Erste Schritte mit Scratch

Lösungen - Hintergründe - Anwendungen

Inhaltsverzeichnis

Vorwort

Hallo Freunde der Programmierung,

herzlich willkommen im Lehrhandbuch.

Der Aufbau des Programmes Scratch ist zu vergleichen mit einem Legobaukasten. Es gibt Bausteine mit verschiedenen Farben, Formen und Funktionen. Je nachdem wie man diese Bausteine dann zusammensteckt erhält man ein dementsprechendes Ergebnis.

Obwohl ein Großteil der angeführten Beispiele mit der Figur Scratch veranschaulicht wird, funktioniert jeder Baustein auch mit jeder anderen, beliebigen Figur in gleichem Maße.

Nicht jeder Baustein, den es in Scratch gibt, wird in diesem Handbuch erklärt. Manche wurden aufgrund fehlender Relevanz ausgelassen. Jedoch sind alle ausgelassenen Bausteine, der Vollständigkeit halber, in einer Liste zusammengefasst und kurz beschrieben worden.

Dieses Lehrhandbuch ist das zugehörige Gegenstück zum Aufgabenhandbuch. In dem für Kinder gestaltete Aufgabenhandbuch sind viele gute Beispiele, Aufgaben und Informationen enthalten. Die Beispiele aus dem Aufgabenhandbuch werden in diesem Lehrhandbuch erklärt und auch deren Lösungen aufgezeigt.

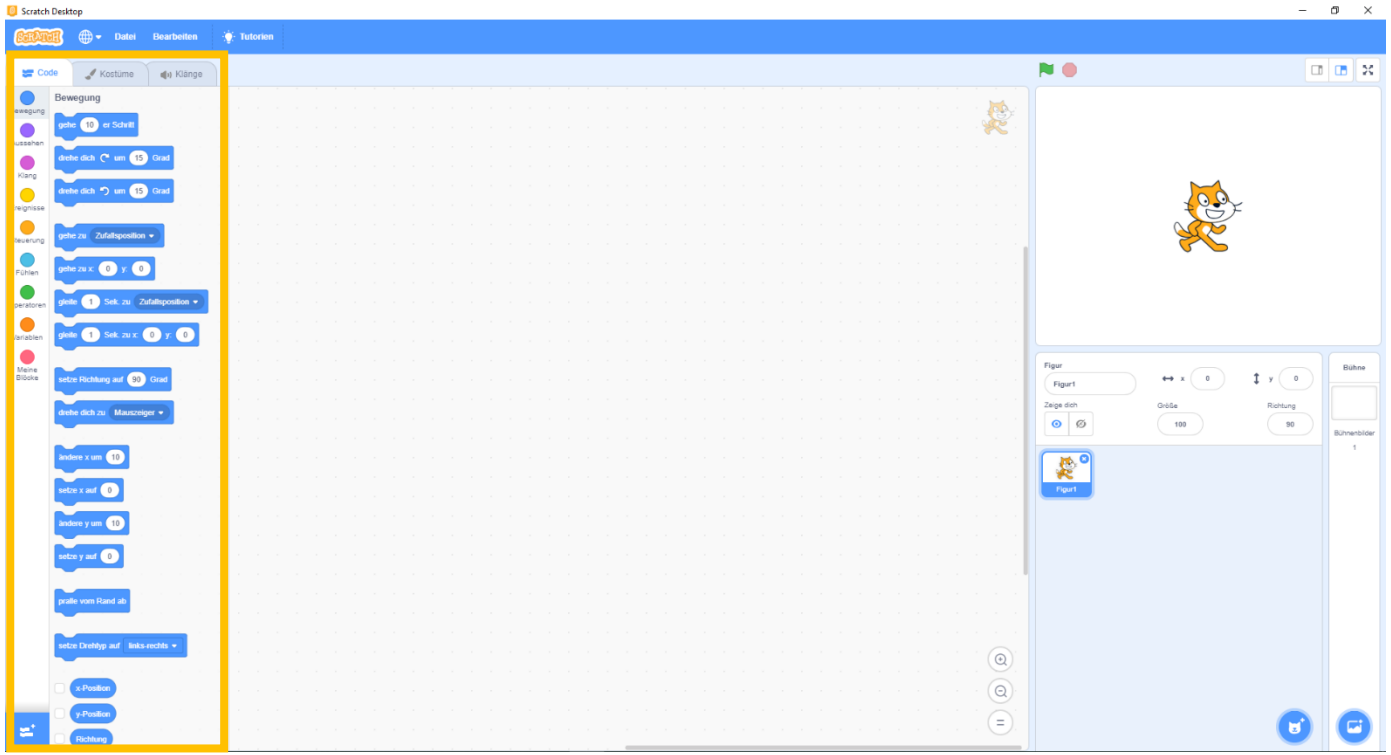
Des Weiteren kann das Lehrhandbuch als Erweiterung zum Aufgabenhandbuch genutzt werden. Die zu den Bausteinen angeführten Beispiele, können als erweitertes Übungsmaterial verwendet und ganz einfach nachgebaut werden.

Dieses Handbuch ist mit Sicherheit hilfreich, um den Einstieg in die Programmierwelt einfacher zu gestalten. Dennoch ist es wichtig immer wieder Dinge selbst auszuprobieren und sich eigene Beispiele zu überlegen und diese aufzubauen.

Viel Spaß und gutes Arbeiten!

Kapitel 01 – Die Bausteine

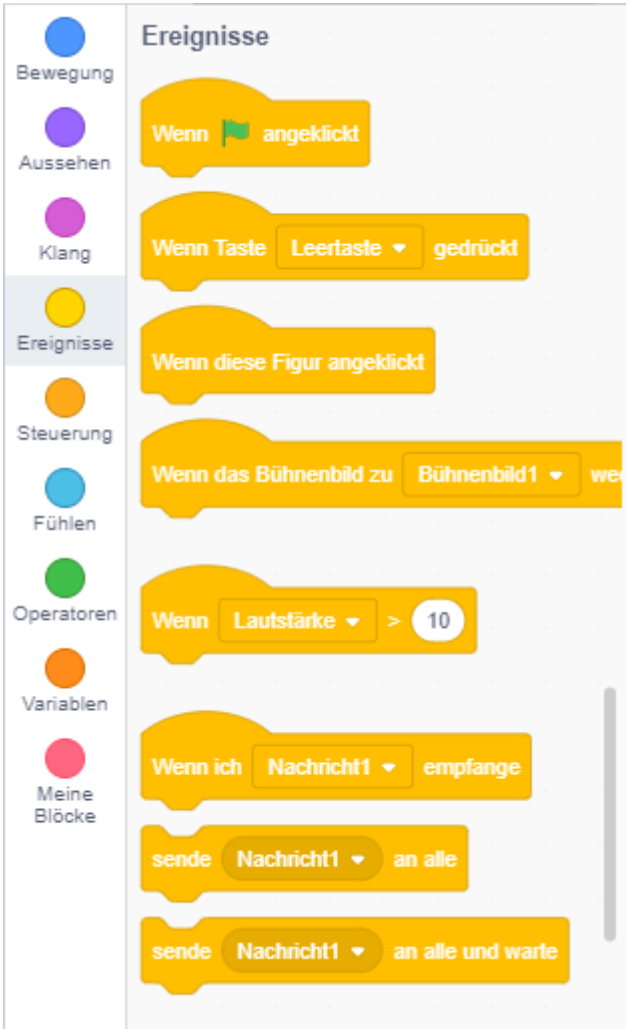
In Scratch gibt es wie schon erwähnt verschiedene Bausteine mit verschiedenen Funktionen und davon soll dieses Kapitel handeln.



Die bestimmten Bausteine sind nach Kategorien und Farben sortiert. Per Mausklick kann man zwischen diesen Kategorien hin und her wechseln, je nach dem welcher Baustein benötigt wird.

Auf den nächsten Seiten wird jede Kategorie und deren Bausteine einzeln unter die Lupe genommen.

Bausteinkategorie: Ereignisse



Übersicht:

Farbe:	Gelb
Aufgabe:	Diese Kategorie wird auch die Kategorie der Startbausteine genannt. Sie starten ein Programm. Erst mit Aufruf eines Bausteins dieser Kategorie fängt das Programm an zu laufen. Auf Startbausteine kann nicht verzichtet werden. Jedes lauffähige Programm braucht zu Beginn einen Startbaustein, ansonsten kann das Programm nicht ausgeführt werden.
Anwendbar auf:	Figuren, Bühnenbilder

Ereignis - Wenn Fahne angeklickt

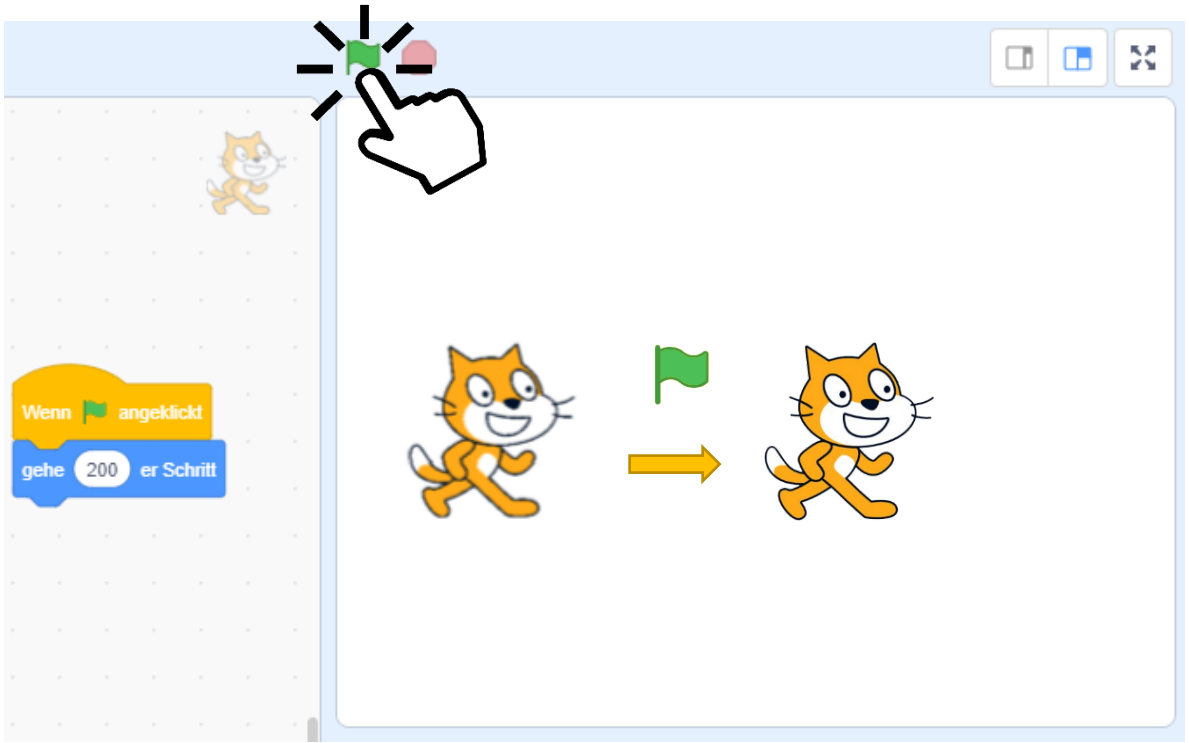
Baustein:

Erklärung:

Dieser Baustein kann nur verwendet werden, um das Spiel per Klick auf die grüne Flagge zu starten. Er ist einer der am häufigsten verwendeten Ereignisbausteine.

