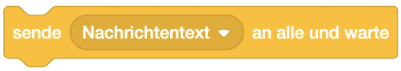
Wenn du den Knopf anklickst hüpft Scratch kurz nach oben. Wenn der Knopf mehrmals kurz nacheinander angeklickt wird, bleibt Scratch oben.

Der Befehl 

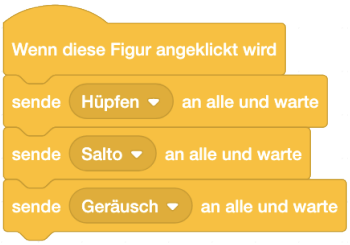
sendet allen (den Objekten und der Bühne) eine *Nachricht* mit dem Nachrichtentext. Die Nachricht startet bei allen Empfängern die Skripte (falls vorhanden) unter dem Hut 

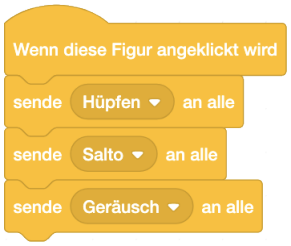
mit dem gleichen Nachrichtentext.

Was ist der Unterschied zwischen dem Befehl 

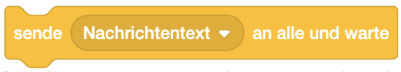
und dem Befehl 

? Probiere es mit dem Programm unten aus:





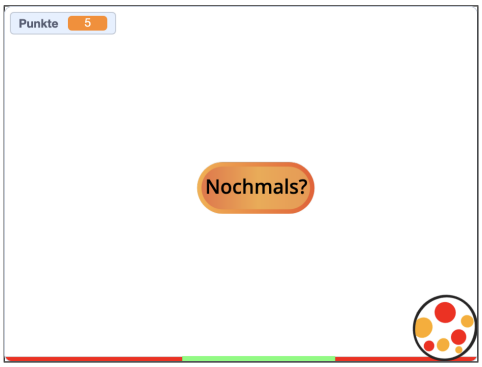
Der Befehl 

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.

In dieser Aufgabe können die Lernenden auch gemeinsam ein Projekt erstellen. Alle erstellen ein Programm, mit dem sie sich kurz vorstellen, zum Beispiel indem sie ihr jeweils ihr Lieblingsessen, Hobbies und Lieblingstier nennen und animieren. Das Programm beginnt mit einem "Wenn ich Nachricht empfange" Block. Die programmierten Figuren werden gespeichert und am Schluss werden die einzelnen Programme nacheinander als Remix zusammengesetzt. Alternativ können die programmierten Figuren auch heruntergeladen werden und per USB-Stick auf einen Computer geladen werden. Dort wird dann direkt ein Programm mit allen Figuren erstellt.
Ein Beispiel findet sich unter https://scratch.mit.edu/projects/236452266/editor/.

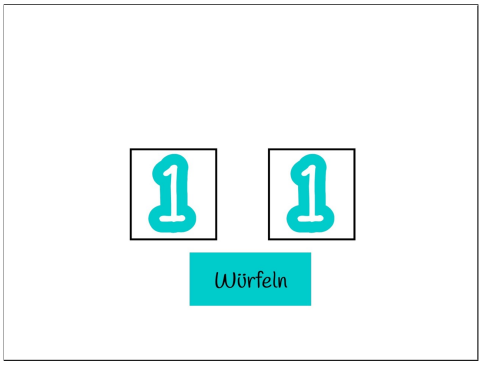
Aufgabe 8.4

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.

Die Lernenden können in dieser Aufgabe in der Visualisierung des "Erfolges" und des "Würfelns" kreativ sein und ihrem Projekt eine eigene Note verleihen. Als Herausforderung kann man auch weitere existierende Würfelspiele programmieren.

Lösungen

Die Lösungen zu den Aufgaben findest du auch online:
Kapitel 3: https://scratch.mit.edu/studios/26899341/
Kapitel 4: https://scratch.mit.edu/studios/27202421/
Kapitel 5: https://scratch.mit.edu/studios/27168352/
Kapitel 6: https://scratch.mit.edu/studios/27225638/
Kapitel 7: https://scratch.mit.edu/studios/27230131/
Kapitel 8: https://scratch.mit.edu/studios/27222532/

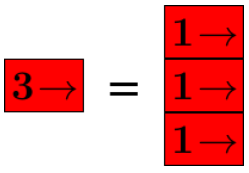
Lösungen zu Kapitel 1

Lösung zu Aufgabe 1.2:
Hier geht es darum, den Begriff des Programmierens zu repetieren und mit anderen zu diskutieren.

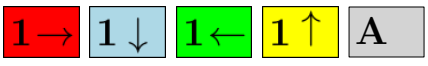
Lösung zu Aufgabe 1.3:

- Herd (Programmierung der Backzeit- und -temperatur)
- Waschmaschine (Programmierung der Waschdauer und -temperatur)
- Kaffeemaschine (Programmierung der Kaffeemenge und -stärke; Milch, Zucker?)
- Wecker (Programmierung von Weckzeiten, Weckton)
- ...

Lösung zu Aufgabe 1.4:
Die 2-er, 3-er und 4-er Befehle braucht man nicht unbedingt, weil man sie aus den 1-er Befehlen zusammenbauen kann, z.B.



Die 1-er Befehle braucht man, und zwar für jede Richtung (das Feuer könnte ja rechts, links, oberhalb, oder unterhalb der Startposition sein). Ebenso braucht man die Axt, falls der Roboter vom Feuer durch Wände getrennt ist. Man braucht also unbedingt diese 5 Befehle:



Lösung zu Aufgabe 1.5:
Mögliche Antworten (nur die dritte trifft den Kern):
Damit der Löschroboter schneller am Ziel ist?

- Das könnte sein, aber nicht, wenn wir annehmen, dass der Roboter für einen 3-er Befehl genauso lange braucht wie für drei 1-er Befehle.

Damit das Löschroboterspiel interessanter ist?

- Nein; beim Programmieren eines echten Löschroboters geht es nur darum, ihn so einfach und schnell wie möglich zum Brand zu bringen

Damit die Programme kürzer und damit einfacher zu erstellen und einfacher zu lesen sind?

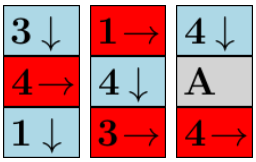
- Genau. Hätten wir nur die 1-er Befehle, wären die Programme lang und mühsam zu erstellen.

Lösung zu Aufgabe 1.6:
Korrekt: b), c), d)

Roboter kommt nicht zum Brand: e), f)

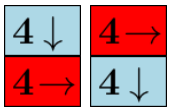
Roboter geht kaputt: a), f)

Lösung zu Aufgabe 1.7:
Es gibt 3 Möglichkeiten:

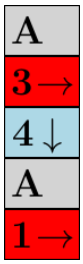


Lösung zu Aufgabe 1.8:

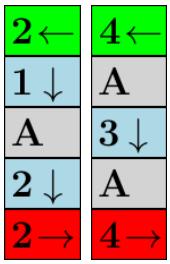
a) (2 kürzeste Programme)



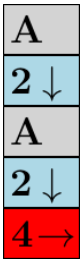
b)



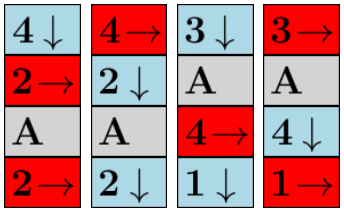
c) (2 kürzeste Programme)



d)

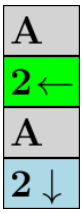


e) (4 kürzeste Programme)

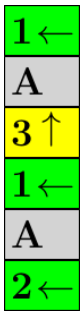


f)

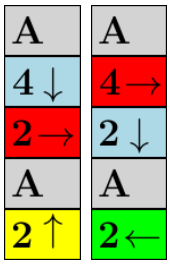
g)



h)



i) (2 kürzeste Programme)



Lösungen zu Kapitel 2

Lösung zu Aufgabe 2.1:
Die folgende Aufzählung von Vor- und Nachteilen ist nicht zwingend vollständig. Wenn dir oder deinen Klassenmitgliedern noch mehr einfällt, dann schreibt es dazu!
Wenn man das Projekt auf dem Computer speichert, ist es auch verfügbar, wenn man keinen Zugang zum Internet hat. Man kann es ausserdem auch auf einen anderen Account hochladen. Durch eine Ordnerstruktur behält man den Überblick über alle Projekte.
Wenn man das Projekt hingegen auf der Scratch-Webseite speichert, dann ist es auch von einem anderen Computer aus verfügbar, solange man die Login-Daten hat. Das Projekt wird ausserdem automatisch regelmässig gespeichert, wenn man die Scratch-Webseite benutzt. Auf der Seite "Meine Sachen" hat man einen Überblick über alle Projekte und kann sie alphabetisch oder nach Beliebtheit sortieren. Wenn man seine Projekte veröffentlicht hat, kann man sie auch in Studios sortieren.
Man kann das Projekt natürlich auch auf einen USB-Stick laden oder auf einer externen Festplatte speichern. Dadurch erhält man einige Vorteile von den beiden vorherigen Möglichkeiten: Das Projekt ist verfügbar, wenn man kein Internet hat und auch von einem anderen Computer aus.
Grundsätzlich gilt: Wenn man etwas auf vielen verschiedenen Speichermedien speichert, ist die Gefahr kleiner, dass man es verliert (weil zum Beispiel der Computer kaputt gegangen ist, gestohlen worden ist oder man die Datei aus Versehen gelöscht hat). Man muss dabei aber aufpassen, dass man auch überall die gleiche Version gespeichert hat. Ausserdem sollte man private Daten nicht über das Internet speichern, da sie sonst leichter gehackt werden können.

Lösungen zu Kapitel 3

Lösung zu Aufgabe 3.1:
Du trägst eine negative Schrittzahl ein, dann geht Scratch rückwärts: 

Lösung zu Aufgabe 3.2:
Die Bühne misst ganz genau 480 Schritte von links nach rechts (von x = -240 bis x = 240).

Lösung zu Aufgabe 3.3:
Zur Erinnerung:
0 Grad = oben
90 Grad = rechts
180 Grad = unten
-90 Grad = links
Gehe *Zahl* Schritte = Scratch bewegt sich um *Zahl* Schritte (Pixel, Bildpunkte) in die gezeigte Richtung
Gehe -*Zahl* Schritte = Scratch bewegt sich um *Zahl* Schritte (Pixel, Bildpunkte) in die entgegengesetzte Richtung (rückwärts), schaut aber immer noch in die gleiche Richtung

1. (a) 100 Schritte

   (b) 100 Schritte

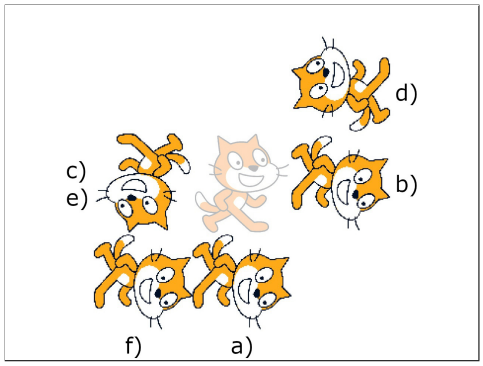
   (c) 100 + 200 = 300 Schritte

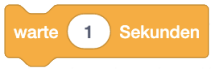
   (d) 100 + 100 = 200 Schritte

   (e) 100 Schritte (rückwärts)

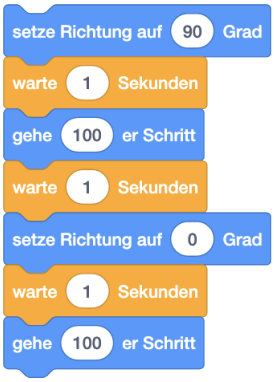
   (f) 100 + 100 + 200 + 200 = 600 Schritte

2.



Wenn du Scratch die Befehle langsam laufen lassen willst, kannst du den Befehl 

benutzen. Zum Beispiel für Teilaufgabe d):



Du findest diesen Block in der Blockpalette unter “Steuerung”. In Kapitel 5 werden wir diesen Block noch genauer anschauen.

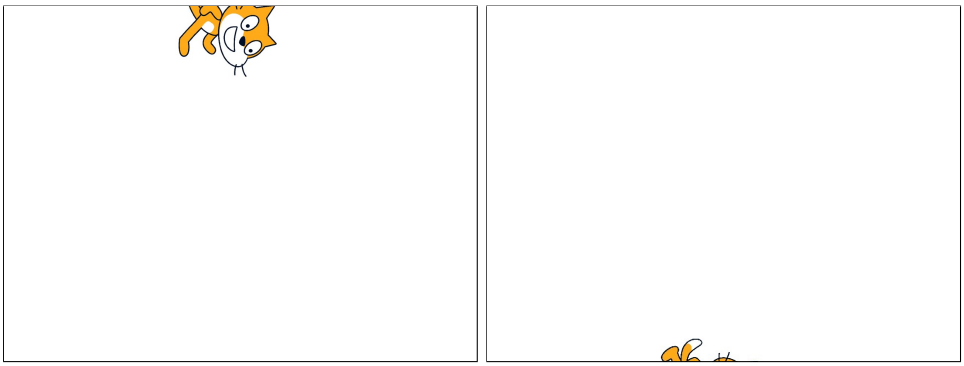
Lösung zu Aufgabe 3.4:

1. Es wurde vergessen, zu Beginn die Richtung zu setzen. Mit dem fehlerhaften Skript läuft Scratch nach rechts los statt nach oben.

2. Das sechste Wegstück wurde vergessen. Richtig wäre das folgende Skript:

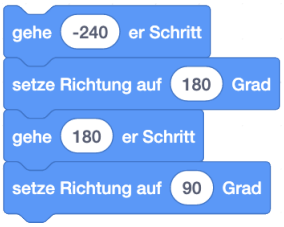
Lösung zu Aufgabe 3.5:
Die Bühne misst ganz genau 360 Schritte von oben nach unten (von y = -180 bis y = 180). Setze zum Überprüfen die Richtung auf 180 Grad und schiebe Scratch an den oberen Rand der Bühne, so dass ein gewähltes Merkmal genau auf dem Bühnenrand liegt. Zähle dann, wie oft du den 

Befehl benutzen musst, damit Scratch den unteren Rand der Bühne erreicht und das gewählte Merkmal genau auf dem Bühnenrand liegt.



Lösung zu Aufgabe 3.6:
Es gibt mehrere Möglichkeiten, diese Aufgabe zu lösen. Hier sind zwei:

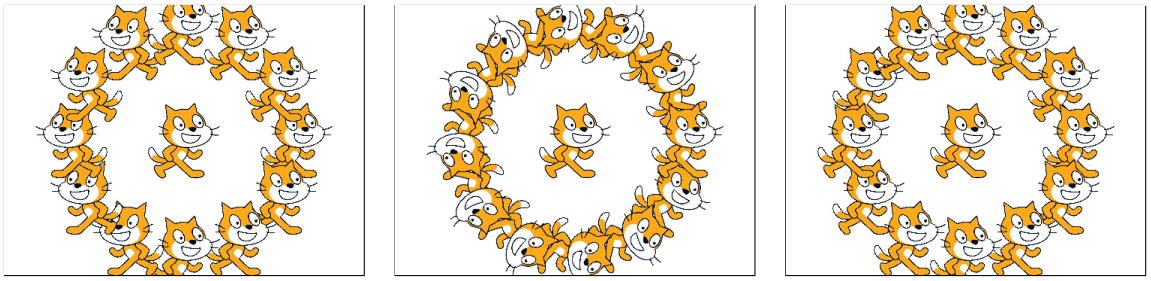
Variante 1: Scratch geht rückwärts zum linken Rand, dreht sich dann nach unten und geht vorwärts zum unteren Rand. Zum Schluss muss Scratch dann wieder nach rechts schauen!



Variante 2: Scratch dreht sich zuerst nach unten und geht vorwärts zum unteren Bühnenrand. Dann dreht sich Scratch wieder zurück nach rechts und geht rückwärts in die untere linke Ecke.