Beispiel:

Hier sehen wir ein einfaches Programm bei dem Scratch einen 200er-Schritt machen soll. Er macht den Schritt, sobald das Programm gestartet wird. Dies erfolgt über den Klick auf die grüne Flagge.



Ereignis - Wenn Taste gedrückt

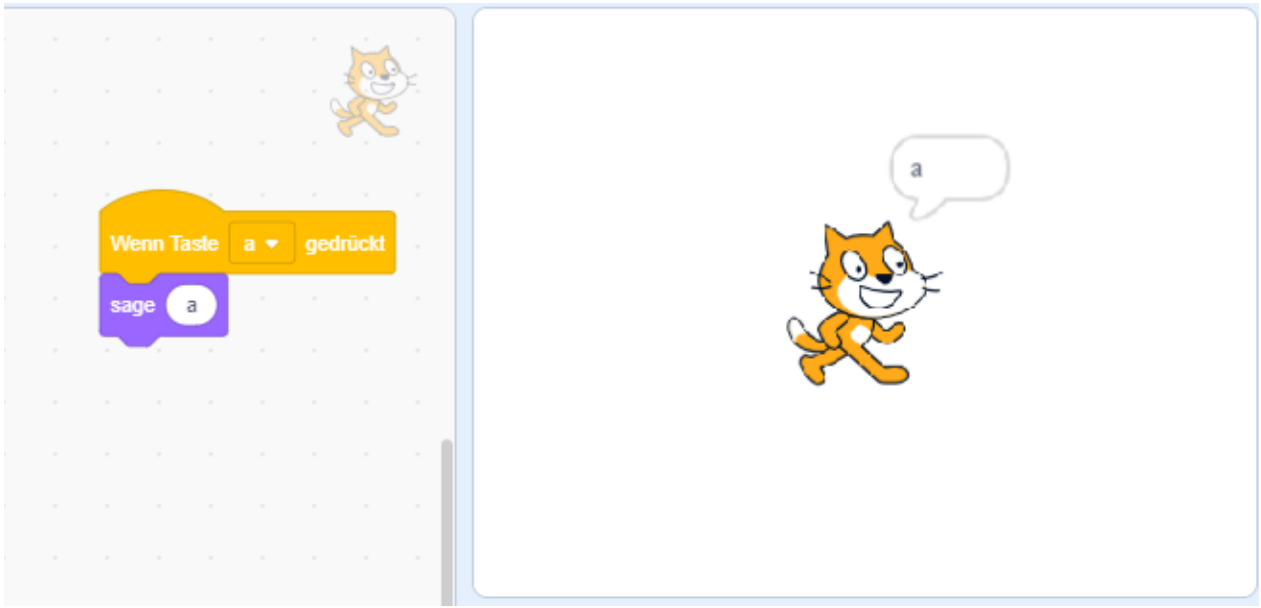
Baustein:

Erklärung:

Dieser Baustein startet das Programm mit einem Druck auf eine beliebig definierbare Taste der Tastatur. Es ist also möglich, eine Starttaste festzulegen.

Beispiel:

Der Ereignisbaustein „Wenn Taste Leertaste gedrückt“ wurde über ein einfaches Drop-down Menü auf den Buchstaben „a“ eingestellt. Drückt die Benutzerin oder der Benutzer nun die Taste „a“ auf der Tastatur, sagt Scratch „a“.



Ereignis - Wenn diese Figur angeklickt

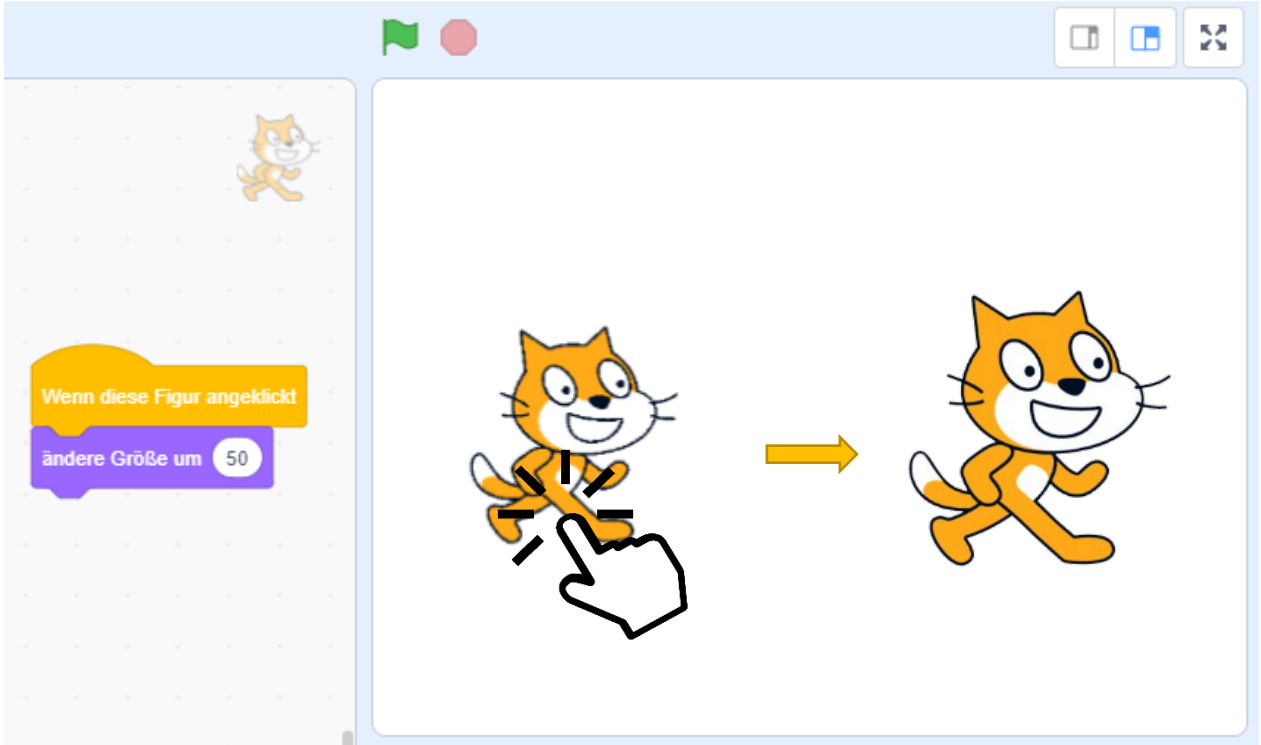
Baustein:

Erklärung:

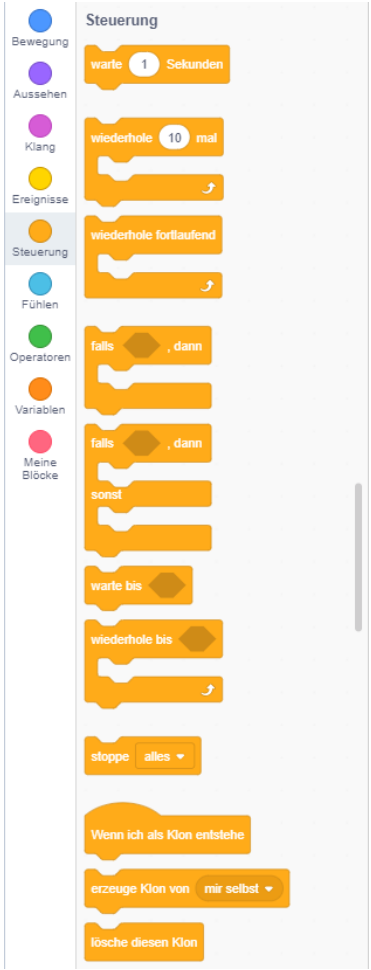
Mit diesem Baustein wird das Programm durch einen einfachen Klick auf die aktuell ausgewählte Figur gestartet. In diesem Fall ist das die Katze Scratch.

Beispiel:

Wird auf die ausgewählte Figur geklickt, soll das Programm gestartet werden. Im Folgenden Beispiel soll sich die Größe der ausgewählten Figur (Scratch, die Katze) per Mausklick um 50 Einheiten ändern.



Bausteinkategorie: Steuerung



Übersicht:

Farbe:	Orange
Aufgabe:	Diese Art von Bausteinen hat streng genommen nichts damit zu tun, die Figur zu steuern. Das wird in Scratch von der Bausteinkategorie „Bewegung“ übernommen. Es ist viel mehr die Steuerung von Abläufen und Vorgängen. Anhand der „wiederhole-Bausteine“ zum Beispiel, können lange Bausteinabfolgen wesentlich vereinfacht und verkürzt werden.
Anwendbar auf:	Figuren, Bühnenbilder

Steuerung – Warte [Zahl eintippen] Sekunden

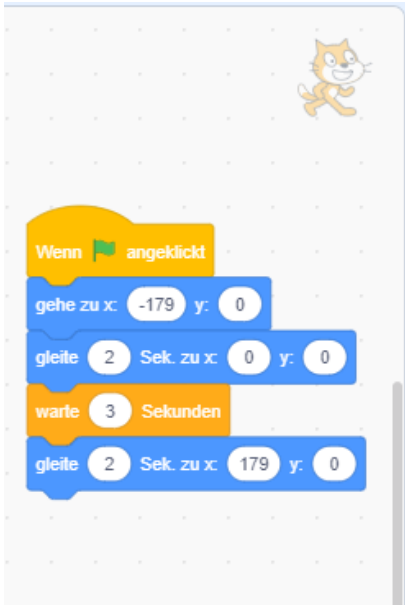
Baustein:

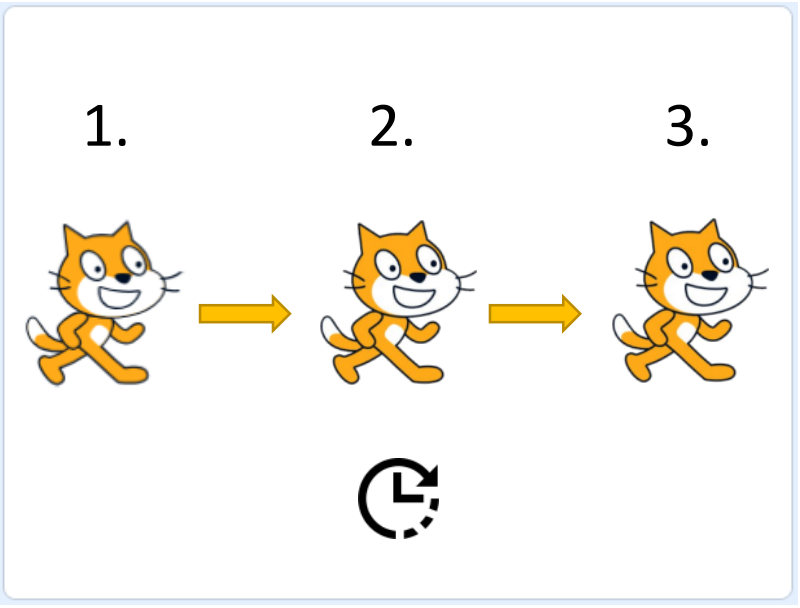
Erklärung:

Hier wird Scratch angewiesen, eine bestimmte Zeit abzuwarten, bevor mit der Abarbeitung der Befehlskette fortgefahren wird.

Beispiel:

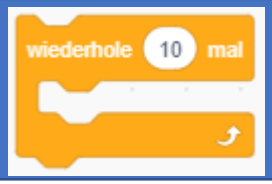
Was wir im folgenden Beispiel sehen, ist ein einfacher Spaziergang vom linken zum rechten Bildschirmrand. Im 1. Schritt gibt man Scratch eine Startposition, von der er losspazieren soll. Anschließend läuft Scratch in Richtung Position 2. Einhergehend mit dem Erreichen von Position 2 macht Scratch eine drei Sekunden dauernde Pause. Danach läuft er weiter bis zu Position 3, bei der seinen Spaziergang beendet.





Steuerung – Wiederhole [Zahl eintippen] mal

Baustein:



Erklärung:

Mithilfe dieses Bausteins kann eine Schleife erzeugt werden. Bei sich oft wiederholenden Abfolgen kann dieser Baustein eine wesentliche Erleichterung und eine deutliche Verkürzung des Codes sein.

Beispiel:

Wie auch schon in der Einleitung dieser Kategorie von Bausteinen, sehen wir hier zwei verschiedene Ansätze einer Umsetzung für ein und dasselbe Problem. Auf der linken Seite sehen wir, dass keine Steuerungsbausteine verwendet werden, auf der rechten hingegen wurde sehr wohl ein Steuerungsbaustein verwendet. Obwohl die Problemstellung hier wirklich einfach ist, sieht man trotzdem einen deutlichen Unterschied in der Länge und dem Aufwand der beiden Bausteinketten.

Steuerung – Wiederhole fortlaufend

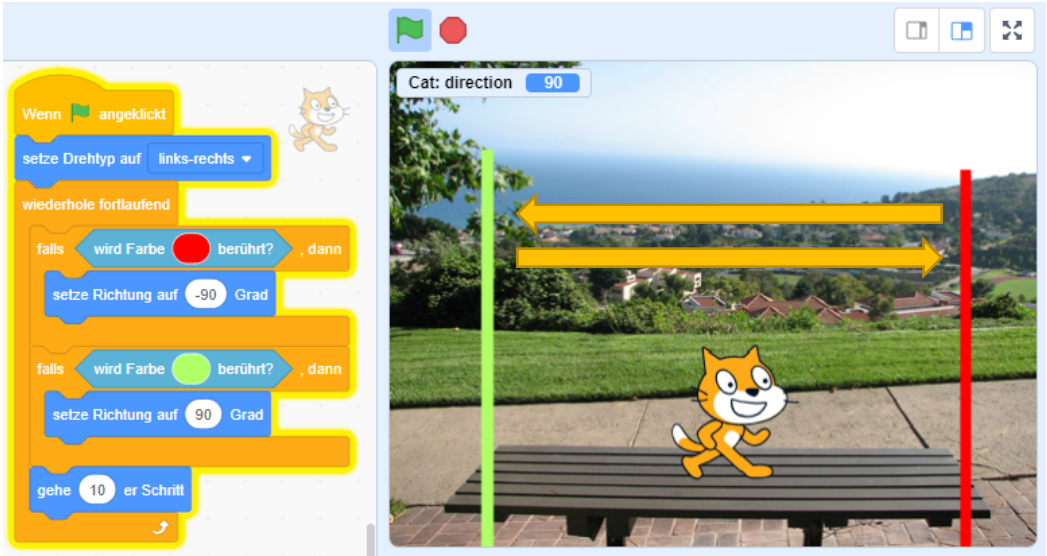
Baustein:

Erklärung:

Ähnlich wie im zuvor erklärten Baustein wird hier ebenfalls eine Schleife erzeugt. Jedoch ist es bei diesem Baustein nicht möglich, eine Anzahl der Wiederholungen festzulegen. Mit diesem Baustein wird eine Endlosschleife initiiert. Dieser wiederholt die in sich eingeschlossene Bausteinkette sequenziell von oben nach unten. Ist man unten angekommen, springt man wieder zum Anfang und die Kette von Befehlen wird erneut ausgeführt. Dies geschieht endlos oft, bis das Programm gestoppt wird.

Beispiel:

In diesem Beispiel darf Scratch nicht von der Tischplatte fallen. Um das zu gewährleisten, nehmen wir zwei verschiedenfarbige Linien zur Hilfe. Mit einem Baustein aus der Kategorie „Fühlen“ sagen wir Scratch, dass er sich umdrehen soll, wenn er eine bestimmte Farbe berührt. Wir benötigen zwei verschiedene Farben, damit er auch weiß, in welche Richtung er sich drehen muss. Je nach berührter Farbe, dreht er sich nach links oder rechts. Unabhängig davon soll Scratch unendlich viele 10er Schritte machen. Dazu wird der „wiederhole fortlaufend“ Baustein benötigt. Nun weiß Scratch immer rechtzeitig, wann er sich umdrehen muss und kann nicht mehr von der Tischplatte fallen. Scratch bewegt sich nun in einer Endlosen Schleife zwischen den beiden Linien.



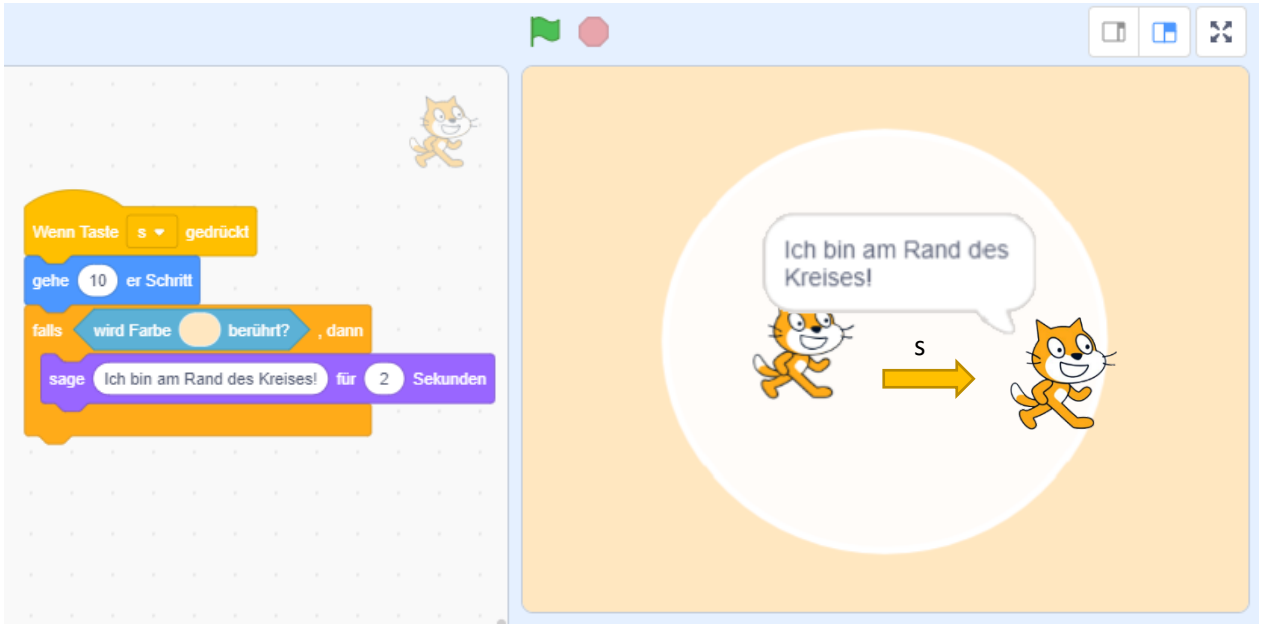
Baustein:

Erklärung:

Hier kann eine Bedingung festgelegt werden, wann eine bestimmte Handlung ausgeführt werden soll. Sollte diese Bedingung eintreffen, wird der zwischenliegende Code ausgeführt. In der textbasierten Programmierung ist dieser Baustein mit der IF-Abfrage zu vergleichen. In das sechseckige Feld ist ein Baustein mit der gleichen Form einzufügen, um die Bedingung festzulegen. Bausteine mit dieser Form sind beispielsweise unter der Bausteinkategorie „Fühlen“ oder „Operatoren“ zu finden.