Lösung zu Aufgabe 3.7:

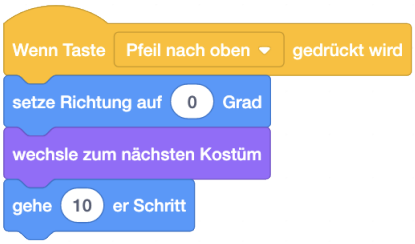
Drehtyp "links-rechts" Drehtyp "rundherum" Drehtyp "nicht drehen"



Mit dem Drehtyp "links-rechts" schaut Scratch nach links, wenn die Richtung auf der linken Hälfte der Auswahlscheibe liegt und somit ein negatives Vorzeichen hat. Sonst schaut Scratch nach rechts, insbesondere auch für die Richtungen 0 Grad (oben) und 180 Grad (unten).
Mit dem Drehtyp "rundherum" schaut Scratch jeweils in die ausgewählte Richtung.
Mit dem Drehtyp "nicht drehen" schaut Scratch immer nach rechts.

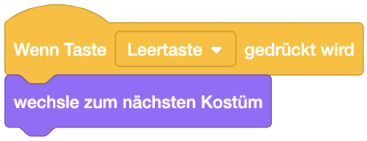
Lösung zu Aufgabe 3.8:
Der Computer ist sehr schnell und führt die vier Befehle deshalb in so kurzer Zeit aus, dass du den Kostümwechsel gar nicht sehen kannst.

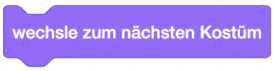
Lösung zu Aufgabe 3.9:
Hochlaufen:



Runterlaufen:

Lösung zu Aufgabe 3.10:
Hier können natürlich auch beliebige andere Tasten benutzt werden.

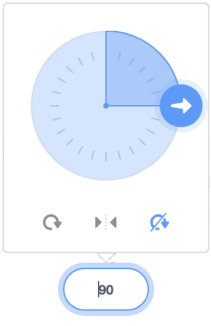


Dieses Skript funktioniert, weil der Computer zu jeder Ausführung des 

Befehls ein neues Skript starten muss. Dadurch bleibt etwas Zeit zwischen den einzelnen Kostümwechseln, und die Animation wird sichtbar.
Achtung: Das Skript unten funktioniert nicht, da hier auch wieder alle Befehle zu schnell nacheinander ausgeführt werden.

Lösung zu Aufgabe 3.11:
Für jede Erweiterung gibt es verschiedene Lösungen. Ein paar Möglichkeiten sind hier vorgestellt. Du musst darauf achten, dass du nicht dieselbe Taste für verschiedene Funktionen benutzt (zum Beispiel die Taste 'r' für einen Farbwechsel zu rot und für den Radiergummi).

1. Man kann den Drehtyp entweder mit einem Skript setzen oder indem man in der Objektliste auf Richtung und das "nicht drehen"-Symbol klickt.



2.

3.


Wenn Taste 5 gedrückt wird
setze Stiftdicke auf 5
Wenn Taste 1 gedrückt wird
setze Stiftdicke auf 1

4.

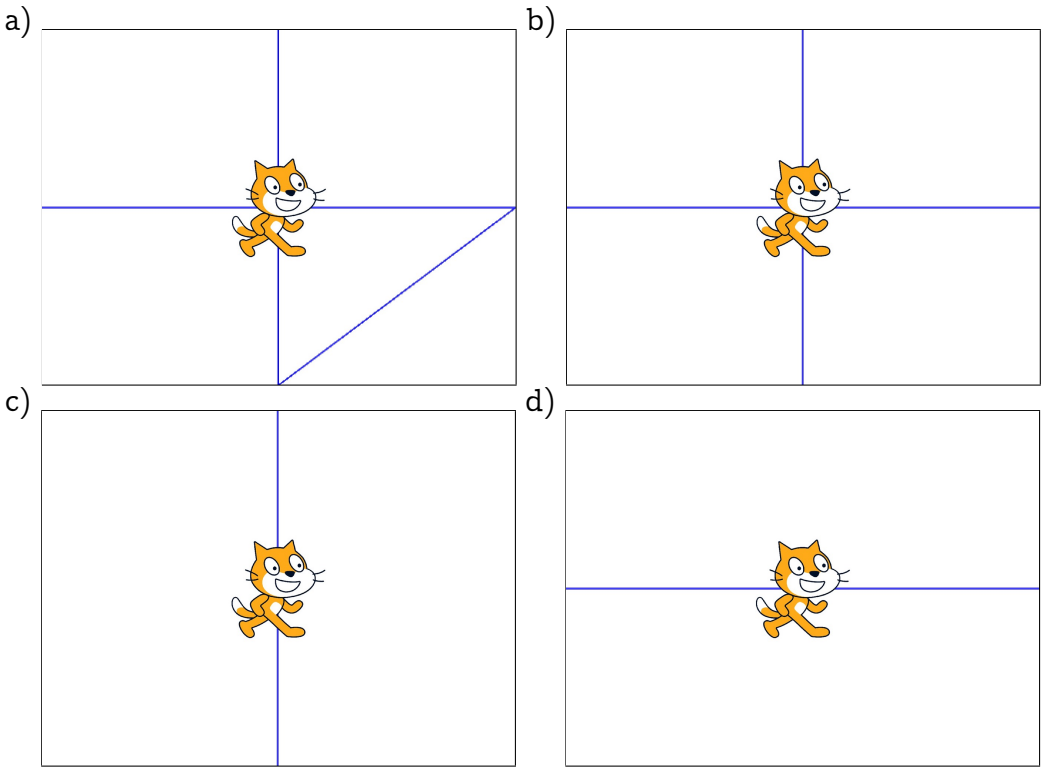

Wenn Taste e gedrückt wird
schalte Stift ein
wechsle zu Kostüm Stift Ein
Wenn Taste a gedrückt wird
schalte Stift aus
wechsle zu Kostüm Stift Aus

5.


Wenn Taste r gedrückt wird
wechsle zu Kostüm Radiergummi
schalte Stift ein
setze Stiftfarbe auf
setze Stiftdicke auf 30

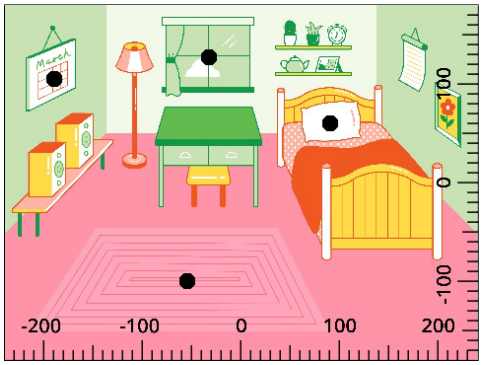
Lösungen zu Kapitel 4

Lösung zu Aufgabe 4.1:



Lösung zu Aufgabe 4.2:

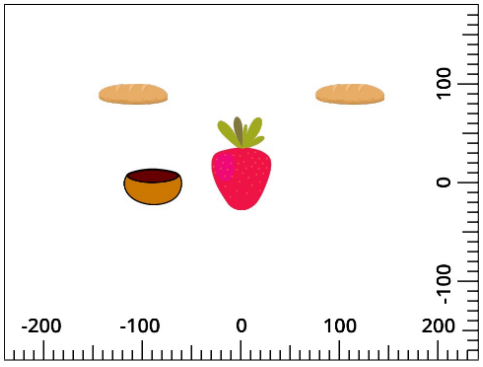
Die geschätzten Werte sollten auf ± 20 stimmen, die gemessenen auf ± 10.



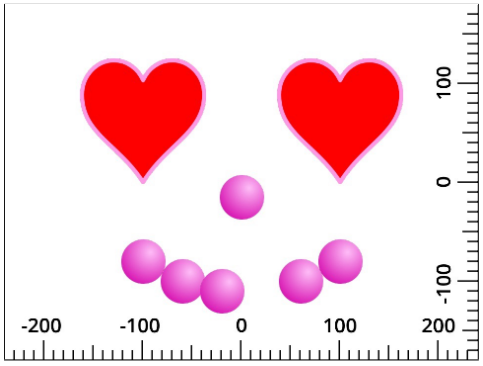
	Kalender:
	(,)
	(-190 , 105)
	Fenster:
	(,)
	(-33 , 127)
	Kissen:
	(,)
	(90 , 60)
	Teppich:
	(,)
	(-55 , -100)

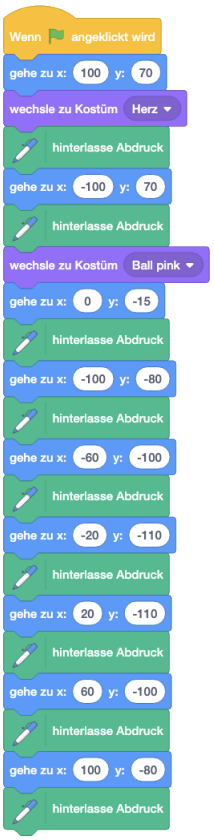
Lösung zu Aufgabe 4.3:

1. Beim letzten Gehe-Zu Befehl wurden die Parameter für x und y verwechselt. Mit dem fehlerhaften Skript wird das Bild unten erzeugt. Rechts ist das korrigierte Skript.



2. Es fehlt eines der Stempelbilder für den Mund. Mit dem fehlerhaften Skript wird das Bild unten erzeugt. Rechts ist das korrigierte Skript.



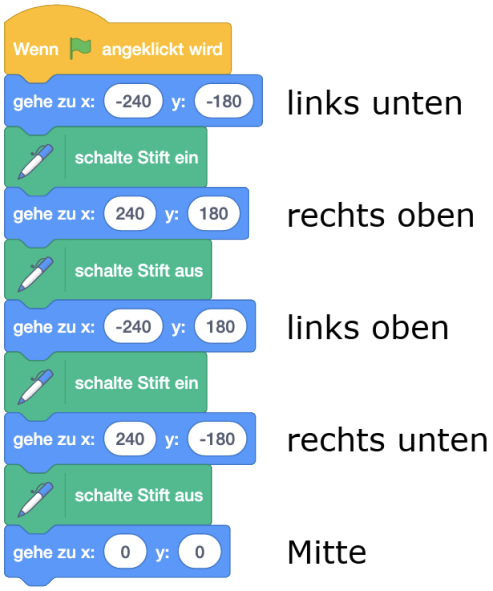


Lösung zu Aufgabe 4.4:

Die genaue Höhe und Länge der Wäscheleine ist nicht so wichtig. Wichtig ist, dass sowohl das linke als auch das rechte Ende die gleiche y-Koordinate (hier 42) haben, denn nur dann haben beide Enden die gleiche Höhe und die Leine ist gerade.

Lösung zu Aufgabe 4.5:

Es gibt viele Möglichkeiten, das Kreuz zu malen. Wichtig ist, dass es keine Linien entlang der Bühnenränder gibt. Selbst wenn man diese nicht oder kaum sieht, ist die Lösung dann nicht korrekt. Der letzte Schalte-Stift-Aus Befehl kann hier auch weggelassen werden, da Scratch diese diagonale Linie bereits gezeichnet hat.



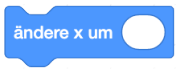
Lösung zu Aufgabe 4.6:

Hier muss darauf geachtet werden, dass Scratch bei den Bewegungen von und zur Mitte keine Malspur hinterlässt (erster und letzter Gehe-Zu Befehl). Auch hier gibt es viele andere Möglichkeiten, dieses Bild zu malen.

Lösung zu Aufgabe 4.7:

1. Das Gehen muss *nach* dem Anheben des Stiftes *oder vor* dem Wegwischen der Malspuren erfolgen, da sonst beim Gehen zur Mitte noch eine Malspur entsteht, die nicht weggewischt wird. Die Reihenfolge ist also nicht egal, und das Skript rechts wäre falsch.

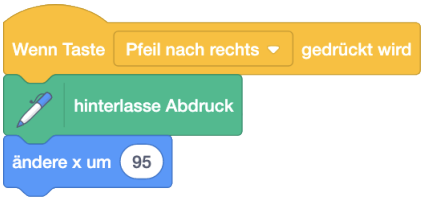
2. Das Kostüm wird nicht zurückgesetzt. Falls Scratch aktuell das "Gehen"-Kostüm (rechts) trägt, ist das Aufräumen unvollständig. Das Aufräumskript sollte wie rechts gezeigt ergänzt werden.

Lösung zu Aufgabe 4.8:
Der 

Befehl bewegt das Objekt unabhängig von der Richtung des Objekts um den eingegebenen Parameter nach rechts (oder nach links, wenn der Parameter negativ ist). Der 

Befehl hingegen bewegt das Objekt um den eingegebenen Parameter in die Richtung des Objekts (oder entgegengesetzt wenn der Parameter negativ ist). Nur wenn die Richtung 90 Grad ist, bewegt sich das Objekt mit beiden Befehlen in die gleiche Richtung.

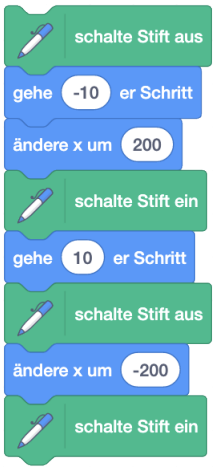
Lösung zu Aufgabe 4.9:
Die konkreten Änderungswerte können auch etwas anders aussehen und sollten durch Ausprobieren gefunden werden.



Lösung zu Aufgabe 4.10:

Wir machen uns den folgenden Plan:

1. Mache die letzte Bewegung rückgängig
2. Bewege dich ohne zu zeichnen auf die rechte Seite
3. Zeichne die letzte Bewegung der linken Seite auf der rechten Seite
4. Bewege dich ohne zu zeichnen zurück auf die rechte Seite, an den Punkt, wo wir aufgehört haben zu zeichnen
5. Schalte den Stift wieder ein



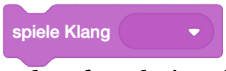
Lösung zu Aufgabe 4.11:

1. Der Befehl 

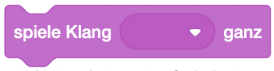
erzeugt eine dauerhafte Sprechblase und ist damit ausgeführt. Der Befehl kostet also fast keine Zeit. Die Sprechblase kann durch andere Sprech- oder Denkblasen ersetzt werden und mit dem Befehl ohne Parameter kann man sie wieder verschwinden lassen.

   Der Befehl 

erzeugt eine Sprechblase, wartet die angegebene Zeit und lässt die Sprechblase dann wieder verschwinden. Erst dann ist der Befehl ausgeführt. Der Befehl kostet also ungefähr die im zweiten Feld angegebene Zeit.

2. Der Befehl 

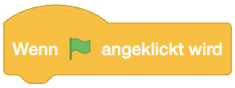
startet einen Klang und ist damit ausgeführt. Der Befehl kostet also fast keine Zeit. Der Klang läuft aber weiter. Wenn der gleiche Klang währenddessen nochmals gestartet wird, wird der erste Klang abgebrochen und nur der zweite Klang läuft weiter. Wenn ein anderer Klang währenddessen gestartet wird, erklingen beide Klänge gleichzeitig.

Der Befehl 

startet einen Klang und wartet, bis er abgespielt ist. Erst dann ist der Befehl ausgeführt. Der Befehl kostet also ungefähr die Zeit, die der Klang dauert.

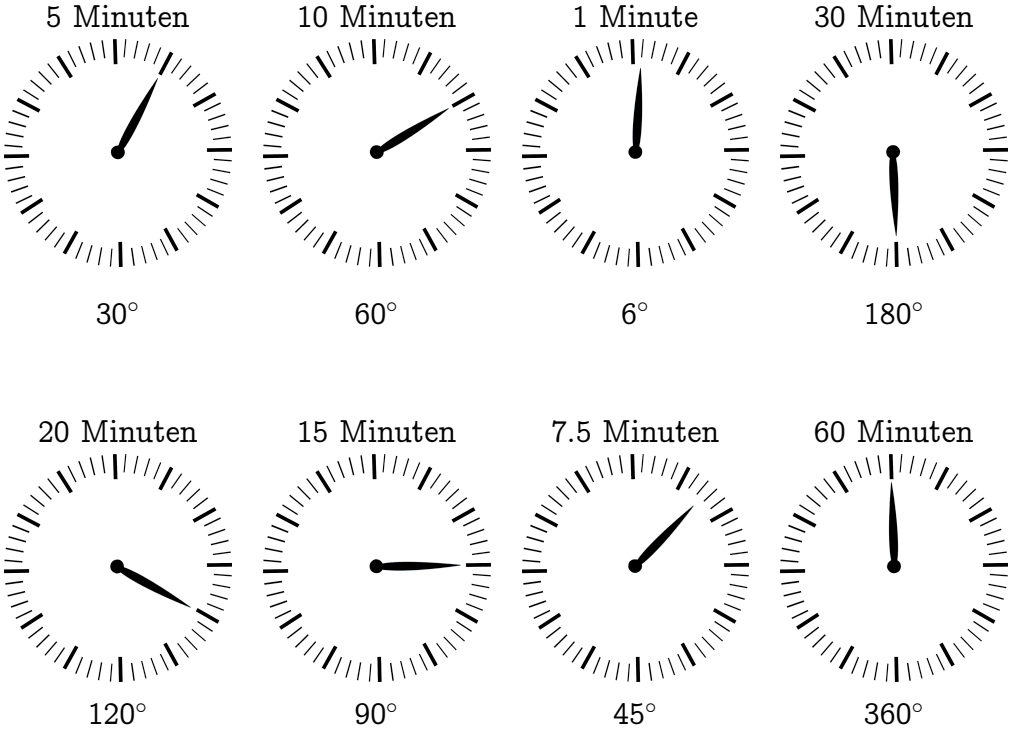
Lösung zu Aufgabe 4.12:
Du kannst bei dieser Aufgabe völlig frei programmieren. Achte einfach darauf, dass deine Figuren dann sprechen, wann du es geplant hast.