Beispiel:

Das hier aufgeführte Beispiel veranschaulicht auf ganz einfache Weise die Funktionsweise des „falls [Bedingung], dann“ Bausteins. Scratch befindet sich zu Beginn innerhalb des Kreises. Durch das wiederholte Drücken der „s“ Taste bewegt er sich dann in 10er Schritten in Richtung Rand des Kreises. Hat Scratch den Rand dann schlussendlich erreicht, meldet er sich mit einem vorher manuell eingetippten Statusbericht zu Wort.



Steuerung – Falls [Bedingung], dann // sonst

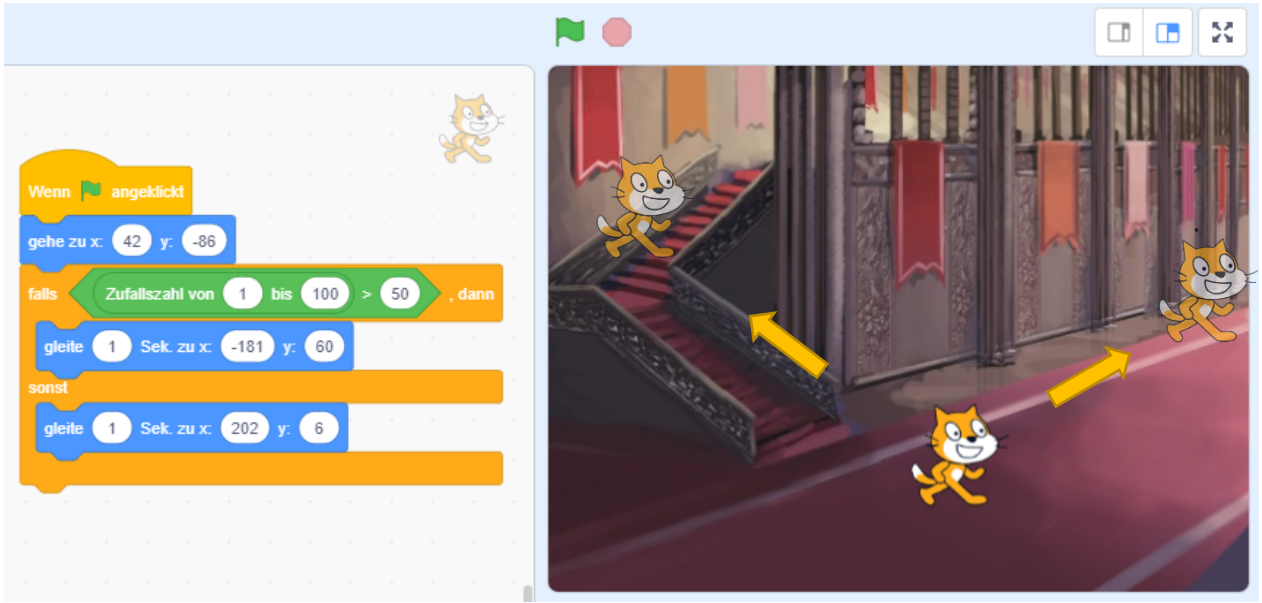
Baustein:

Erklärung:

Dieser Baustein ist dem vorher erklärten IF – Abfrage Baustein sehr ähnlich, mit dem Unterschied, dass dieser Baustein eine Art Plan B bietet. Sollte die festgelegte Bedingung NICHT gelten, so wird die Option „sonst“ ausgeführt. In der textbasierten Programmierung ist dieser Baustein mit einer IF-ELSE-Abfrage zu vergleichen.

Beispiel:

Scratch steht vor einer Abzweigung und weiß nicht, in welche Richtung er gehen soll. Er lässt den Zufall durch Ermittlung einer zufälligen Zahl zwischen 1 und 100 entscheiden. Ist die Zahl kleiner als 50, geht er nach links, ist die Zahl jedoch größer 50, so geht er nach rechts.



Steuerung – Warte bis [Bedingung]

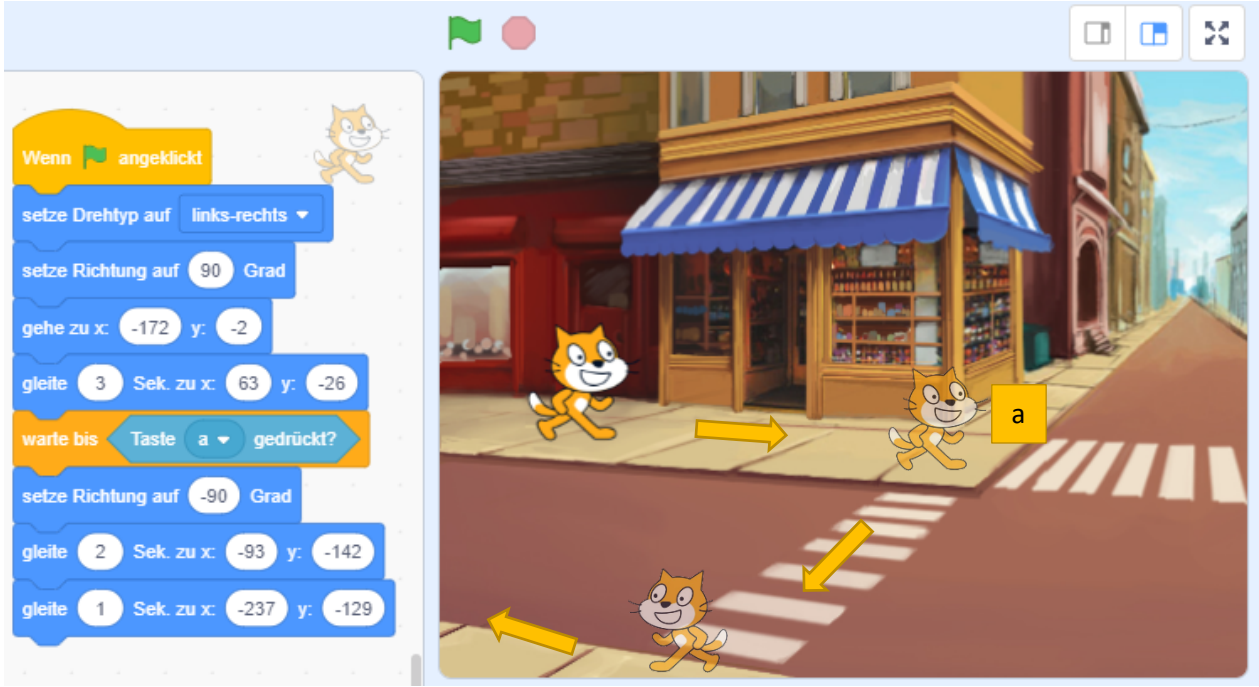
Baustein:

Erklärung:

Anders als bei dem Warte-Baustein, den wir bereits kennengelernt haben, wird hier nicht eine bestimmte Zeit festgelegt, die gewertet werden soll, sondern es wird auf den Eintritt eines bestimmten Ereignisses gewartet. Auch hier kann das Ereignis über die beiden Bausteinkategorien „Fühlen“ und „Operatoren“ definiert werden.

Beispiel:

Es geht um das Wohlergehen von Scratch! Er soll unverletzt die Straße queren. Zunächst einmal geht Scratch den Gehsteig entlang bis zum Zebrastreifen. Dann soll Scratch warten, bis kein Auto kommt. Drückt man dann die Taste „a“ auf der Tastatur, geht Scratch schnell über die Straße und läuft weiter entlang dem Gehsteig.



Steuerung – Wiederhole bis [Bedingung]

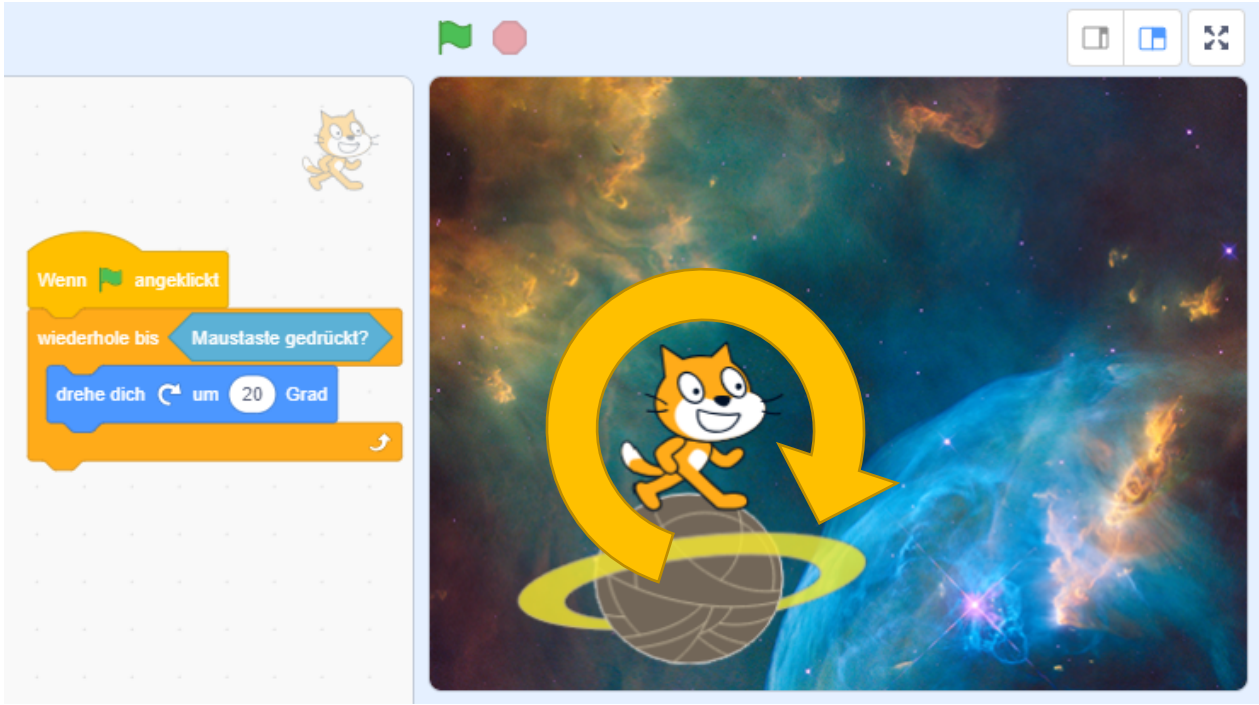
Baustein:

Erklärung:

Mit der Ausführung des Codes wird bei diesem Baustein gewartet, bis eine bestimmte Bedingung eintritt. Dieser Baustein ist ebenfalls eine Schleife. Das Äquivalent in der textbasierten Programmierung ist wäre die sogenannte WHILE-Schleife.

Beispiel:

Scratch ist irgendwo im Weltall in Rotation geraten und kann nur mit unserer Hilfe gestoppt werden, indem wir die Maus klicken und er mit beiden Beinen wieder festen Boden unter den Füßen hat.



Baustein:

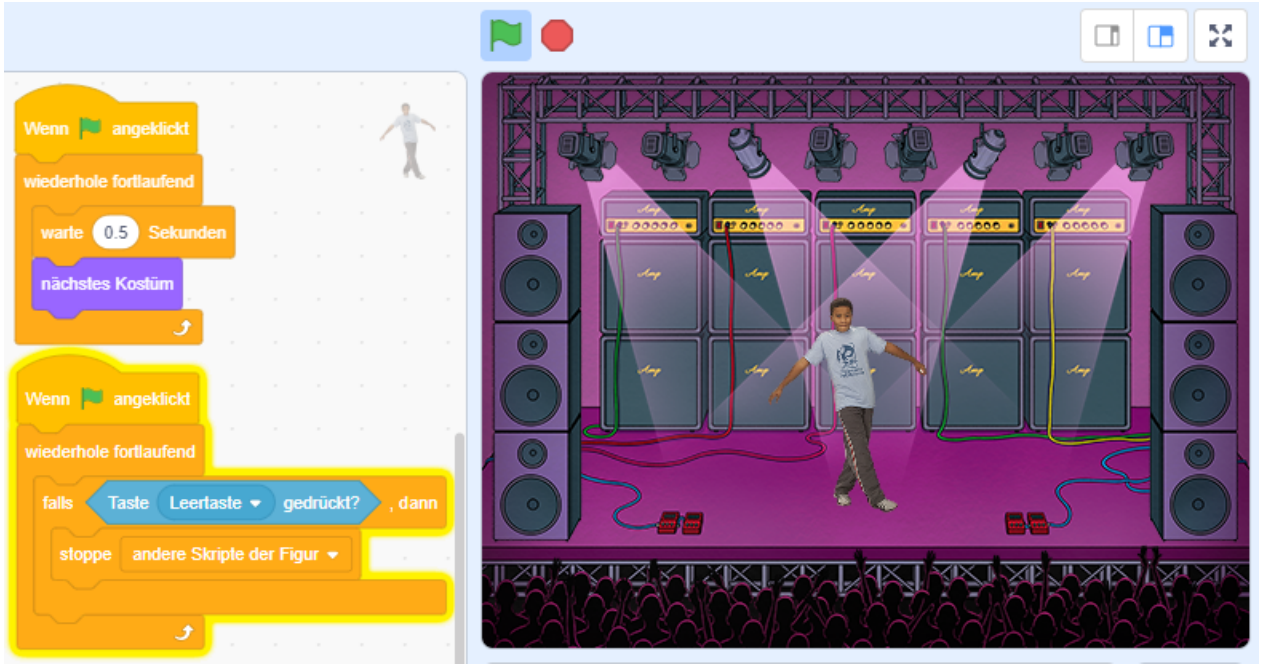
Erklärung:

Dieser Baustein dient zum Stoppen eines Skripts. Der Baustein hat 3 Funktionen, die über ein Dropdown Menü ausgewählt werden können.

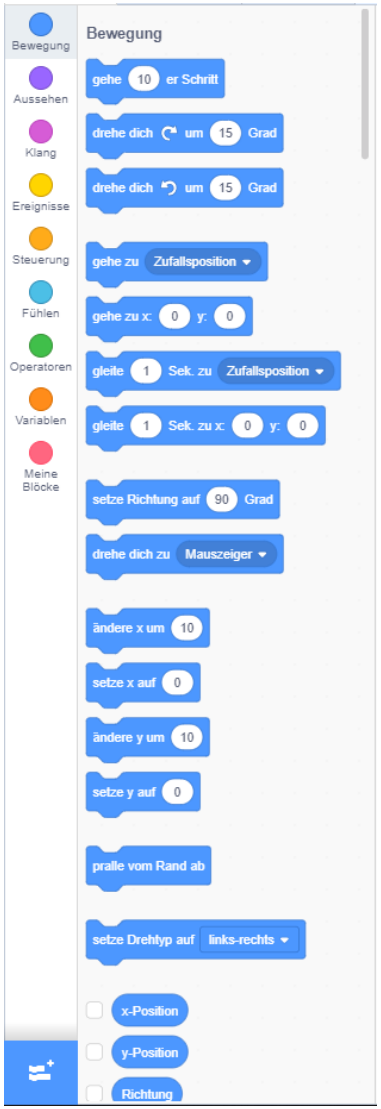
- **Stoppe alles:**
  - Es werden alle laufenden Skripte gestoppt
- **Stoppe dieses Skript:**
  - Es wird nur jenes Skript gestoppt, in dem der Baustein verbaut wurde.
- **Stoppe andere Skripte der Figur:**
  - Alle Skripte, mit Ausnahme von dem Skript, das den Baustein verbaut hat, werden gestoppt.

Beispiel:

In diesem Beispiel wird die „stoppe alle andere Skripte der Figur“ Variation des Bausteins verwendet. Der Junge tanzt fortlaufend und hört erst auf, wenn der „stoppe“ Baustein aktiviert wird.



Bausteinkategorie: Bewegung



Übersicht:

Farbe:	Blau
Aufgabe:	Einige Bewegungsbausteine haben wir bereits in vorherigen Bausteinerklärungen kennengelernt. Es lässt sich nur schwer vermeiden, Bewegungsbausteine einzubauen. Sie dienen der Fortbewegung, der Drehung oder der sonstigen Umsetzung von Aktivitäten einer Figur.
Anwendbar auf:	Figuren

Bewegung – gehe [Zahl eintippen] er Schritt

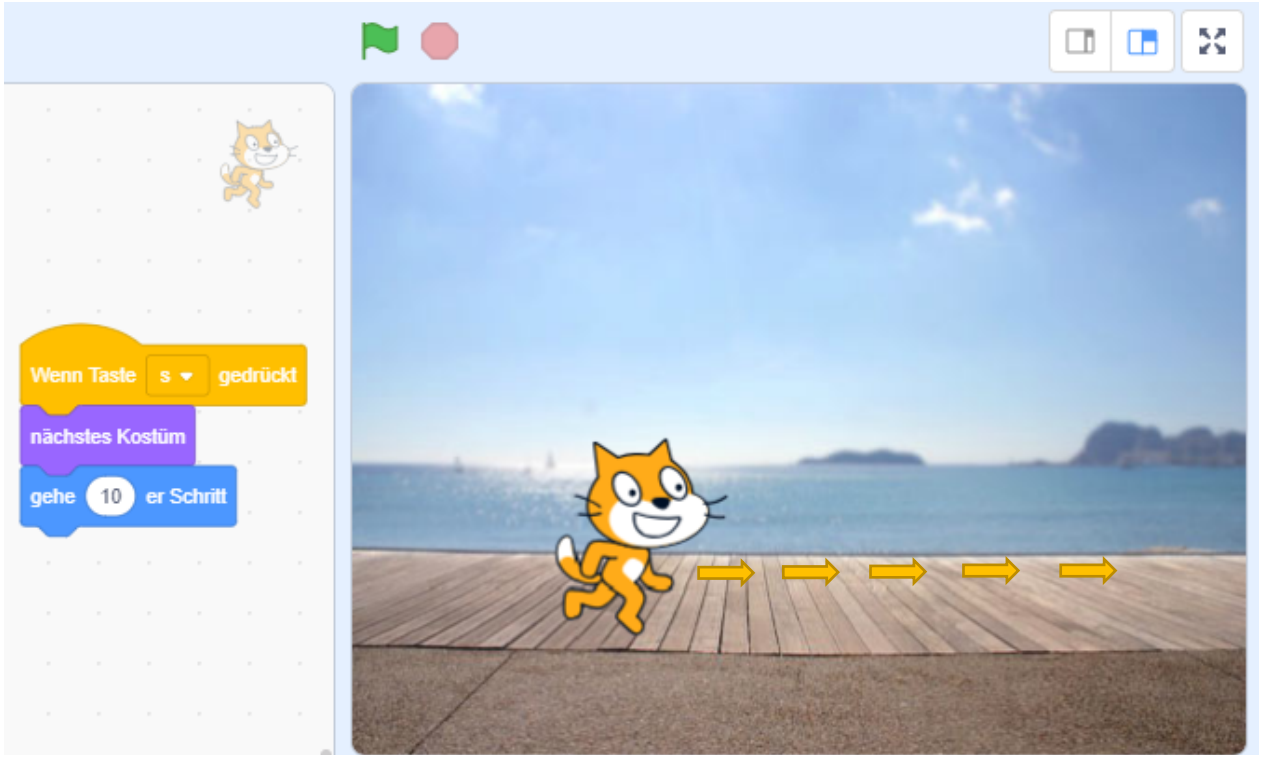
Baustein:

Erklärung:

Scratch wird angewiesen, einen beliebig großen oder kleinen Schritt in Blickrichtung zu gehen. Die Zahl innerhalb des weißen Feldes kann frei gewählt werden. Wird diese Zahl negativ gesetzt, so geht Scratch also rückwärts.

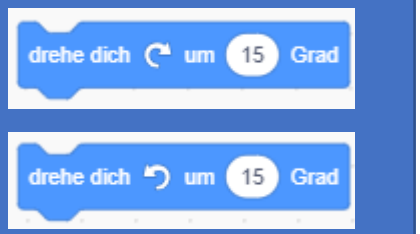
Beispiel:

Scratch macht einen Spaziergang. Mit einem Druck auf die Taste „s“ auf der Tastatur, macht Scratch einen Schritt. Um das Ganze noch ein bisschen zu untermalen, wechselt er mit jedem Schritt sein Kostüm.