Lösung zu Aufgabe 4.13:

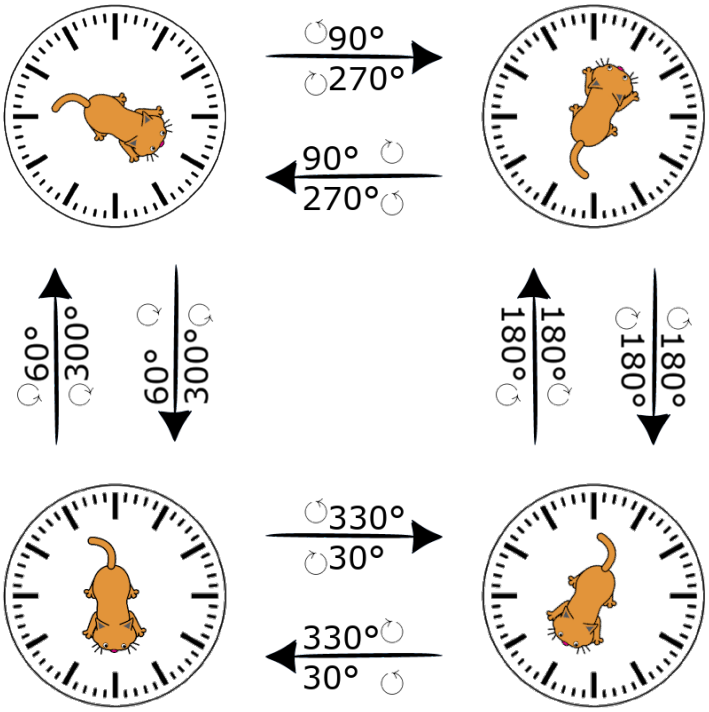
Falls du die Aufgabe 4.10 gelöst hast, entferne zuerst die Ergänzung aus dieser Lösung. Dupliziere dann das Objekt und ändere nur das Skript unter dem 

Hut zu diesem:

Lösung zu Aufgabe 4.14:

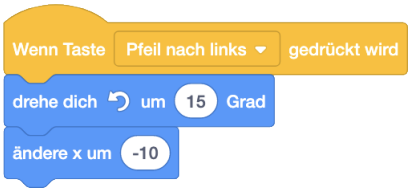


Lösung zu Aufgabe 4.15:



Drehwinkel im Uhrzeigersinn + Drehwinkel im Gegenuhrzeigersinn = 360°
Drehwinkel im Uhrzeigersinn von Bild A zu Bild B = Drehwinkel im Gegenuhrzeigersinn von Bild B zu Bild A

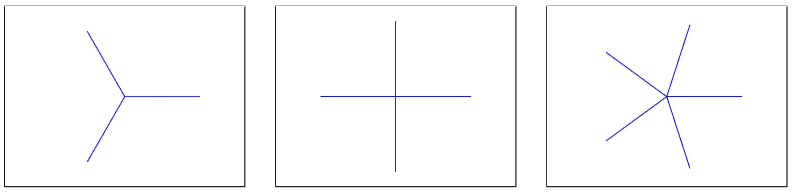
Lösung zu Aufgabe 4.16:
Die konkreten Werte können auch anders aussehen und sollten so gewählt werden, dass die Bewegung realistisch aussieht.



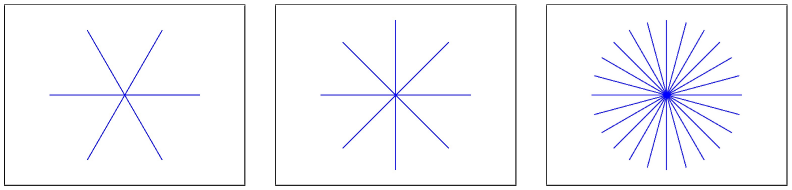
Lösung zu Aufgabe 4.17:
Der Wert im 

-Block (in der Lösung hier 100) beeinflusst nur die Grösse des Sterns, aber nicht seine Form.
Der Drehwinkel lässt sich folgendermassen berechnen:
Drehwinkel = 360° (volle Drehung) / Anzahl Zacken

Grundskript:



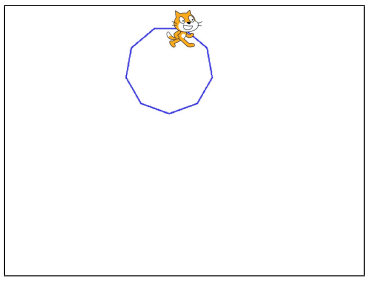
Drehwinkel: 120° 90° 72°



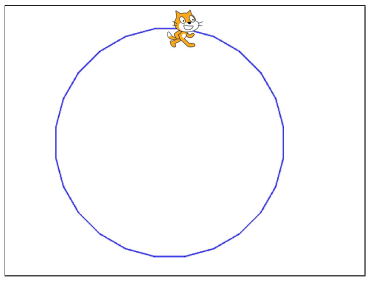
Drehwinkel: 60° 45° 15°

Im nächsten Kapitel lernen wir eine Methode kennen, mit der wir direkt den ganzen Stern zeichnen lassen können.

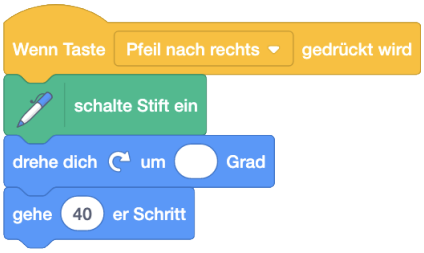
Lösung zu Aufgabe 4.18:



Drehwinkel 40°

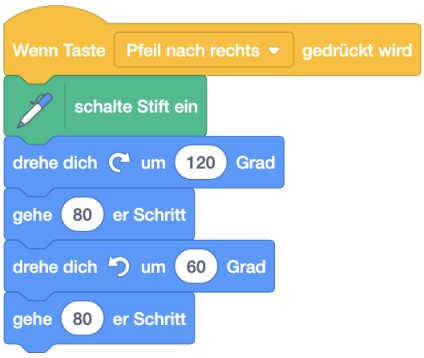


Drehwinkel 15°

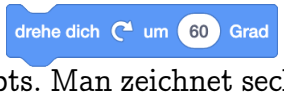


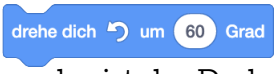
Lösung zu Aufgabe 4.19:
Bei dieser Aufgabe muss man beachten, dass die erste Drehung im Uhrzeigersinn, die zweite Drehung aber gegen den Uhrzeigersinn gemacht werden muss. Die Schrittzahl muss nicht gleich gewählt werden wie hier, aber sie muss für die beiden Linien gleich sein.

a)

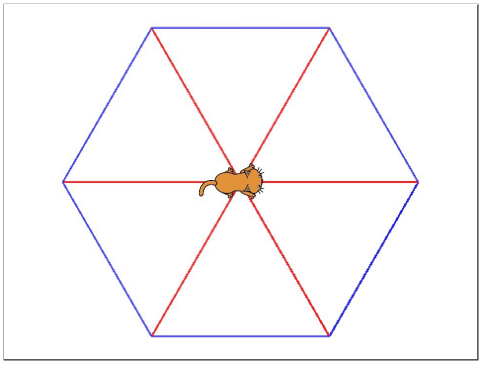


b)

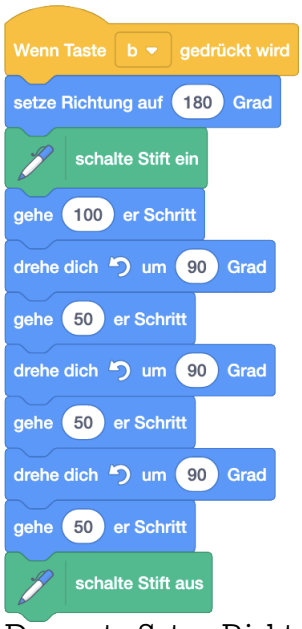
Lösung zu Aufgabe 4.20:
Ergänzung: Hinzufügen von 

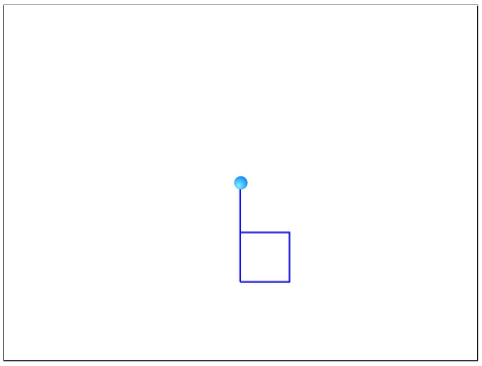
oder 

am Anfang oder am Ende des Skripts. Man zeichnet sechs Dreiecke, also ist der Drehwinkel 360° / 6 = 60°. Im Bild erkennt man auch der 6er-Stern aus Aufgabe 4.17 wieder:



Lösung zu Aufgabe 4.21:
Beispielslösung für ein kleines b:





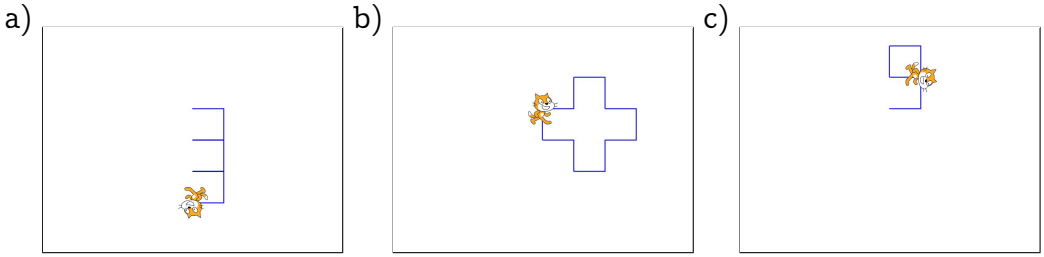
Der erste Setze-Richtung-Befehl ist notwendig, da wir nicht wissen, in welche Richtung Scratch vor dem Zeichnen des Buchstabens schaut.
Du kannst statt den “Drehe”-Blocks auch “Zeige Richtung”-Blocks verwenden.

Lösungen zu Kapitel 5

Lösung zu Aufgabe 5.1:
Weitere Beispiele könnten sein: Krafttraining (Wiederholungen von Sets), Küche aufräumen (jedes dreckige Geschirrstück wäscht man zuerst ab, trocknet es dann ab und räumt es dann in den Schrank ein) oder der Tanz Macarena.
Was hier eher nicht passt sind Wiederholte Ausstrahlung von Fernsehserien, das Fussballtraining jede Woche oder das tägliche Zähneputzen, da diese Beispiele eher regelmässig wiederkehrende Ereignisse beschreiben und nicht eine direkt aufeinanderfolgende Wiederholung von Ereignissen oder Tätigkeiten.

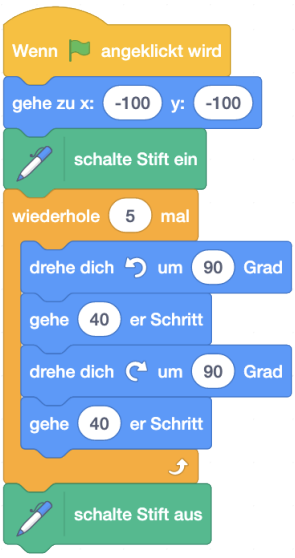
Lösung zu Aufgabe 5.2:
Scratch sagt 'a b c b c b c d'.

Lösung zu Aufgabe 5.3:
Scratch zeichnet die folgenden Muster:



Lösung zu Aufgabe 5.4:
Es gibt verschiedene Möglichkeiten, um diese Bilder zu zeichnen. So kann man zum Beispiel für die Acht zuerst den oberen und dann den unteren Kreis zeichnen oder umgekehrt. Die Treppe kann man von oben beginnen oder von unten, und vielleicht geht das Objekt sogar einen Teil des Weges rückwärts. Bei der Acht musst du dir aber genau überlegen, wo der erste Kreis aufhört und wie du von dort zum Anfang des zweiten Kreises kommst. Wenn du also zuerst den oberen Kreis zeichnen willst, und dabei von oben beginnst, so musst du nochmals einen Halbkreis zeichnen, um zum Anfang des unteren Kreises zu kommen. Dasselbe gilt, wenn du die liegende Acht von ganz links oder ganz rechts aus zeichnen willst. Deine Lösung kann also ganz anders aussehen als die Lösungen unten, das Ergebnis sollte aber gleich sein!

Treppe

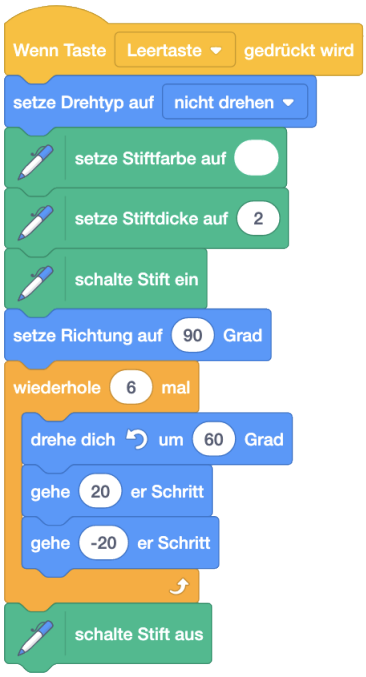


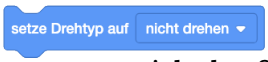
Acht

Liegende Acht

Lösung zu Aufgabe 5.5:

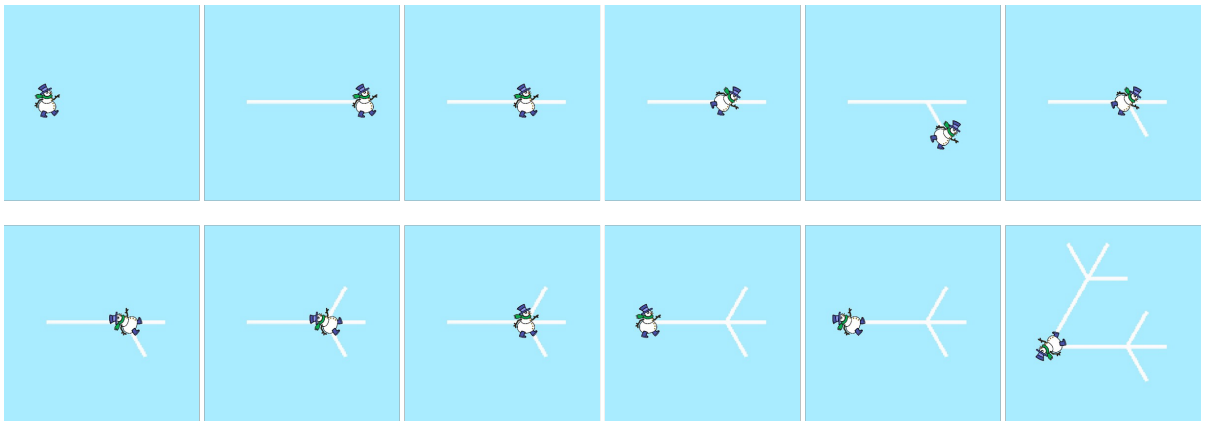
Einfache Schneeflocke



Der Befehl 

ist nicht zwingend nötig, es sieht aber etwas eindrucksvoller aus, wenn sich der Schneemann beim Zeichnen nicht zu bewegen scheint. Wenn du mit der schwierigeren Schneeflocke Mühe hast, probiere zuerst, dem Schneemann die passenden Befehle zu geben, damit er einen "Arm" der Schneeflocke zeichnet.