Bewegung – drehe dich links / rechts um [Zahl eintippen] Grad

Baustein:



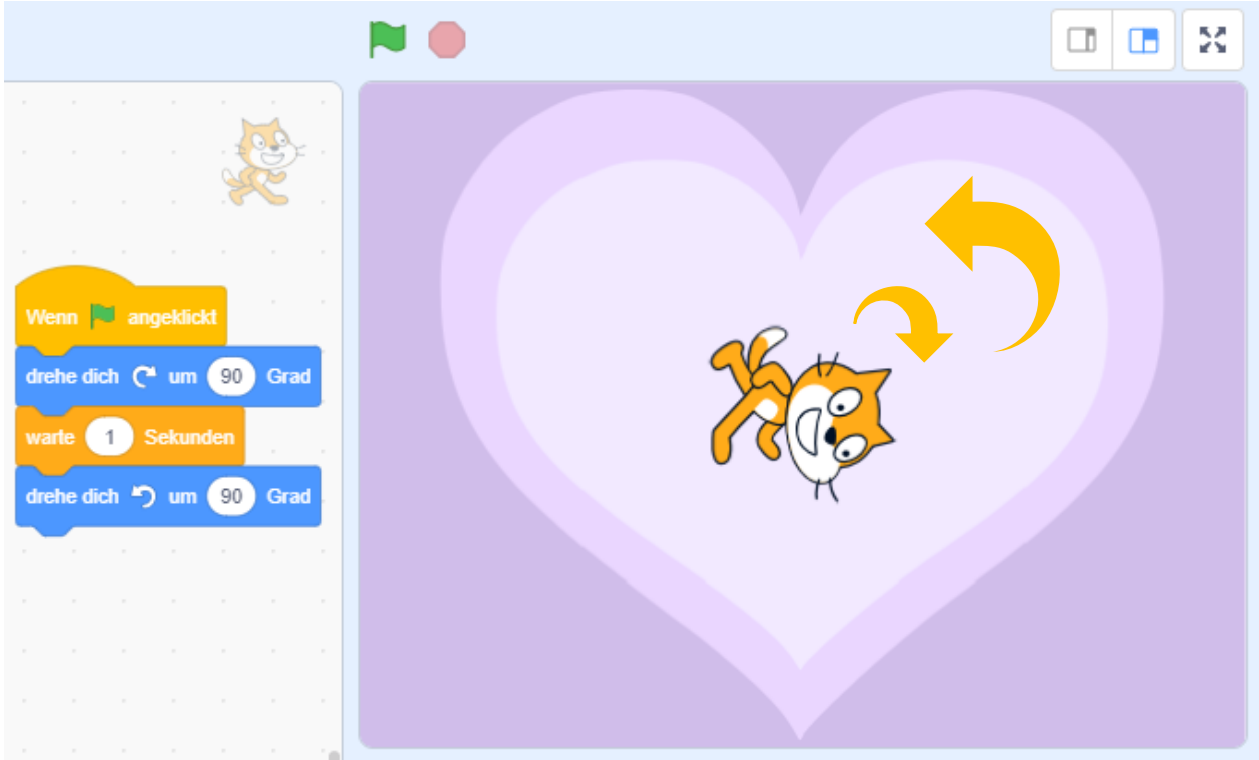
Erklärung:

Scratch soll sich drehen. Um wie viel Grad er sich drehen soll, hängt davon ab, welche Zahl man in das weiße Feld eingibt.

Es gibt einen Baustein, der die Figur nach links drehen lässt und es gibt einen Baustein, der die Figur nach rechts drehen lässt. Welchen Baustein man verwendet, ist von der Aufgabenstellung, für die er benötigt wird, abhängig zu machen.

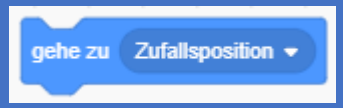
Beispiel:

Scratch soll sich mit dem Klick auf die grüne Fahne um 90 Grad nach rechts drehen und nach einer Sekunde soll er sich wieder nach links in seine ursprüngliche Position drehen.



Bewegung – gehe zu Zufallsposition / Mauszeiger

Baustein:



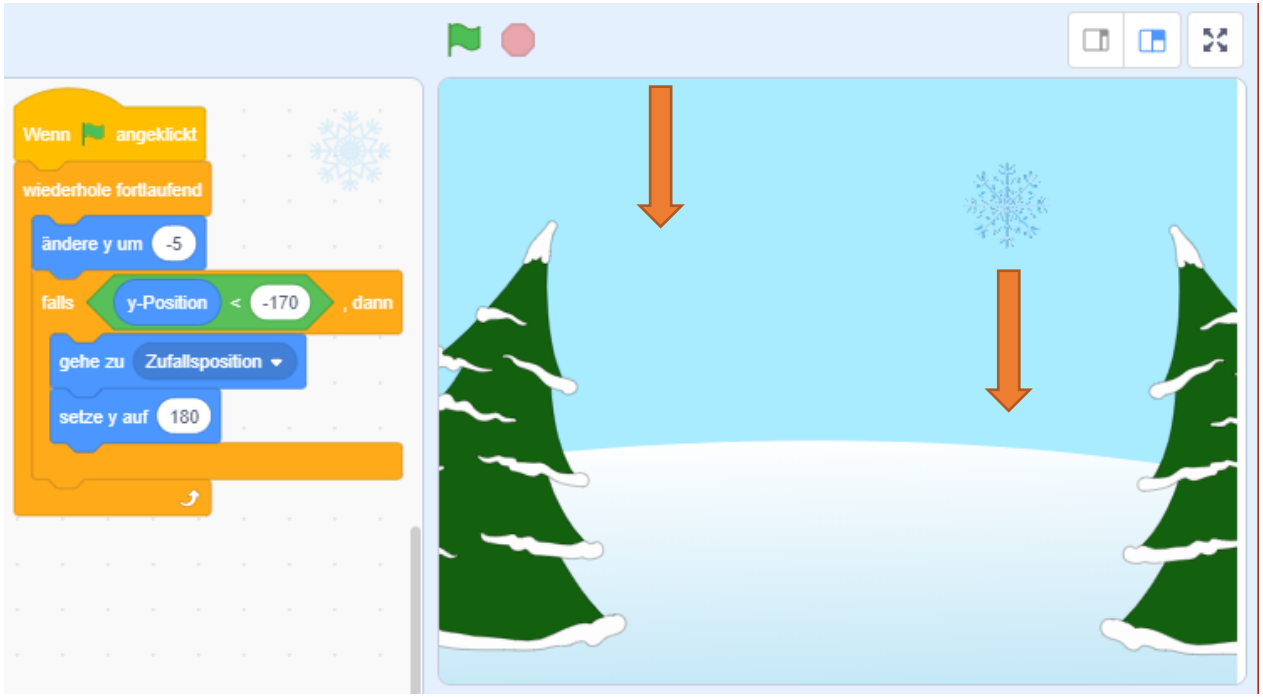
Erklärung:

Scratch wird es durch diesen Baustein ermöglicht, an eine zufällig ausgewählte Position zu springen. Es werden zwei Koordinaten errechnet, eine x-Koordinate und eine y-Koordinate. Scratch hüpft dann an diesen errechneten Punkt.

Über das Dropdown Menü kann Scratch auch befohlen werden, zum Mauszeiger zu hüpfen, das beispielsweise bei einem interaktiven Spiel oder etwas Ähnlichem sehr praktisch sein kann.

Beispiel:

Hier wird der „gehe zu Zufallsposition“ Baustein für einen Schnee-Effekt verwendet. Sobald die Schneeflocke den Boden erreicht, taucht sie wieder an einer zufälligen Position am oberen Rand des Bildschirms auf, aber nie an der gleichen Stelle mithilfe des Bausteins. Ohne den „setze y auf 180“ Baustein könnte die Schneeflocke auch in die Mitte des Bildschirms auftauchen.



Bewegung – gehe zu [x-Koordinate] // [y-Koordinate]

Baustein:

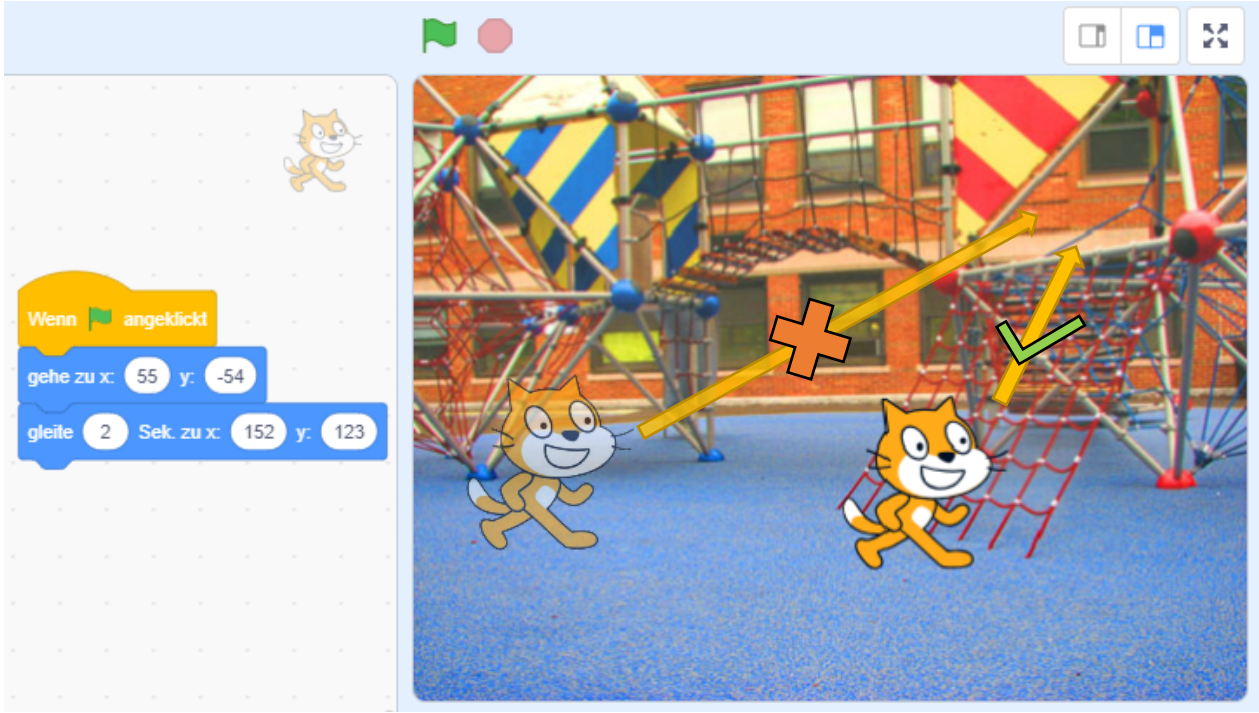
Erklärung:

Fast genau gleich wie der „gehe zu Zufallsposition / Mauszeiger“ Baustein, funktioniert dieser Baustein auch. Jedoch werden die Koordinaten hier nicht zufällig ausgewählt, sie können manuell in die beiden dafür vorgesehenen Felder eingegeben werden.

Das bedeutet, man kann Scratch beispielsweise eine Start- bzw. Endposition zuweisen.

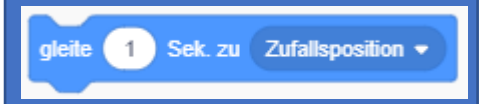
Beispiel:

Scratch muss das rote Kletternetz hinaufklettern. Das kann er logischerweise nur, wenn er direkt davorsteht. Andernfalls müsste er fliegen können, das ist hier leider nicht der Fall ist. Steht er dann vor dem Netz, ist es für Scratch als Katze ein Leichtes nach oben zu klettern. Wie wir sehen, definiert der „gehe zu“ Baustein die optimale Startposition.



Bewegung – gleite [Zahl eintippen] Sek. zu Zufallsposition / Mauszeiger

Baustein:

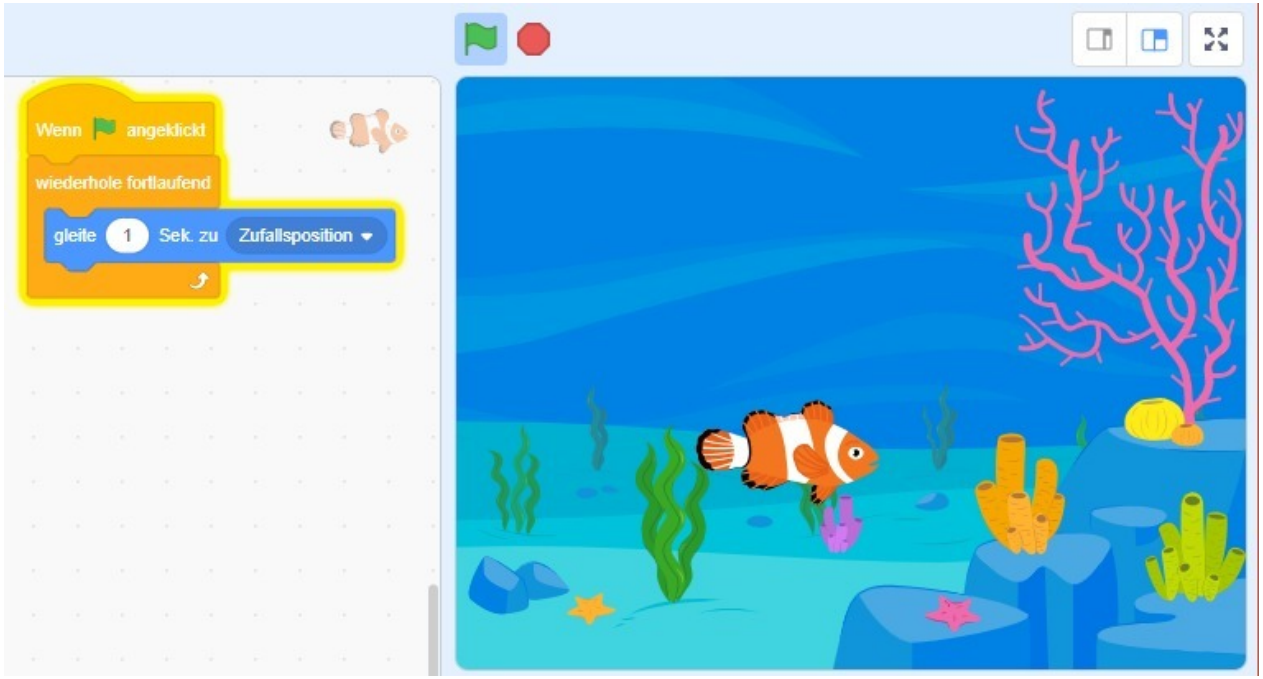


Erklärung:

Der „gleite zu“ Baustein ist dem „gehe zu“ Baustein sehr ähnlich, mit dem Unterschied, dass Scratch nicht an eine Position hüpft, sondern sich in einer manuell festlegbaren Zeit darauf zu bewegt. Der Baustein erzeugt eine flüssige Bewegung, daher wird er oftmals zur Animation von Aktivitäten wie: laufen, klettern, schwimmen, fahren, etc. verwendet. Dieser Baustein lässt Scratch zu einer zufälligen Position oder zum Mauszeiger gleiten.

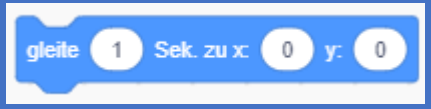
Beispiel:

Einem Fisch ist sehr langweilig. Er beschließt ein bisschen herumzuschwimmen, um die Landschaft zu erkunden. Dazu verwendet er den „gleite zu Zufallsposition“ Baustein. Er bewegt sich also zufallsgesteuert über den Bildschirm und schaut sich die schöne Unterwasserwelt an.



Bewegung – gleite [Zahl eintippen] Sek. zu [x-Koordinate] // [y-Koordinate]

Baustein:



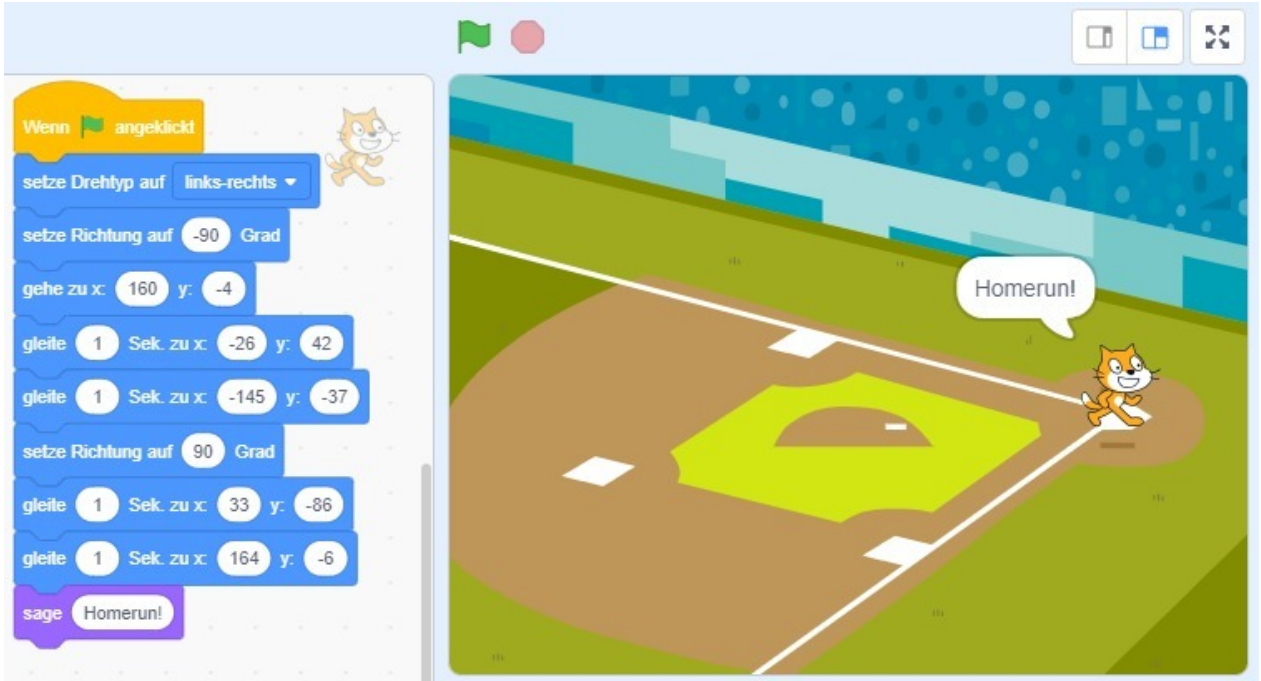
Erklärung:

Der „gleite zu“ Baustein ist dem „gehe zu“ Baustein sehr ähnlich, mit dem Unterschied, dass Scratch nicht an eine Position hüpft, sondern sich in einer manuell festlegbaren Zeit darauf zu bewegt. Der Baustein erzeugt eine flüssige Bewegung, daher wird er oftmals zur Animation von Aktivitäten wie: laufen, klettern, schwimmen, fahren, etc. verwendet. Man kann manuell eine x-Koordinate und eine y-Koordinate festlegen, zu der Scratch gleiten soll.

Beispiel:

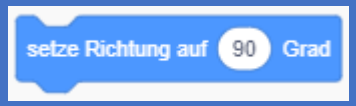
Scratch hat einen Homerun geschlagen! Nun heißt es rennen – Scratch gleitet über die weißen Felder, bis er schlussendlich „HOMERUN!“ brüllt. Man sieht deutlich, wie der „gleite zu“ Baustein hier zum Einsatz kommt.

Ändert man die Anzahl der Sekunden beim „gleite zu“ Baustein, braucht Scratch entweder länger oder kürzer, um seinen Homerun zu rennen.