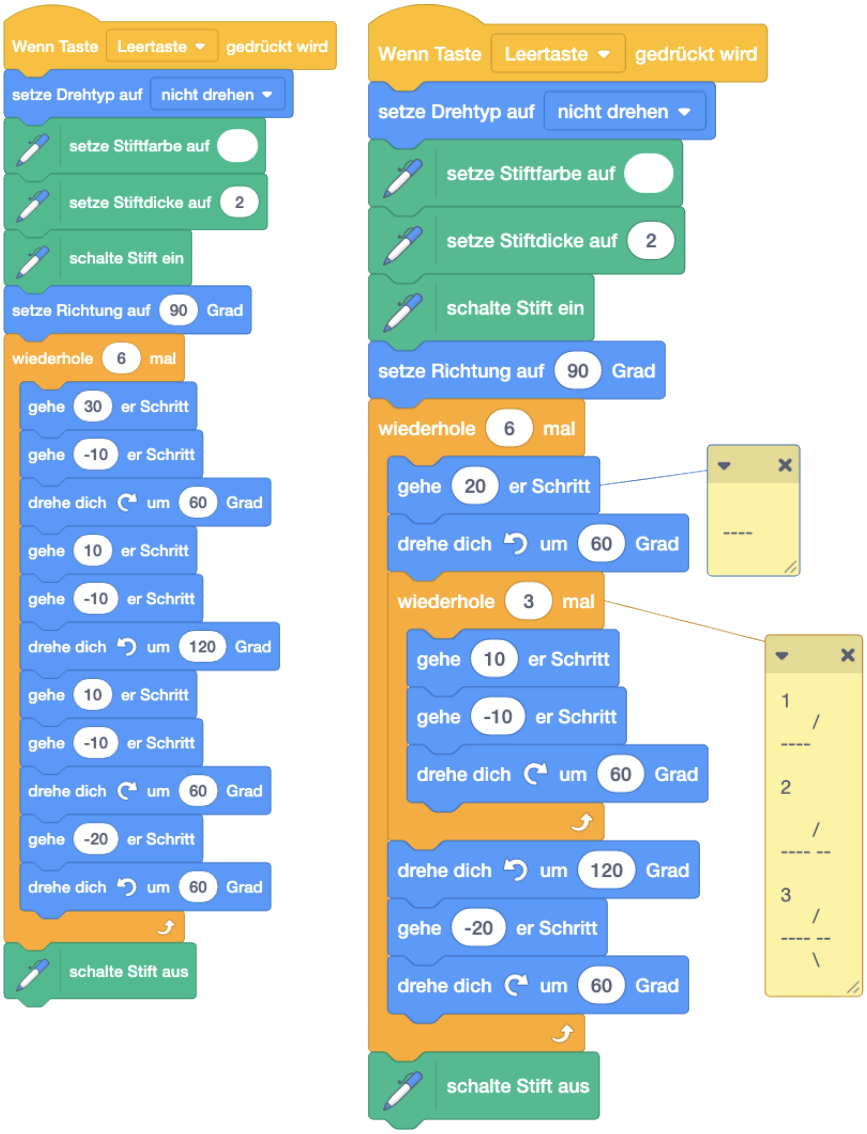
Hier ist ein möglicher Plan, um einen "Arm" der Schneeflocke zu zeichnen. Der Schneemann auf den Bildern ist verkleinert, damit man besser sehen kann, was er gerade zeichnet.



Wieviele "Arme" muss der Schneemann zeichnen und welchen Befehl braucht man für den Übergang zwischen zwei "Armen"?

Schwierigere Schneeflocke

Version 1 Version 2

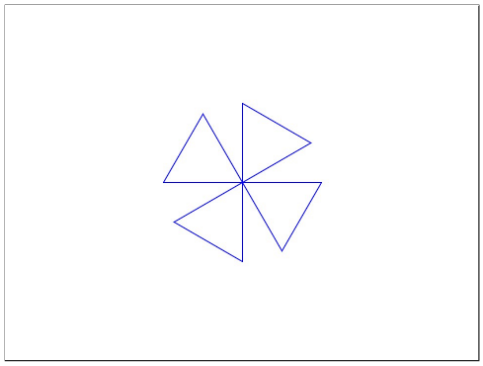


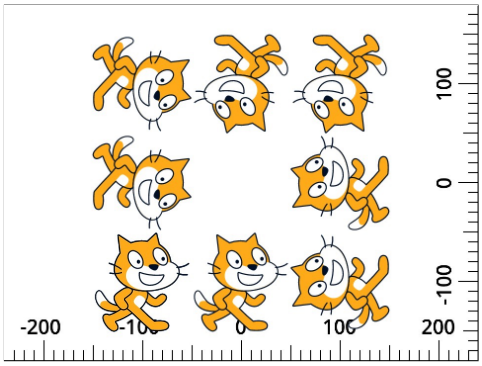
Version 2 verwendet verschachtelte Schleifen. Auf dieses Konzept werden wir im folgenden Kapitel 5.2 noch genauer eingehen.

Lösung zu Aufgabe 5.6:

1. Richtungswechsel kannst du in diesem Teil der Aufgabe ignorieren, da Scratch dafür keine Schritte läuft. Schritte nacheinander werden addiert, während alle Schritte innerhalb eines *Wiederhole-Mehrmals*-Blocks mit der zu wiederholenden Zahl multipliziert werden.
   a) $4 \cdot 3 \cdot 80 = 960$ Schritte
   b) $100 + 100 + 4 \cdot 2 \cdot 100 = 1000$ Schritte

2. a) b)

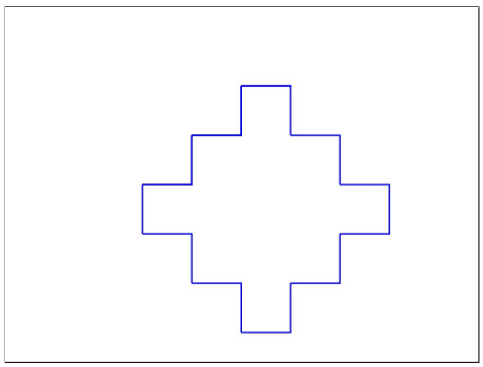


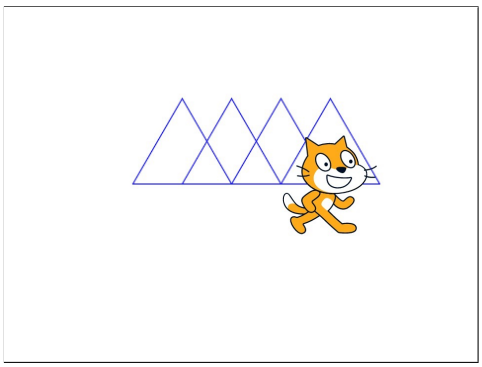


Lösung zu Aufgabe 5.7:

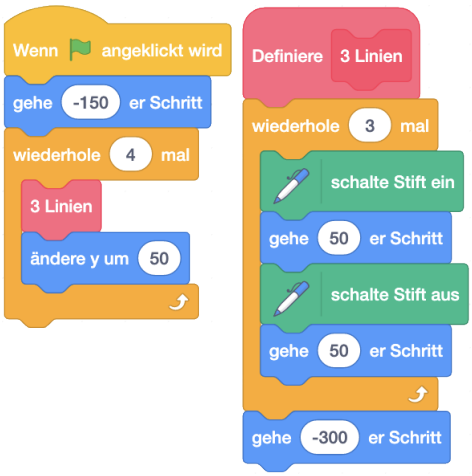
1. a) Pro Treppe läuft Scratch $2 \cdot (50 + 50) = 200$ Schritte. Somit läuft Scratch insgesamt 100 (rückwärts) $+ 4 \cdot$ (200 (Treppe) $+ 50) = 1100$ Schritte
   b) Pro Dreieck läuft Scratch $3 \cdot 100 = 300$ Schritte. Somit läuft Scratch insgesamt 120 (rückwärts) $+ 4 \cdot$ (300 (Dreieck) $+ 50) = 1520$ Schritte

2. a) b)

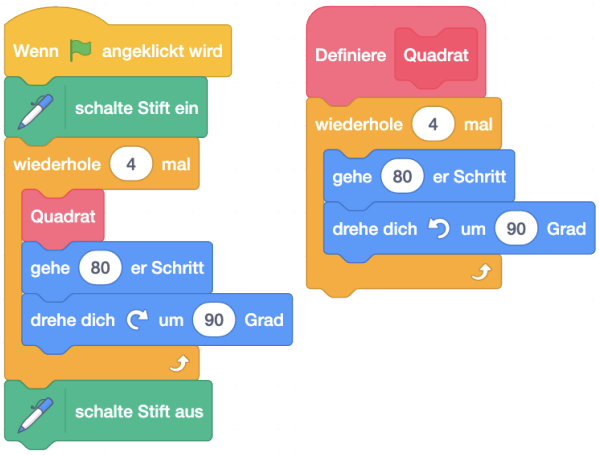




Lösung zu Aufgabe 5.8:
a) Wir erkennen, dass die innere Schleife 3 Linien hintereinander zeichnet. Mit dem Befehl "gehe -300er Schritt" geht man dann wieder zu dem Punkt zurück, wo man war bevor die Linien gezeichnet wurden. Wir definieren daher den "3 Linien" Block und schreiben das Programm folgendermassen um:



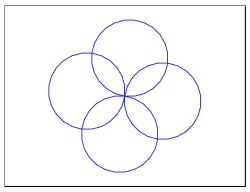
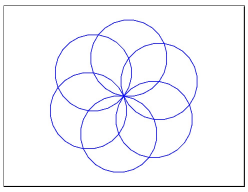
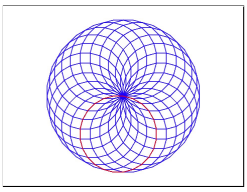
b) Mit der inneren Schleife zeichnet Scratch ein Quadrat.



Lösung zu Aufgabe 5.9:

Grundskript: Kreis-Block Definition:

a) b) c)

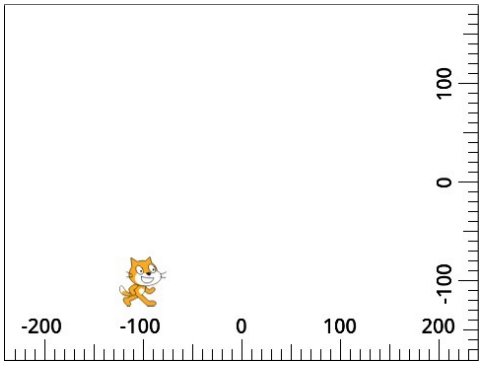




Drehwinkel:	90°	60°	15°
Wiederholungen:	4	6	24

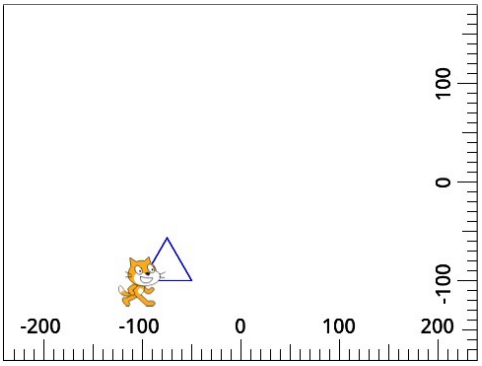
Wiederholungen = Anzahl Kreise
Drehwinkel = 360° / Wiederholungen

Lösung zu Aufgabe 5.10:
Nach dem Zeichnen eines einzelnen Dreiecks ist man wieder an der gleichen Stelle wie vor dem Zeichnen.

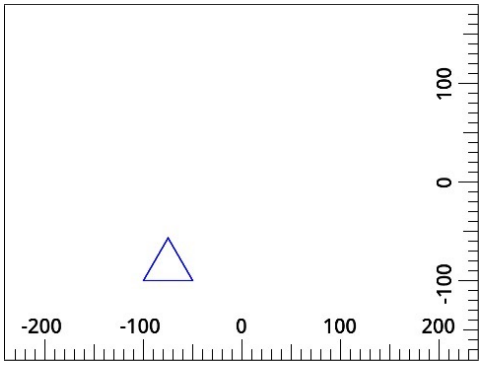
Vor dem Zeichnen:



Nach dem Zeichnen:

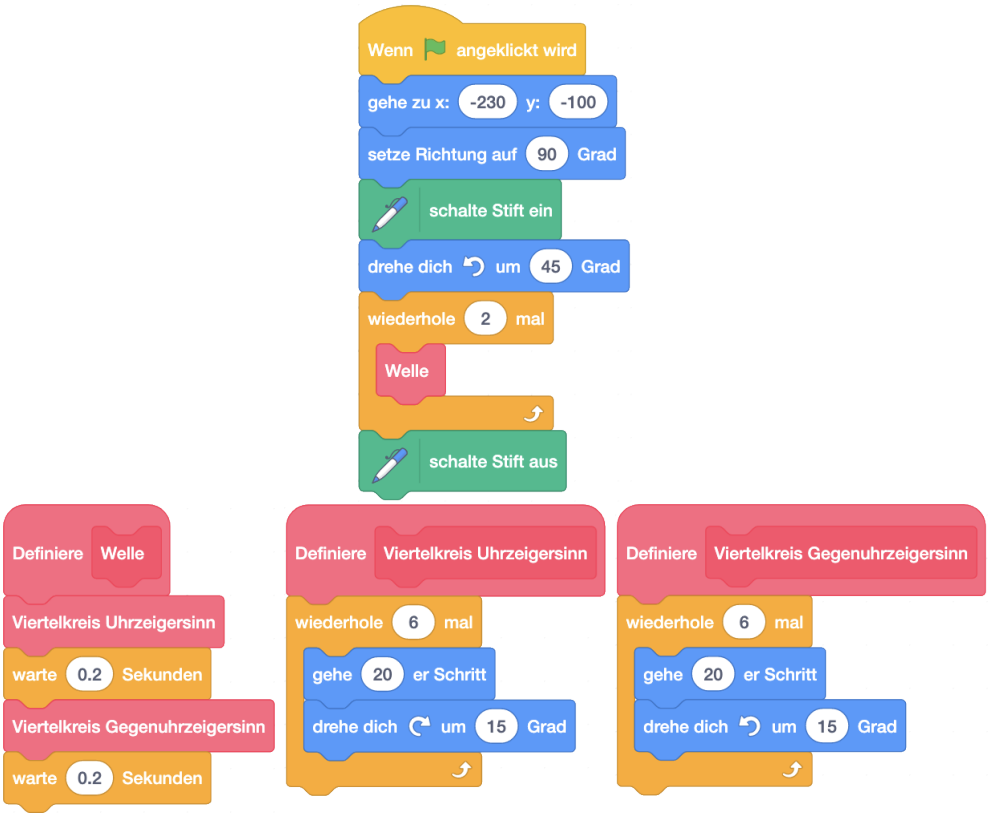


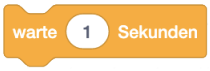
Daher zeichnet Scratch mit dem fehlerhaften Skript alle Dreiecke an der gleichen Stelle, wie im Bild unten. Im Programm wurde vergessen, Scratch zwischen den Dreiecken um die Seitenlänge des Dreiecks vorwärts zu bewegen. An der Definition des Blocks muss man nichts verändern. Das korrigierte Skript ist rechts:



Lösung zu Aufgabe 5.11:
Bei dieser Aufgabe kannst du deiner Kreativität freien Lauf lassen! Hier ist ein Beispiel für einen einfachen Tanz:

Lösung zu Aufgabe 5.12:



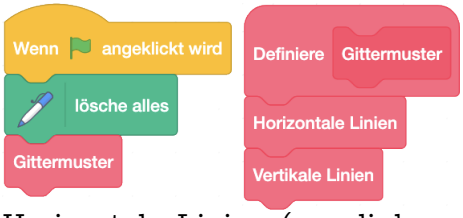
Der Startpunkt der Welle kann natürlich auch anders gewählt werden. Ebenso kann die Geschwindigkeit des Zeichnens mit der Zahl im 

Block variiert werden.

Lösung zu Aufgabe 5.13:
Bei dieser Aufgabe muss man sich gut überlegen, wieviele Wiederholungen nötig sind.

Ausserdem muss man sich entscheiden, ob man jeweils am Anfang oder am Ende der Schleife (bzw. vor oder nach dem Zeichnen der Linie) die Position wechselt.

Ablauf:
Wir zeichnen zuerst die horizontalen Linien (von links nach rechts) und dann die vertikalen Linien (von oben nach unten).



Horizontale Linien (von links nach rechts): Gehe ohne zu zeichnen in die linke untere Ecke und wiederhole dann mehrmals diese Schritte:

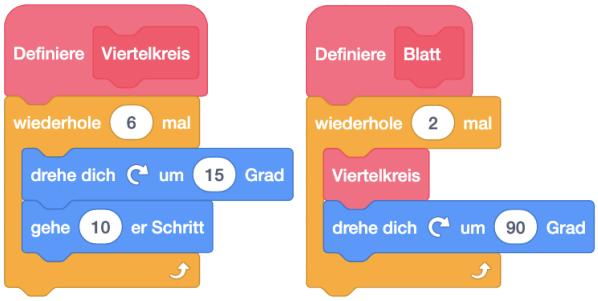
1. Bewege dich um 60 Schritte nach oben ohne zu zeichnen
2. Zeichne eine gerade Linie bis zum rechten Ende der Bühne
3. Gehe wieder zurück an den linken Rand der Bühne (egal ob mit oder ohne Zeichnen, da wir diese Linie bereits im vorherigen Schritt gezeichnet haben)

Vertikale Linien (von unten nach oben): Gehe ohne zu zeichnen in die linke untere Ecke und wiederhole dann mehrmals diese Schritte:

1. Bewege dich um 60 Schritte nach rechts ohne zu zeichnen
2. Zeichne eine gerade Linie bis zum oberen Ende der Bühne
3. Gehe wieder zurück an den unteren Rand der Bühne (egal ob mit oder ohne Zeichnen, da wir diese Linie bereits im vorherigen Schritt gezeichnet haben)

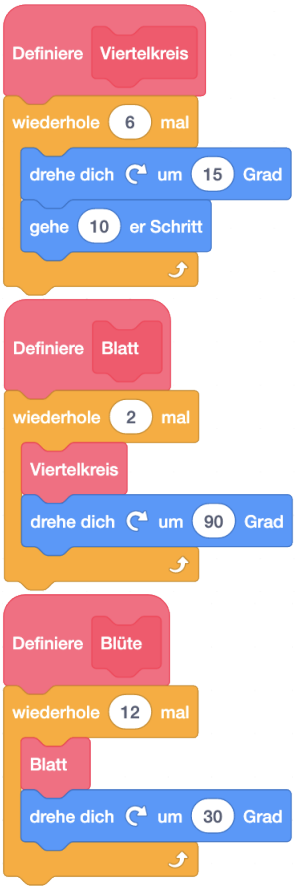
Lösung zu Aufgabe 5.14:
Wir erstellen zuerst den eigenen Block "Viertelkreis". Diesen benutzen wir dann in-

nerhalb des eigenen Blocks “Blatt”, mit dem Scratch zweimal zuerst einen Viertelkreis läuft und sich dann um 90 Grad dreht.



Um nur die Blüte der Blume zu zeichnen, können wir dann dieses Skript verwenden:

Wenn wir aber die ganze Blume zeichnen wollen, können wir dieses Skript wiederum als eigenen Block “Blüte” definieren. Das Programm für die ganze Blume sieht dann wie folgt aus:



Für einzelne Befehle (wie hier zum Beispiel das Zeichnen des Stiels) lohnt es sich nicht, einen eigenen Block zu erstellen. Es kann hier aber hilfreich sein, einen Kommentar hinzuzufügen, damit man weiss, wozu dieser Befehl ausgeführt wird.

Für das Skript der ganzen Blume hätten wir ohne die eigenen Blöcke dreifach verschachtelte Schleifen benutzen müssen, was unübersichtlich wird (Skript rechts). Mit den eigenen Blöcken lässt sich zudem gut überprüfen, ob die einzelnen Schritte des Ablaufs richtig programmiert sind. So kann man bei einem Fehler zum Beispiel zuerst überprüfen, ob der Block für den Viertelkreis so stimmt. Dann testet man, ob der Block für ein einzelnes Blatt korrekt ist, und so weiter.

Skript ohne Blöcke mit dreifach verschachtelten Schleifen:

Lösung zu Aufgabe 5.15:

Ablauf:
Wiederhole 12-mal: Male einen Fünf-Minuten-Abschnitt.
Ablauf für den Fünf-Minuten-Abschnitt:

1. Male einen langen Strich
2. Drehe dich
3. Wiederhole 4-mal: Male einen kurzen Strich und drehe dich.

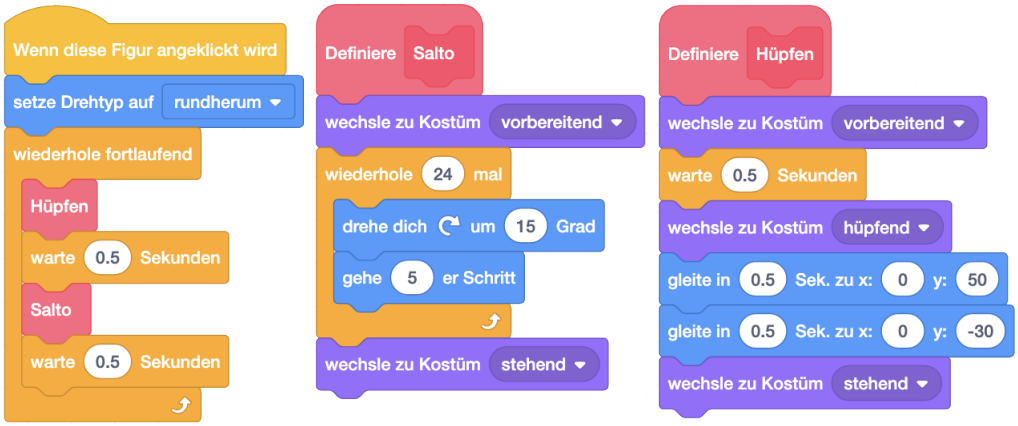
Ablauf für einen Strich:

1. Gehe 150 Schritte
2. Schalte den Stift ein
3. Gehe eine passende Anzahl Schritte zurück
4. Schalte den Stift aus
5. Gehe die restlichen Schritte zurück zur Mitte der Uhr

Es gibt viele andere Abläufe, die auch funktionieren. Der Fünf-Minuten-Abschnitt lässt sich zum Beispiel auch zeichnen, indem man sich zuerst dreht, dann die vier kurzen Striche zeichnet, sich weiter dreht und dann den längeren Strich zeichnet. Oder man könnte auch zuerst 60 kleine Striche in 6-Grad-Schritten zeichnen, und danach 12 grosse Striche in 30-Grad-Schritten.

Lösung zu Aufgabe 5.16:

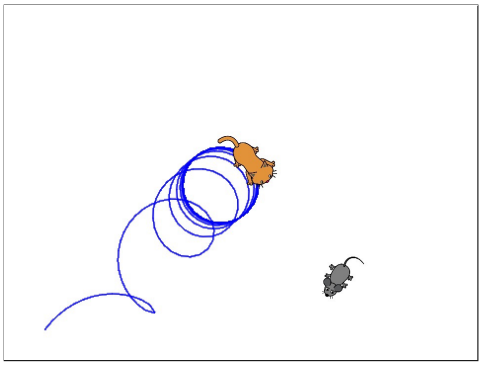
Wie in Aufgabe 5.11 kann deine Lösung hier wieder ganz anders aussehen. Hier ist wieder ein Beispiel für einen Tanz.



Lösung zu Aufgabe 5.17:

Skript für Scratch: Skript für die Maus:

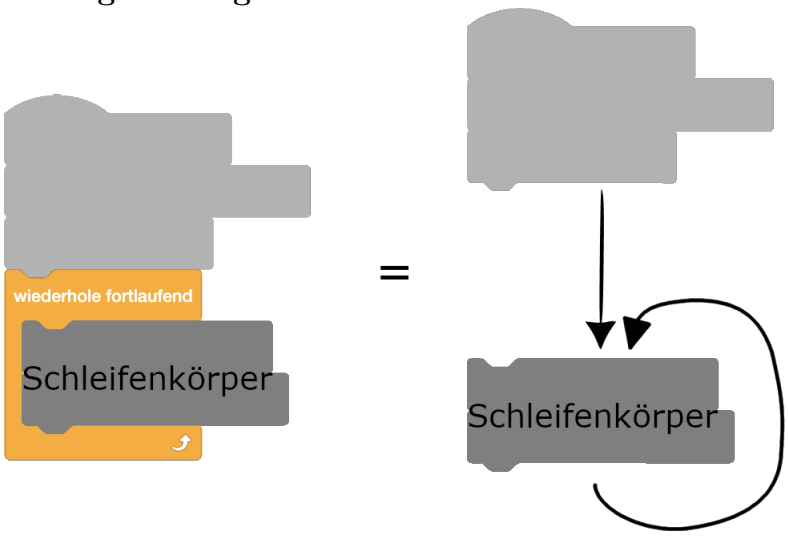
Lassen wir die Maus zum Beispiel mit diesem Skript fortwährend im Kreis laufen, während Scratch links unten startet, so ergibt sich die Spur unten. Wenn wir lange genug warten, ist Scratch auch auf eine Kreisbahn eingeschwenkt.



Lösungen zu Kapitel 6

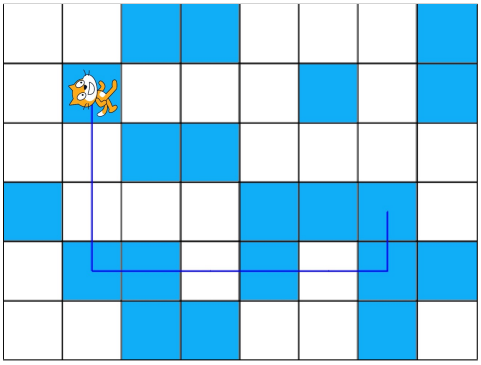
Lösung zu Aufgabe 6.1:
Ein paar Beispiele:
“Falls jemand erwachsen ist, dann sagt man ’Grüezi’, sonst sagt man ’Hoi’.”
“Falls es regnet, dann spiele ich drinnen, sonst spiele ich draussen.”

Lösung zu Aufgabe 6.2:

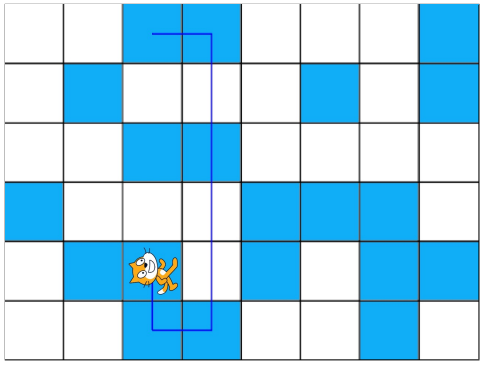


Lösung zu Aufgabe 6.3:
Hier ist zusätzlich der Weg eingezeichnet, den Scratch zurückgelegt hat.

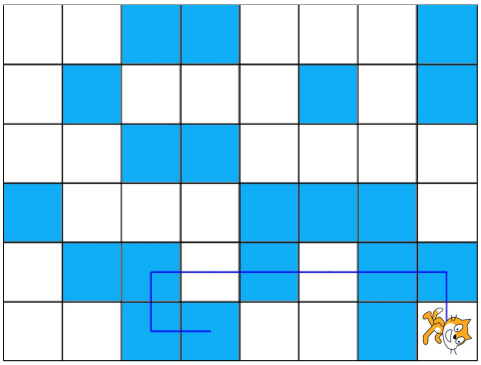
1.



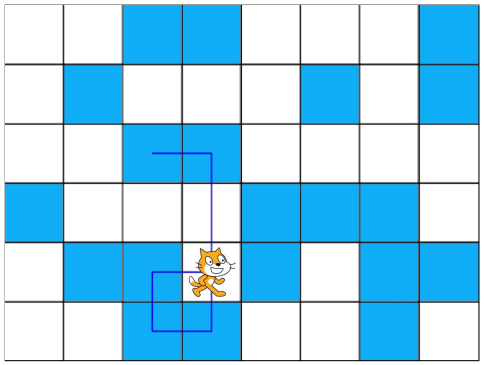
2.



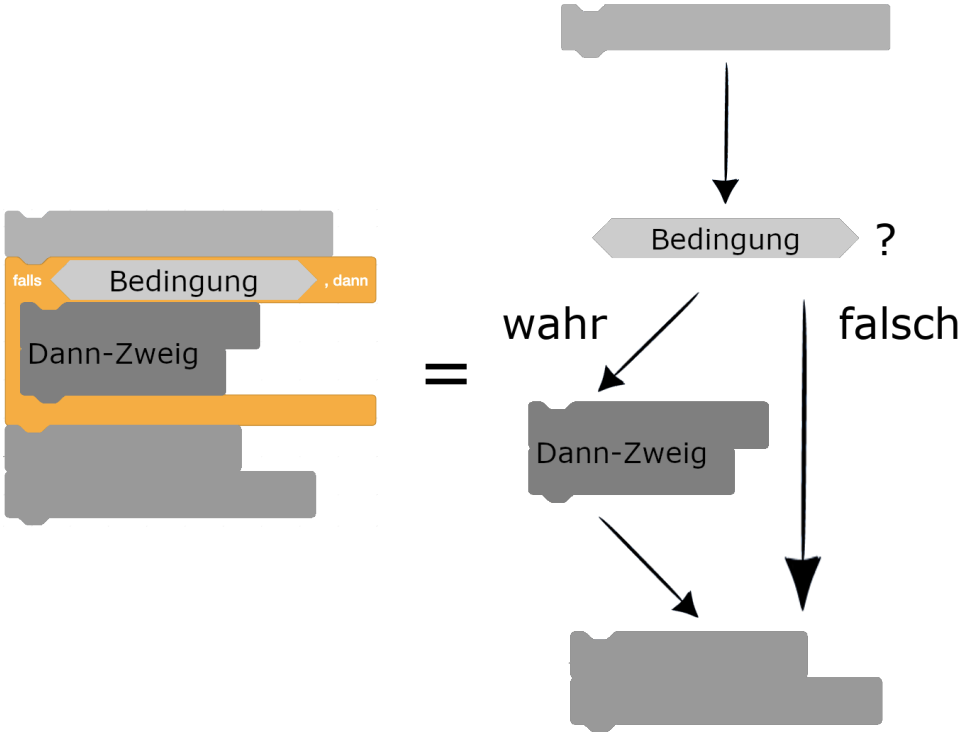
3.



4.



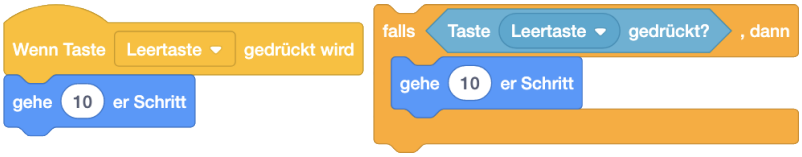
Lösung zu Aufgabe 6.4:



Lösung zu Aufgabe 6.5:

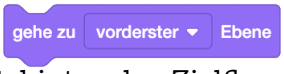
Für diese Aufgabe brauchen wir den Falls-Sonst-Befehl, da wir das Kostüm wieder wechseln müssen, wenn die Leertaste nicht mehr gedrückt ist.

Lösung zu Aufgabe 6.6:

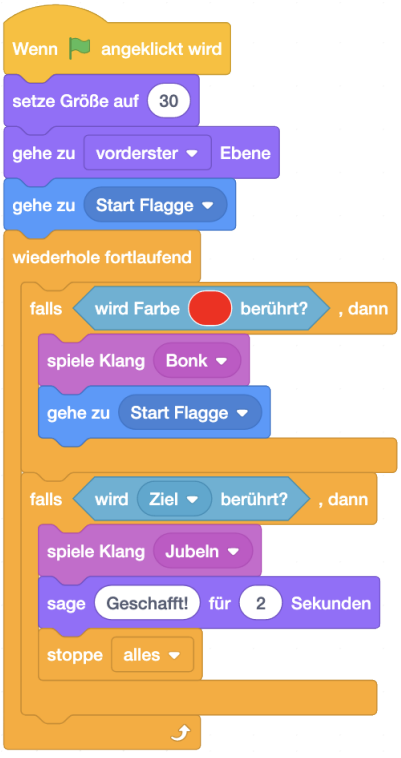


Das linke Skript wird jedes Mal ausgeführt, wenn die Leertaste gedrückt wird. Das rechte Skript wird nur ausgeführt, wenn es angeklickt wird. Dann wird überprüft, ob die Leertaste gedrückt ist. Im linken Skript geht das Objekt also bei jeder Ausführung 10 Schritte, im rechten ist es aber auch möglich, dass bei einer Ausführung nichts passiert. Das rechte Skript kann im Präsentationsmodus nicht ausgeführt werden, da es keinen Hut trägt.

Lösung zu Aufgabe 6.7:

Um Scratch mit den Pfeiltasten steuern zu können, nehmen wir das in der Aufgabenstellung vorgestellte Skript. Zusätzlich erstellen wir dieses Skript:
Mit diesem Skript passen wir als erstes die Grösse so an, dass das Objekt durch das Labyrinth hindurchlaufen kann. Der Befehl 

sorgt dafür, dass Scratch nicht hinter der Zielflagge oder dem Regenbogen angezeigt wird. Die richtige Farbe in der Bedingung kannst du mit der Pipette auswählen. Es muss die gleiche Farbe sein wie die Farbe der Wände des Labyrinths. Die Start- und Zielobjekte, sowie die Klänge heissen bei dir vielleicht anders.



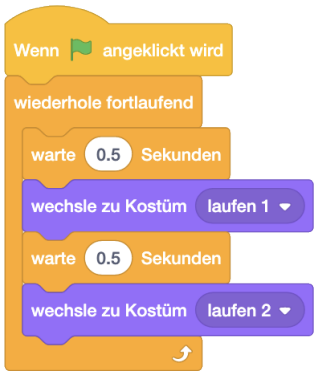
Lösung zu Aufgabe 6.8:

In dieser Aufgabe brauchen wir mehrere Skripte, die jeweils eine Aufgabe erfüllen. So können mehrere Aktionen gleichzeitig ausgeführt werden.

Um das Objekt steuern zu können, benutzen wir dieses Skript:
Wir setzen dabei zuerst das Objekt an den Start und lassen es in die Startrichtung schauen. Der Klang zu Beginn ist nicht notwendig, verleiht deinem Projekt aber eine eigene Note!

Um zu überprüfen ob das Objekt die Strecke verlässt fügen wir dieses Skript hinzu:
Die richtige Farbe in der Bedingung kannst du mit der Pipette auswählen. Hier muss darauf geachtet werden, dass die Berührung mit dem Streckenrand aus einem vorherigen Spiel das jetzige Spiel nicht beenden soll. Daher warten wir zuerst eine gewisse Zeit, bis das Objekt gestartet ist.

Die Laufanimation erzielen wir mit diesem Skript:



Und die Jubelklänge fügen wir mit diesem Skript für das Ziellinien-Objekt hinzu:
Hier muss darauf geachtet werden, dass die Berührung der Ziellinie am Anfang noch kein Jubeln auslösen soll. Daher warten wir zuerst eine gewisse Zeit, bis das Objekt sicher nicht mehr auf der Ziellinie steht.

Lösung zu Aufgabe 6.9:

Um die Leben zu zählen kombinieren wir das Skript für die Steuerung des Objekts und das Skript zum Testen ob der Rand berührt wird aus der Lösung zu Aufgabe 6.8 zu diesem Skript:
Wir haben dabei auch den Startpunkt geändert. Das Objekt wird kurz hinter die Ziellinie gesetzt, denn sonst erklingt der Jubel auch dann wenn das Objekt wieder an den Start gesetzt wird, nachdem es in den Rand gefahren ist.

Um die Runden zu zählen erstellen wir dieses Skript für die Ziellinie:

Man kann alternativ auch ein ähnliches Skript für das Objekt erstellen, in dem die Berührung mit der Ziellinie überprüft wird. Wichtig ist, dass der 

-Block innerhalb der Schleife ist. Die Bedingung im Warte-Bis-Befehl gilt, solange das Rennauto über die Ziellinie fährt. Ohne den Warte-Befehl überprüft der Computer diese Bedingung bereits dreimal in dieser kurzen Zeit und das Objekt löst nach der ersten Runde bereits Jubel aus.

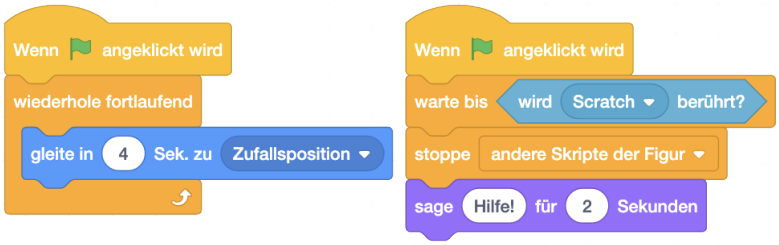
Die beiden Erweiterungen lassen sich nur gleichzeitig benutzen, wenn das Objekt nicht direkt auf die Ziellinie gesetzt wird, wenn es den Rand der Rennbahn berührt hat. Ansonsten zählt diese Berührung als vollendete Runde und man kann das Spiel einfacher gewinnen, indem man den Rand absichtlich berührt. Wenn man will, dass nur eine bestimmte Anzahl Runden direkt nacheinander ohne Randberührung zählt, um das Spiel zu gewinnen, dann lassen sich die Lösungen mit den Blöcken die wir bis jetzt kennen nicht kombinieren. Wir werden aber im nächsten Kapitel eine Möglichkeit kennenlernen, mit der man dies doch erreichen kann.

Lösung zu Aufgabe 6.10:

Die Maus überprüft die Bedingung nur alle 4 Sekunden. Dazwischen führt sie den 

Befehl vollständig aus.

Damit die Maus sofort anhält, teilen wir ihr Skript in zwei Skripte auf: ein Skript übernimmt die Bewegung während das andere Skript die Bedingung überprüft und das Bewegungsskript stoppt, wenn die Bedingung eintrifft.



Lösung zu Aufgabe 6.11:

Für den Ball verwenden wir dieses Skript, wobei der Startpunkt und die Startrichtung auch anders gewählt werden kann. Ebenso kann der Ball mit etwas kleinerer oder grösserer Schrittgrösse langsamer oder schneller programmiert werden.

Für den Schläger verwenden wir dieses Skript, wobei auch hier der y-Wert variieren kann, je nachdem wie gross der Schläger ist:

Lösungen zu Kapitel 7

Lösung zu Aufgabe 7.1:

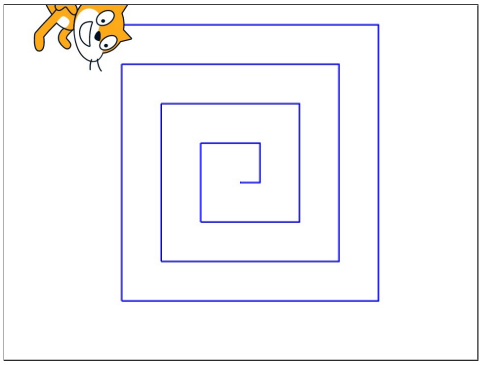
Das Skript für den Schläger sieht nun so aus:

Lösung zu Aufgabe 7.2:

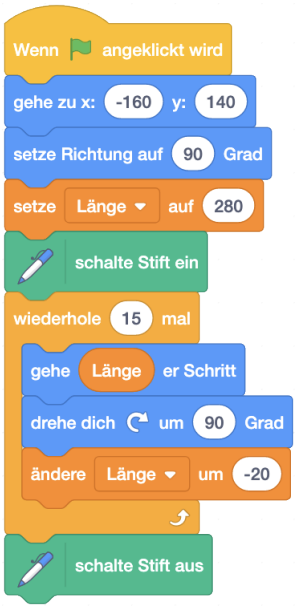
1. Die Variable "meine Variable" hat den Wert 20 und Scratch sagt 20.
2. Der Wert der Variable "meine Variable" wird mit dem Wert von "zweite Variable" ersetzt. Somit haben am Ende beide Variablen den Wert 20.
3. Auch hier wird zuerst der Wert der Variable mit dem Namen "meine Variable" mit dem Wert von "zweite Variable" ersetzt. Danach wird der Wert der Variable "zweite Variable" auf 30 gesetzt. Dies hat keinen Einfluss auf den Wert von "meine Variable". Somit hat am Ende "meine Variable" den Wert 20 und "zweite Variable" den Wert 30.
4. Diese Variablen wurden schlecht benannt. Der Computer führt die Befehle einfach genau so aus, wie sie dastehen, auch wenn wir zum Beispiel wissen, dass 100 die grössere Zahl ist und 1 die kleinere. Beide Variablen haben am Schluss den Wert 100.
5. Die Variable "Zähler" hat am Schluss den Wert 5. Scratch sagt 5.
6. Die Variable "Zähler" hat am Schluss den Wert 5. Scratch sagt nacheinander 1, 2, 3, 4 und 5.
7. Die beiden Variablen "a" und "b" tauschen ihre Werte. Somit hat "a" am Schluss den Wert 5 und "b" den Wert 4. Die Variable "c" hat ebenfalls den Wert 4. Sie

ist nötig, damit wir beim Austauschen der Werte nicht den Wert einer Variable überschreiben, ohne ihn vorher woanders zu speichern.

Lösung zu Aufgabe 7.3:
Scratch zeichnet eine Spirale:



Mit diesem Skript malt Scratch die Spirale von aussen nach innen:



Lösung zu Aufgabe 7.4:

Wir fügen das folgende Skript für den Ball unserem Programm hinzu. Wir definieren eine neue Variable “Punkte”, mit der wir mitzählen, wie oft der Ball den Schläger berührt hat. Wir müssen die Variable also zu Beginn auf 0 setzen. Bei jeder Berührung des Schlägers (kann auch über die Farbe getestet werden) erhöhen wir die Variable um 1. Wenn du bei der Variable das Häkchen setzt, kann man mitverfolgen, wie viele Punkte man bereits gesammelt hat.

Lösung zu Aufgabe 7.5:

1. Da die Bedingung falsch ist, wird nur der zweite Ändere-Variable Befehl ausgeführt und die Variable “meine Variable” hat am Ende den Wert 40.

2. Die Variable “meine Variable” hat den Wert 5, die Variable “zweite Variable” den Wert 4 und die Variable “gesamt” hat den Wert 3. Scratch sagt 3.

3. Die Variable “meine Variable” hat den Wert 3. Scratch sagt zuerst 2 und dann 3.

4. Die Variable “meine Variable” hat den Wert 4. Scratch sagt 9.

5. Die Variable “meine Variable” hat den Wert 9. Scratch sagt 9.

Lösung zu Aufgabe 7.6:

1. Die folgende Tabelle zeigt den Wert von “meine Variable” am Ende des jeweiligen Durchlaufes der Schleife.

Durchlauf:	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Wert:	1	2	3	4	5	6	4	5	6	4	5	6	4	5	6

2. Da die Bedingung bereits zu Beginn stimmt, wird die Schleife gar nicht ausgeführt und der Wert von “meine Variable” ist 100.

3. Der Wert der Variable “meine Variable” wird mehrmals halbiert: 100, 50, 25. Da 25 noch nicht kleiner ist als 25, wird die Schleife nochmals ausgeführt und “meine Variable” hat zum Schluss den Wert 12.5.

Lösung zu Aufgabe 7.7:

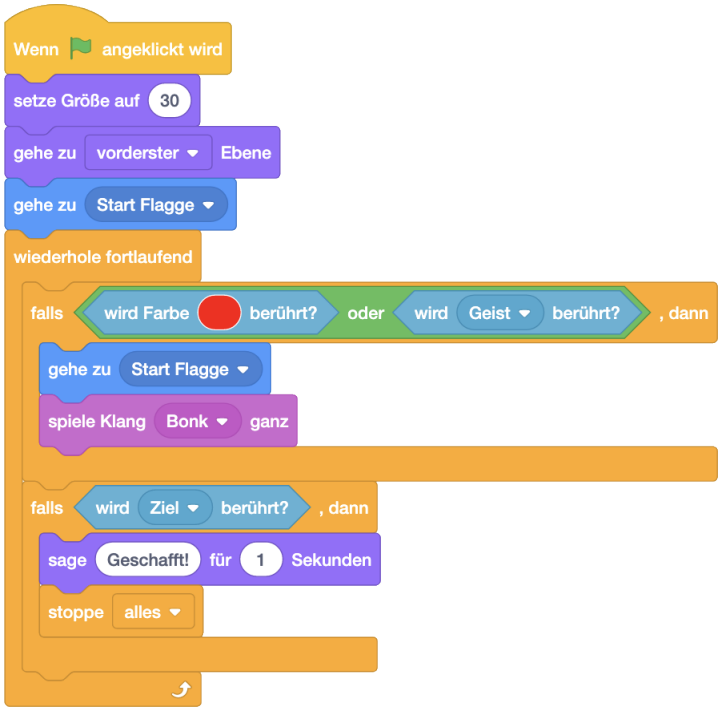
1. Wahrheits-Block	2. Wahrheits-Block	1. Wahrheits-Block oder 2. Wahrheits-Block
false	false	false
false	true	true
true	false	true
true	true	true

Der Unterschied zum alltäglichen Sprachgebrauch liegt in der letzten Zeile. Häufig meint man mit "oder", dass nur *entweder* das eine *oder* das andere wahr sein darf und nicht beides gleichzeitig. In der Logik gilt es aber auch als wahr, wenn beide Bedingungen wahr sind.

Lösung zu Aufgabe 7.8:

Das Bewegungsskript für Scratch verändert sich nicht. Das andere Skript hat auch nur eine kleine Änderung: Wir überprüfen nicht nur die Berührung mit den Wänden des Labyrinths sondern auch die Berührung mit dem Monster.

Skript für Scratch:



Skript für das Monster:

Lösung zu Aufgabe 7.9:

Wichtig ist, dass deine gezeichneten Zeiger horizontal sind, wenn die Richtung auf 90 Grad gesetzt ist. Die Umrechnung von Sekunde / Minute / Stunde zu Grad geht folgendermassen:

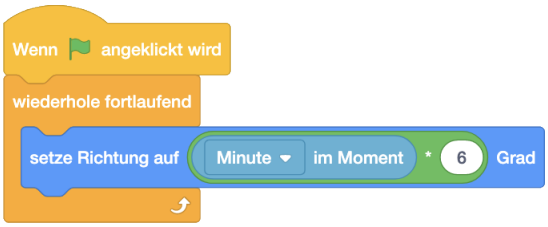
60 Sekunden = 360 Grad
→ 1 Sekunde = 360 Grad / 60 = 6 Grad
→ Winkelrichtung = Anzahl Sekunden · 6 Grad

60 Minuten = 360 Grad
→ 1 Minute = 360 Grad / 60 = 6 Grad
→ Winkelrichtung = Anzahl Minuten · 6 Grad

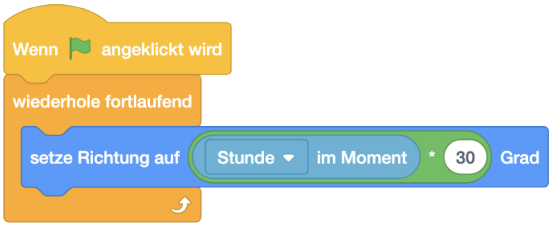
12 Stunden = 360 Grad
→ 1 Stunde = 360 Grad / 12 = 30 Grad
→ Winkelrichtung = Anzahl Stunden · 30 Grad

Skript für den Sekundenzeiger:

Skript für den Minutenzeiger:

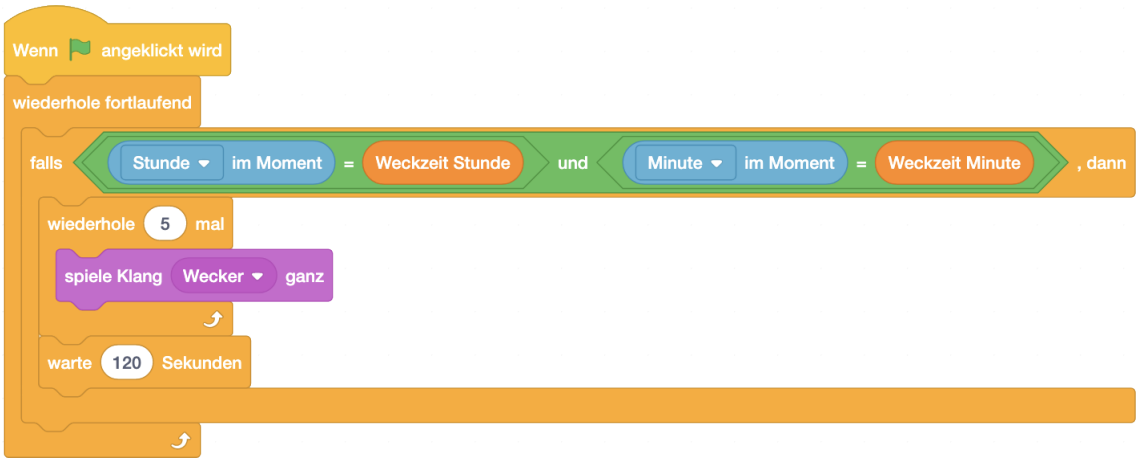


Skript für den Stundenzeiger:



Zusatzaufgaben:

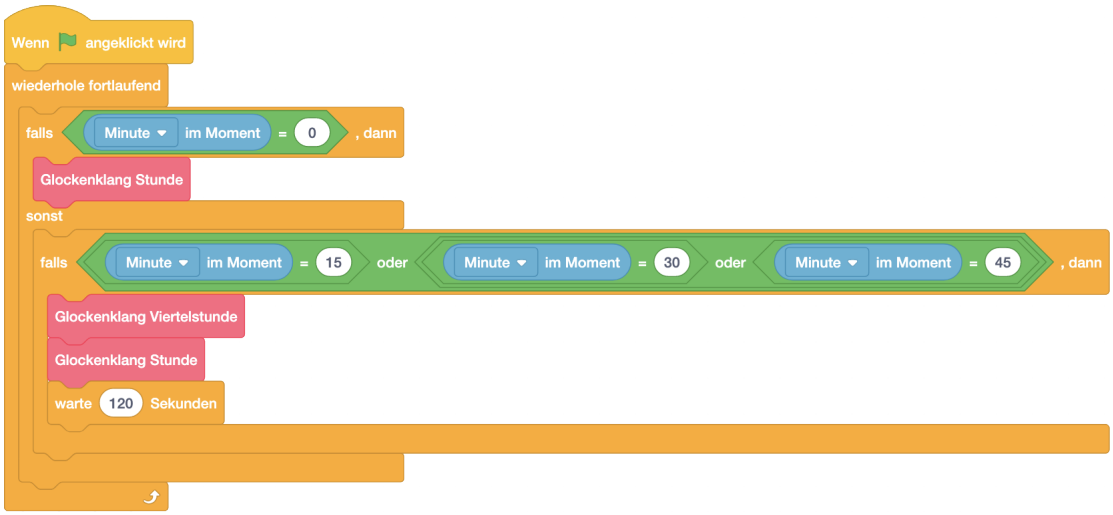
1. Wir definieren zwei zusätzliche Variablen: “Weckzeit Stunde” und “Weckzeit Minute”. Wir setzen bei beiden Variablen das Häkchen, damit sie auf der Bühne angezeigt werden. Mit Rechtsklick verändern wir die Darstellung zu einem Schieberegler, und mit nochmals Rechtsklick setzen wir den Wertebereich. Für “Weckzeit Minute” wählen wir hier 0-59 und für “Weckzeit Stunde” 0-23. Dann erstellen wir das folgende Skript (für welches Objekt spielt hier keine Rolle):



Der "Warte"-Befehl ist notwendig, damit der Wecker nicht mehrmals während der gleichen Minute erklingt. Man kann zum Beispiel auch ein neues Objekt hinzufügen - den Wecker - der bei Anklicken den Weckklang abbrechen soll:

Man muss dabei auch wieder eine Schleife verwenden, denn sonst wird nur der jetzige Weckklang gestoppt und die restlichen erklingen trotzdem.

2. Wir erstellen die folgenden Skripte:



Der 

Befehl ist notwendig, damit die Glockenschläge nicht während derselben Minute nochmals ausgelöst werden.

Um dein Programm zu testen kannst du zwei Variablen erstellen (“Minuten” und “Stunden”). Diese setzt du auf die Uhrzeit, die du testen willst. Statt dem 

Block benutzt du dann die entsprechenden Variablenblocks. Dann überprüfst du, ob dein Programm für die gewählte Uhrzeit richtig funktioniert.

Lösung zu Aufgabe 7.10:

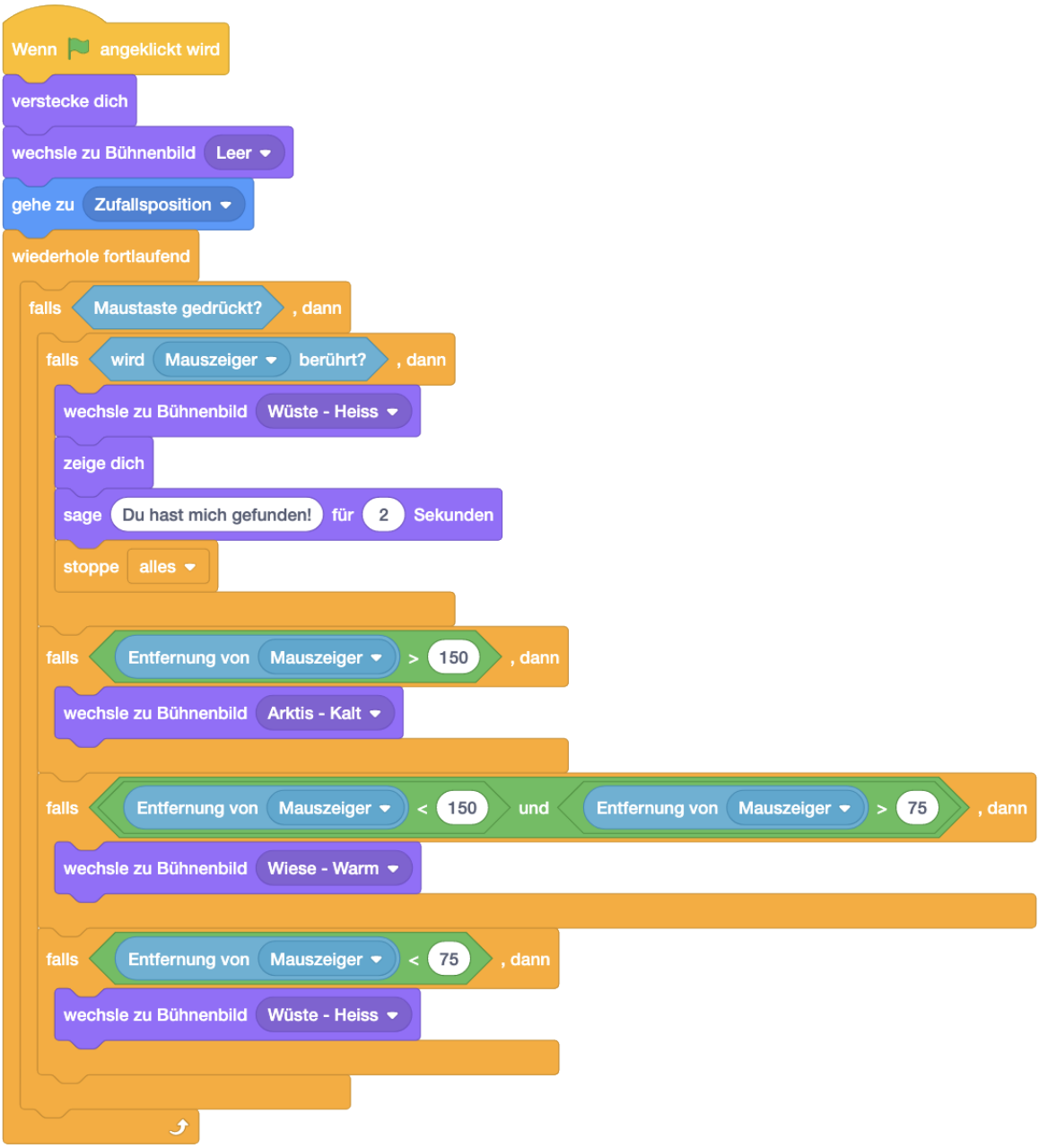
Wir brauchen eine Variable, die mitzählt, wie oft die Schleife schon wiederholt wurde. Dazu müssen wir sie zuerst auf 0 setzen. Am Ende jeder Wiederholung erhöhen wir die Variable um 1, somit entspricht der Wert der Variable der Anzahl Wiederholungen. Bevor die nächste Wiederholung ausgeführt wird, wird überprüft, ob schon genug Wiederholungen ausgeführt worden sind. Da wir bei diesem Beispiel schon wissen, was eine Wiederholung macht, können wir hier der Variable auch einen spezifischeren Namen geben, wie zum Beispiel “Ecken”.

Diese Art von Schleife gibt es in vielen Programmiersprachen. Die Variable, die dabei die Wiederholungen mitzählt, heisst *Laufvariable*. Eine Laufvariable kann hilfreich sein, wenn man zum Beispiel bei jeder dritten Wiederholung etwas anderes machen will.

Lösung zu Aufgabe 7.11:

Lösung zu Aufgabe 7.12:
Man kann Scratch nicht sagen lassen, wie weit der Mausklick entfernt war, weil man nicht sehen kann, was versteckte Objekte sagen. Ausserdem würden wir durch die Sprechblase ja direkt sehen, wo Scratch sich versteckt hat! Man muss hier auch aufpassen, dass sich Scratch zuerst versteckt und dann zu einer zufälligen Position geht.

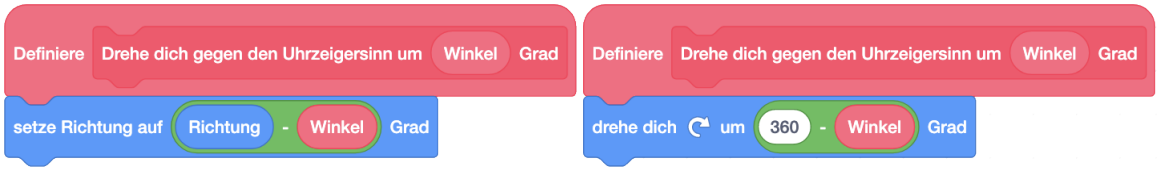
Lösung der Variante:



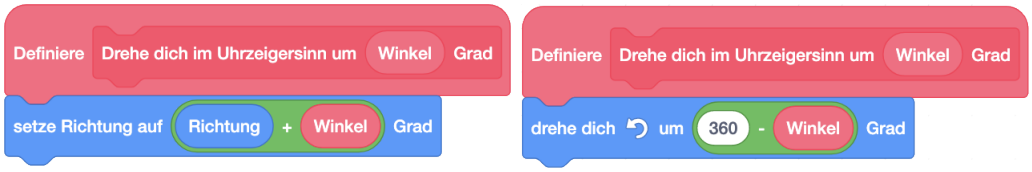
Lösung zu Aufgabe 7.13:
Wir erstellen einen Block, mit dem wir ausgefüllte Halbkreise zeichnen können. Der Parameter "Radius" gibt dabei den Radius an, den der Halbkreis haben soll. Für einen Regenbogen brauchen wir 7 solche Halbkreise, wobei wir mit dem grössten anfangen, dann den zweitgrössten darüber zeichnen, und so weiter. Zuletzt zeichnen wir noch einen weissen Halbkreis, damit es mehr wie ein Bogen aussieht. Die Halbkreise zeichnen wir mit einer Schleife, wobei eine Variable die Grösse angibt. Wir ändern ausserdem in jedem Durchlauf der Schleife die Farbe. Die Stiftdicke muss (durch Ausprobieren) gross genug gewählt werden, damit auch die grossen Halbkreise ausgefüllt sind.

Lösung zu Aufgabe 7.14:

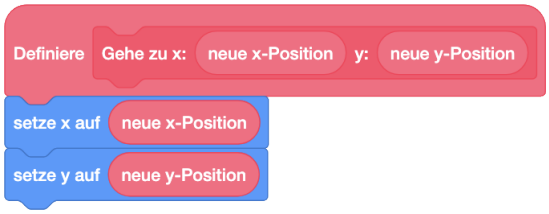
1. Es gibt zwei verschiedene Varianten. Entweder man dreht Scratch im Uhrzeigersinn bis die Richtung stimmt oder man setzt direkt die neue Richtung.



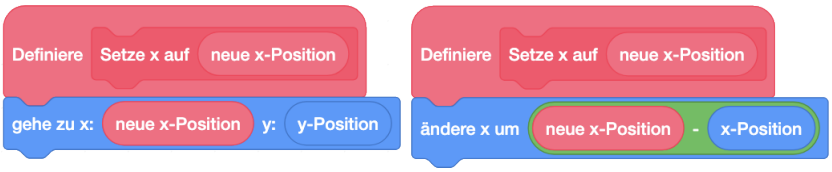
2. Es gibt zwei verschiedene Varianten. Entweder man dreht Scratch im Gegenuhrzeigersinn bis die Richtung stimmt oder man setzt direkt die neue Richtung.



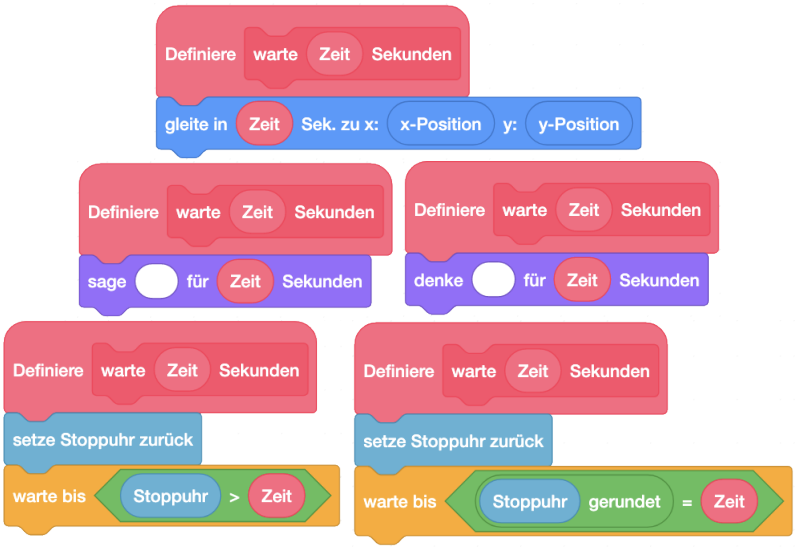
3.



4. Hier gibt es auch wieder zwei Varianten:



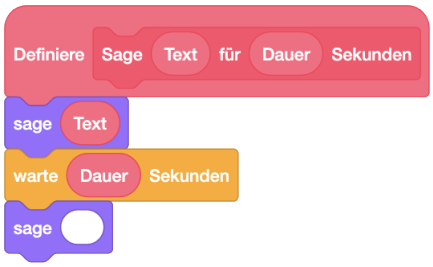
5. Hier gibt es viele verschiedene Lösungen. Jede Lösung hat aber auch Nachteile. Bei den ersten drei Befehlen gibt es einen Nebeneffekt: Wenn Scratch zum Beispiel in einem Skript gerade etwas sagt, und in einem anderen Skript zur gleichen Zeit warten soll, dann verschwindet das Gesagte. Bei der 4. Lösung muss man den $>$ Befehl benutzen. Wenn man es mit dem $=$ Befehl versucht, kann es sein, dass der Computer die Bedingung nicht genau in der Millisekunde überprüft, in der der Timer den gleichen Wert antwortet. Mit Runden lässt sich das umgehen. Hier bleibt aber das Problem bestehen, dass man bei diesem Befehl auch eine negative Zahl eingeben kann. Dann wartet Scratch mit den letzten beiden Lösungen unendlich lange!



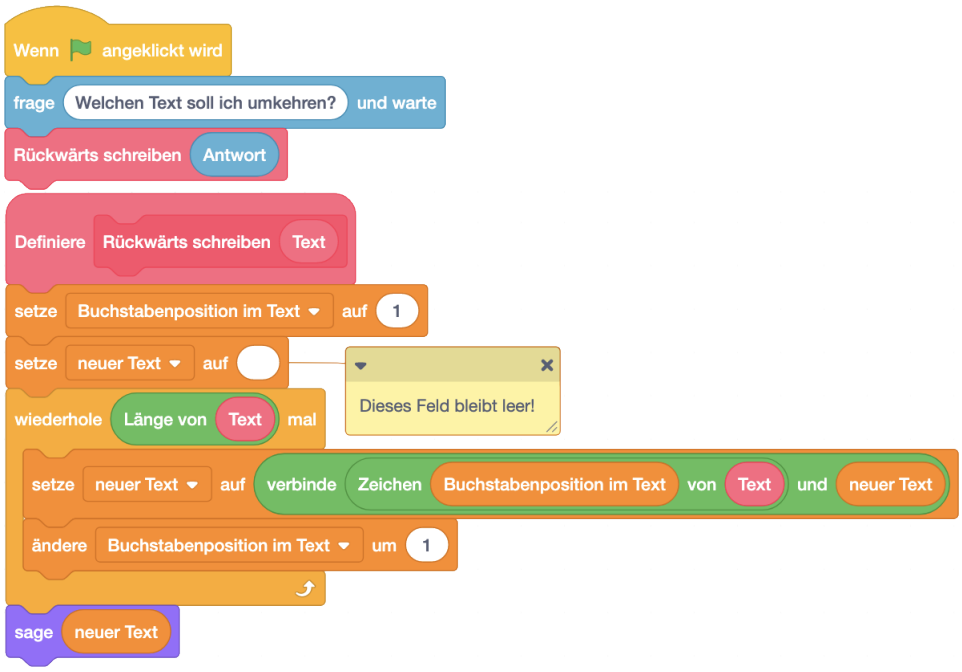
6. Wenn man hier davon ausgeht, dass die anderen Befehle auch kaputt sind, muss man statt dem 

-Befehl eine der Lösungen aus Teilaufgabe 5 benutzen (aber natürlich nicht diejenige, die den 

-Befehl braucht).

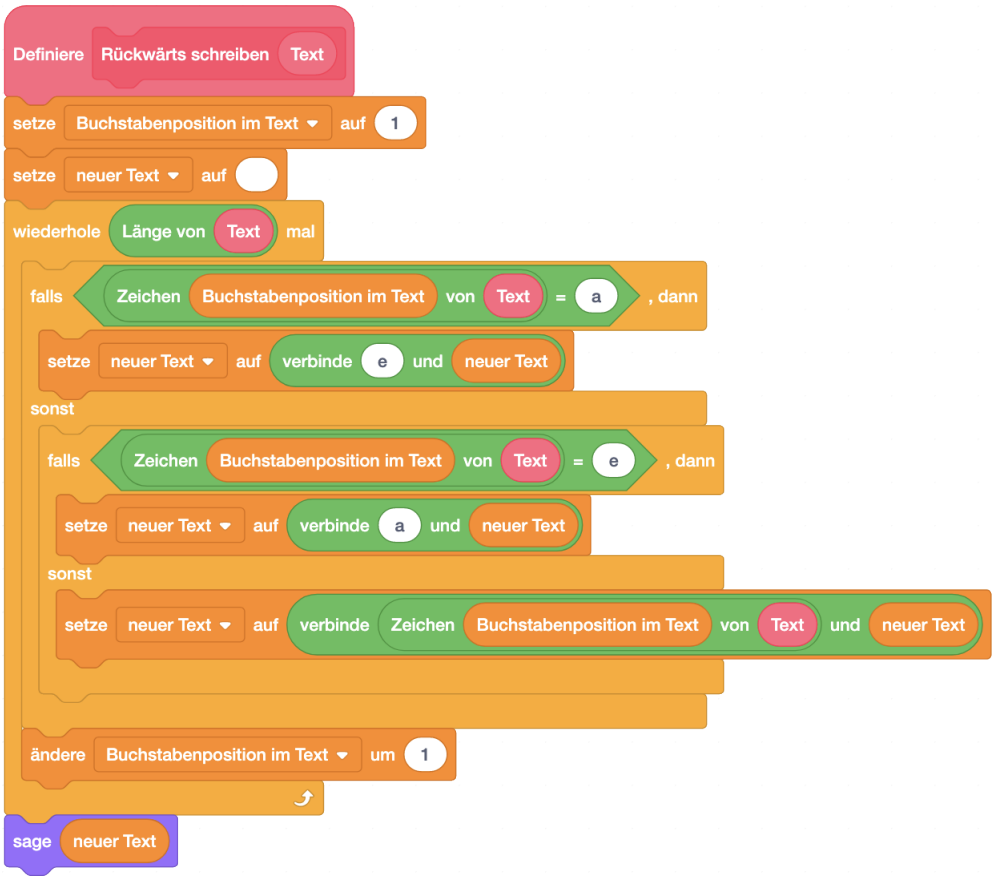


Lösung zu Aufgabe 7.15:
Das Programm wird zum folgenden Programm ergänzt:



Dabei wird Schritt für Schritt von vorne durch den Text durchgegangen und der jeweilige Buchstabe vorne an den neuen Text angefügt. Somit hat man am Schluss den letzten Buchstaben des Originaltextes zuerst im neuen Text und den ersten zuletzt.

Die vorgeschlagene Erweiterung lässt sich folgendermassen programmieren:


Definiere Rückwärts schreiben Text
setze Buchstabenposition im Text auf 1
setze neuer Text auf
wiederhole Länge von Text mal
falls Zeichen Buchstabenposition im Text von Text = a , dann
setze neuer Text auf verbinde e und neuer Text
sonst
falls Zeichen Buchstabenposition im Text von Text = e , dann
setze neuer Text auf verbinde a und neuer Text
sonst
setze neuer Text auf verbinde Zeichen Buchstabenposition im Text von Text und neuer Text
ändere Buchstabenposition im Text um 1
sage neuer Text

Lösungen zu Kapitel 8

Lösung zu Aufgabe 8.1:
Das Skript für der Schläger wird nicht verändert:

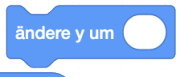
Wir überlegen uns zuerst, wann wir Nachrichten brauchen. Wir wollen, dass die Taste darauf reagieren kann, dass der Ball den Boden berührt (Befehl zeige dich !). Diese Nachricht vom Ball nennen wir "Nochmals?". Ausserdem muss der Ball darauf reagieren können, dass die Taste angeklickt wird (Spiel neu starten). Diese Nachricht von der Taste nennen wir "Nochmals!". Ausserdem erstellen wir für den Beachball einen eigenen Block für das Spielen, da wir das Spiel ausführen wollen, wenn die grüne Flagge angeklickt wird, oder wenn wir von der Taste die Nachricht kriegen, dass sie angeklickt wurde. Es ist wichtig, dass im Skript der Taste nicht der sende Nachrichtentext an alle und warte Befehl verwendet wird, denn sonst versteckt sich die Taste erst wieder, wenn das Spiel zu Ende ist. Man könnte hier aber auch den verstecke dich Befehl vor dem sende Nachrichtentext an alle Befehl verwenden. Dann spielt es keine Rolle, ob man wartet, bis das zugehörige Skript zur Nachricht fertig ist oder nicht.

Skripte Ball

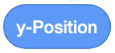
Skripte Taste

Lösung zu Aufgabe 8.2:
Für diese Aufgabe muss man den Sende-Nachricht (ohne warten) verwenden, da wir wollen, dass die Instrumente gleichzeitig spielen. Man könnte aber auch den Sende-Nachricht-und-warte Befehl verwenden, wenn zum Beispiel mehrere Objekte gleichzeitig auf diese Nachricht reagieren und das Musikstück erst weitermachen soll, wenn diese Instrumente fertig sind.

Lösung zu Aufgabe 8.3:
Für diese Aufgabe muss man den Sende-Nachricht-und-warte Befehl verwenden, denn die Figuren sollen sich nicht gleichzeitig vorstellen.

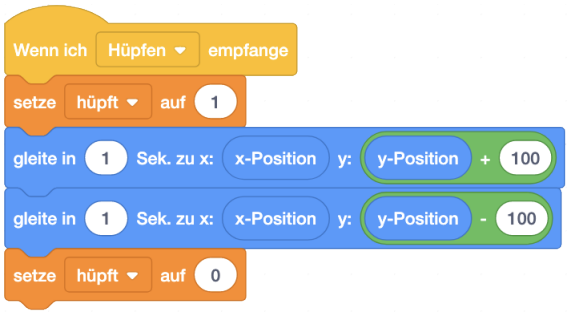
Lösung zu Aufgabe 8.4:
Weil wir wollen, dass die Bewegung flüssig ist, können wir nicht den 

Block benutzen. Wir verwenden die Wert-Blöcke 

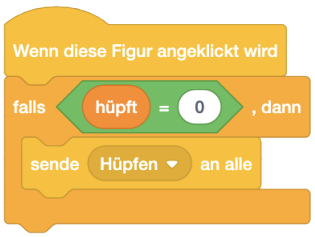
und 

sowie Addition und Subtraktion um Scratch nach oben und wieder nach unten gleiten zu lassen. Um zu verhindern, dass Scratch immer weiter hoch hüpft, wenn der Knopf angeklickt wird benutzen wir eine Variable. Die "Hüpfen"-Nachricht senden wir nur, wenn Scratch nicht gerade am Hüpfen ist.

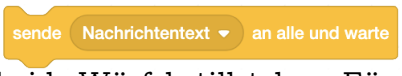
Skript für Scratch:



Skript für den Knopf:

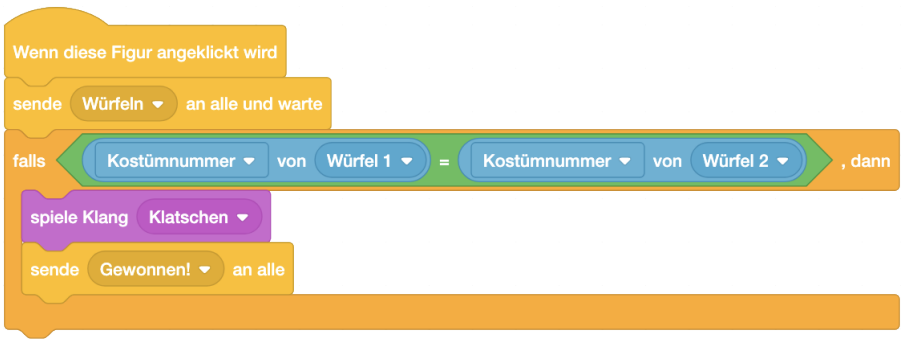


Lösung zu Aufgabe 8.5:

Wir überlegen uns wieder zuerst, wann wir Nachrichten brauchen. Die beiden Würfel sollen darauf reagieren, wenn der Knopf angeklickt wird. Dazu verwenden wir die Nachricht "Würfeln". Der Hintergrund der Bühne soll auf die Gewinn-Bedingung reagieren. Da wir das Würfeln durch mehrere Kostümwechsel hintereinander darstellen, lassen wir den Knopf die Bedingung überprüfen. So können wir sicherstellen, dass der Hintergrund nicht auf einen "Zwischenstand" der Würfel reagiert. Wir brauchen im Skript der Taste den 

Befehl, damit die Bedingung erst überprüft wird, wenn beide Würfel stillstehen. Für die "Gewonnen"-Nachricht spielt es keine Rolle, ob wir warten oder nicht. Beide Würfel haben die gleichen Skripte.

Skript für den Knopf:



Skripte für die Würfel:

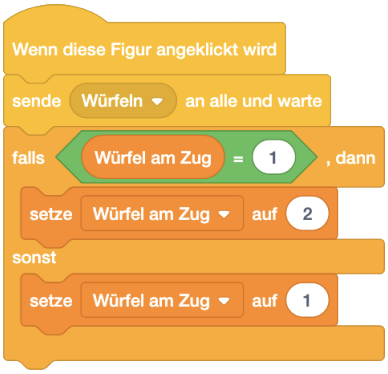
Skript für die Bühne:

Zusatz-Aufgaben:

1. Die neue Gewinn-Bedingung kann nun zum Beispiel so aussehen:

2. Welcher Würfel gerade am Zug ist kann man zum Beispiel durch eine Variable steuern. Der Wert der Variable soll angeben, welcher Würfel gerade am Zug ist. Nach jedem Würfeln setzt die Taste die Variable auf 1, wenn sie vorher den Wert 2 hatte, und umgekehrt. Die Würfel müssen dann noch überprüfen, ob sie gerade am Zug sind. In dieser Lösung überprüfen die Würfel die Gewinn-Bedingung (man gewinnt, wenn man eine 6 würfelt). Das Skript für die Bühne bleibt unverändert.

Skript für die Taste:



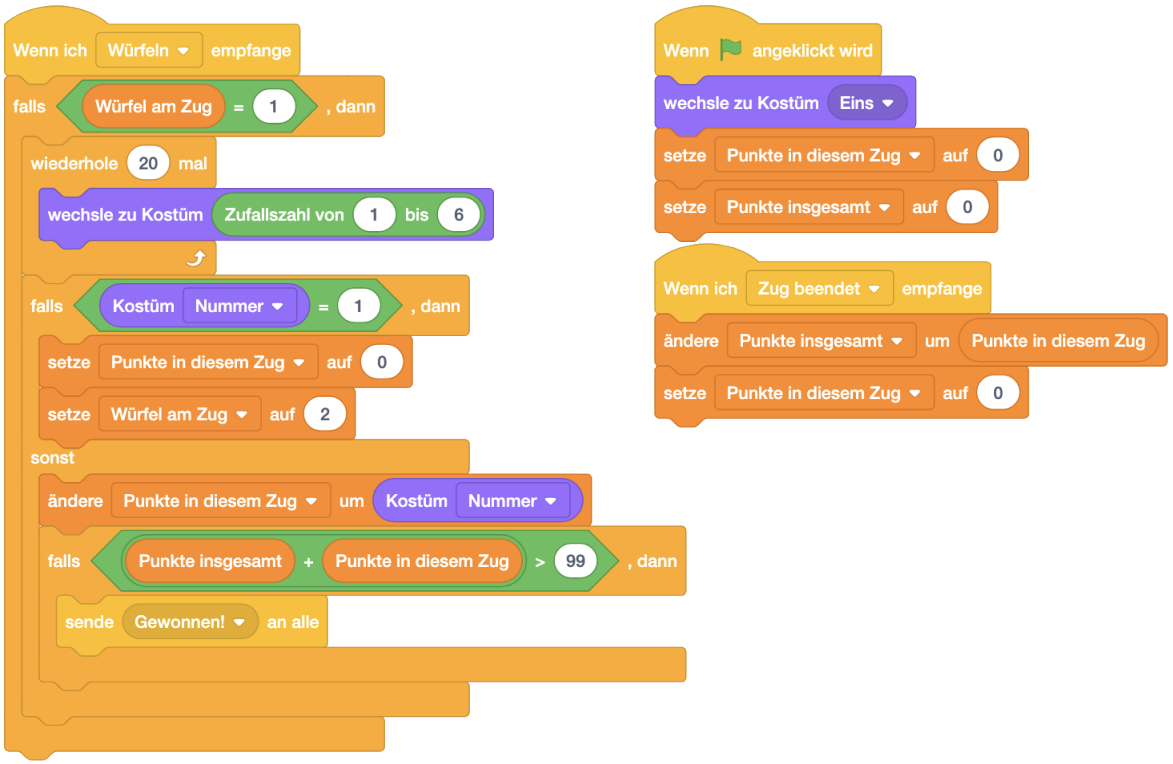
Skripte für Würfel 1:

Skripte für Würfel 2:

3. Auch für diese Aufgabe verwenden wir die Variable, die angibt, welcher Würfel gerade am Zug ist. Hier soll aber nicht nach jedem Würfeln gewechselt werden, sondern erst, wenn eine Eins gewürfelt wird oder wenn man den Zug beendet. Wir brauchen ausserdem noch jeweils zwei weitere Variablen für jeden Würfel. Eine Variable zählt, wieviele Punkte man im jetzigen Zug erreicht hat, die andere zählt die Gesamtpunktzahl. Damit die jeweilige Variable nur für den zugehörigen Würfel gilt, wähle beim Erstellen der Variable "Nur für diese Figur" aus. Dadurch kannst du die Variablen gleich benennen, es kann aber nur das Objekt, für das du die Variable erstellt hast, diese Variable verändern. Du kannst aber auch jeweils eine öffentliche Variable verwenden, dann musst du aber verschiedene Namen verwenden. Das gesamte Programm sieht dann wie folgt aus:

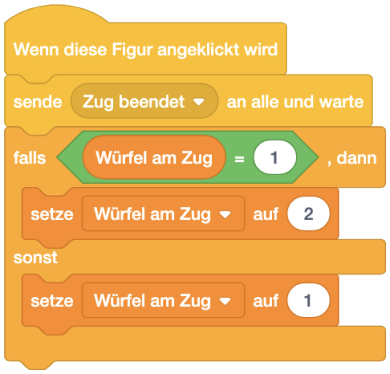
Skripte für die "Würfeln"-Taste:

Skripte für Würfel 1:

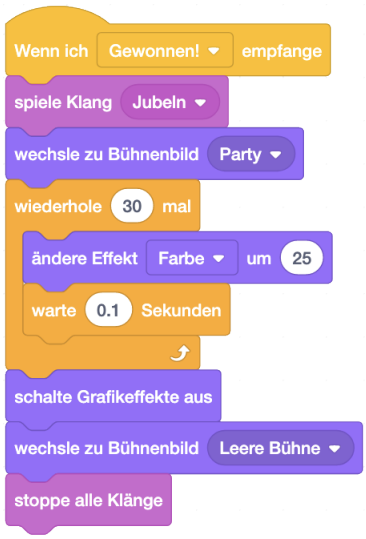


Skripte für Würfel 2:

Skript für die "Zug beenden"-Taste:



Skript für die Bühne:



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