Bewegung – setze Richtung auf [Zahl eintippen] Grad

Baustein:

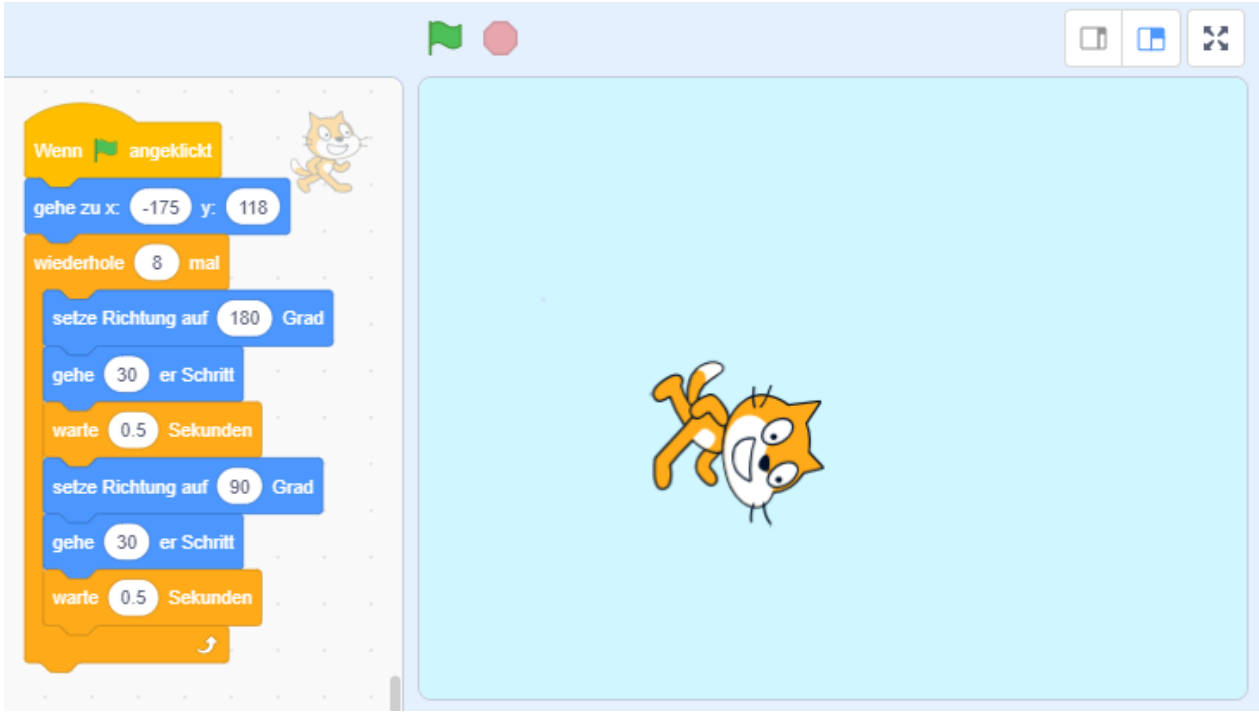


Erklärung:

Ein Baustein, der es Scratch ermöglicht, sich zu drehen. Standardmäßig dreht sich Scratch rundherum und seine Neigung wird über die Maßeinheit Grad [°] definiert. Scratch kann sich mit Hilfe des „setze Drehtyp“ Bausteins auch um andere Achsen drehen bzw. kann damit auch seine Drehung unterbunden werden. Mehr dazu ist auf der Erklärungsseite des „setze Drehtyp“ Bausteins zu finden. Beim Starten der Scratch Anwendung ist Scratch standardmäßig auf 90° ausgerichtet.

Beispiel:

Scratch läuft in diesem Beispiel schräg über den Bildschirm in Richtung rechts unten. Hier sind mehrere Bausteine, die wir bereits kennengelernt haben, zum Einsatz gekommen. Wir machen uns die Eigenschaft des „gehe Schritt“ Bausteins zu Nutze, Scratch bewegt sich immer einen Schritt in Blickrichtung. Die Richtung wird jedoch ständig geändert und so kommt es, dass er schräg über den Bildschirm läuft.



Bewegung – drehe dich zu Mauszeiger // anderer Figur

Baustein:

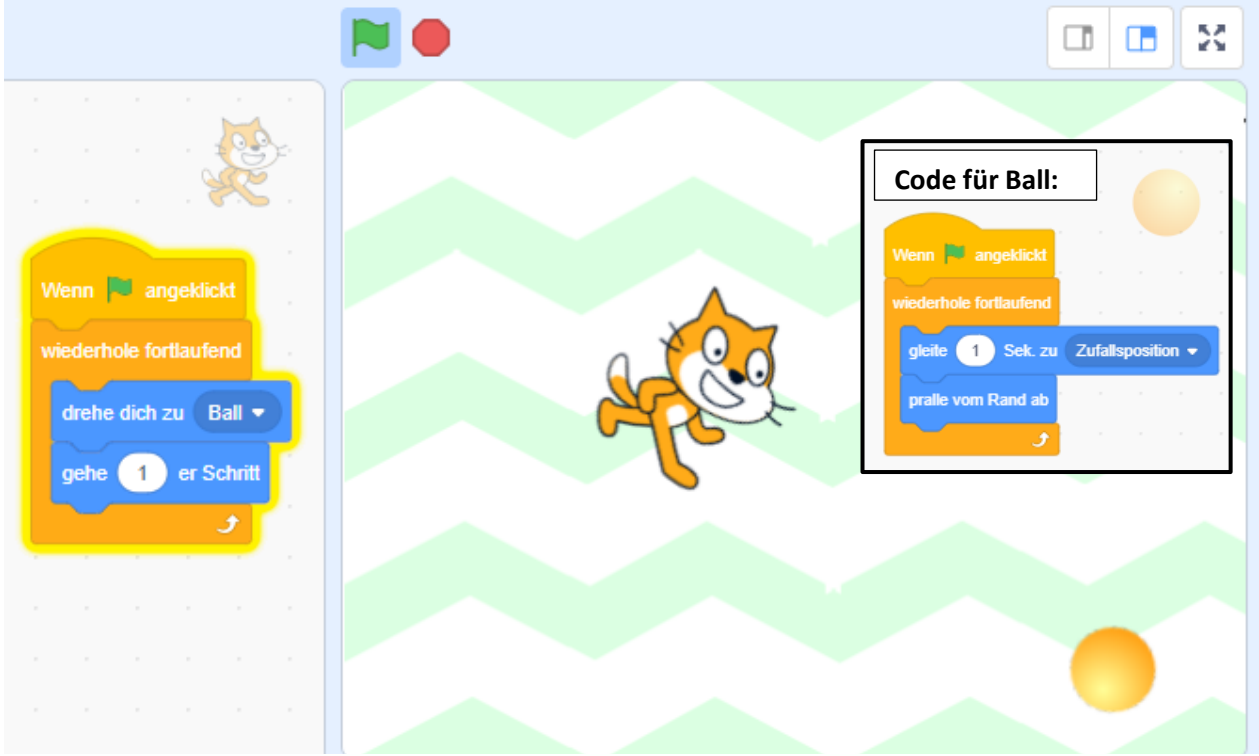
Erklärung:

Dieser Baustein ist dazu gedacht, Scratch in Richtung eines Objekts schauen zu lassen. Das kann entweder der eigene Mauszeiger oder eine andere Figur sein.

Wie wir im Beispiel sehen können, kann dieser Baustein sehr gut für das Verfolgen eines Objektes eingesetzt werden.

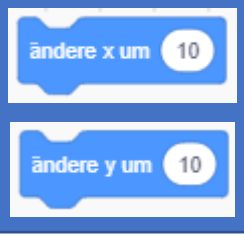
Beispiel:

Verspielt wie Scratch eben ist, jagt er dem orangen Ball hinterher, der sich fortlaufend zu einer neuen Zufallsposition begibt. Scratch prüft mit der Schleife, die wir im Programmierbereich sehen ständig, wo sich der Ball aktuell befindet und geht dann einen kleinen Schritt in diese Richtung. Da der Ball aber ständig seine Position ändert, ist es ein ewiges Katz‘ und Maus-Spiel.



Bewegung – ändere x-Koordinate / y-Koordinate um [Zahl eintippen]

Baustein:



Erklärung:

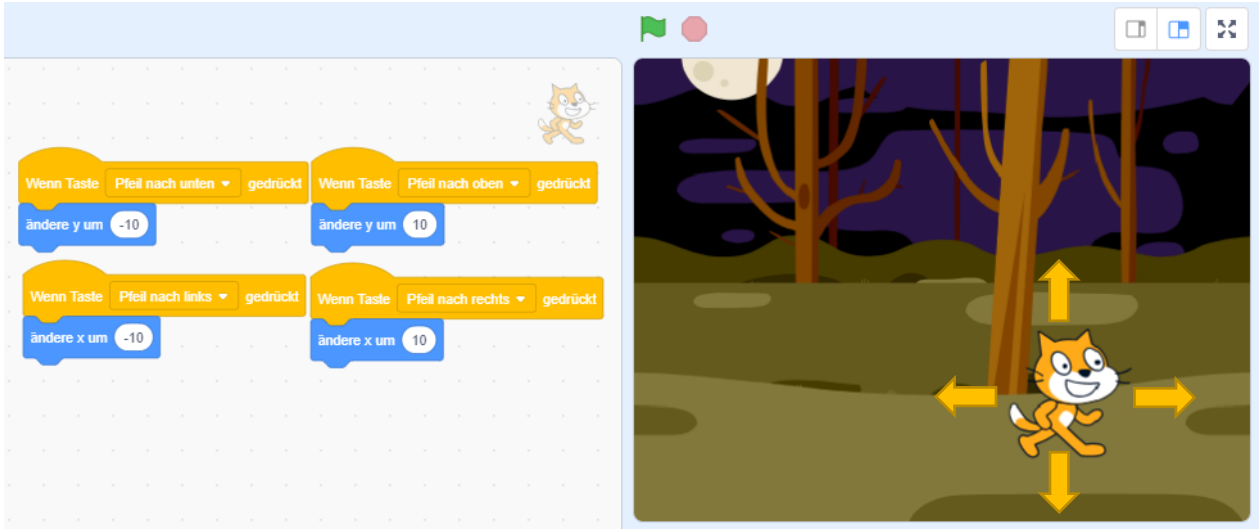
Anders als der „gehe Schritt“ Baustein geht Scratch hier nicht einen Schritt in Blickrichtung, sondern er orientiert sich am Koordinatensystem und geht entweder in x- oder y-Richtung.

Diese Bausteine sind beispielsweise sehr hilfreich zum Programmieren von manuellen Steuerungen über die Pfeil- oder anderen Tasten auf der Tastatur.

Die Werte können auch negativ gewählt werden. (siehe Beispiel)

Beispiel:

Scratch kann so ganz leicht in alle Richtungen bewegt werden. Abhängig davon, welche der vier Pfeiltasten man drückt, geht Scratch in die entsprechende Richtung.



Bewegung – setze x-Koordinate / y-Koordinate auf [Zahl eintippen]

Baustein:

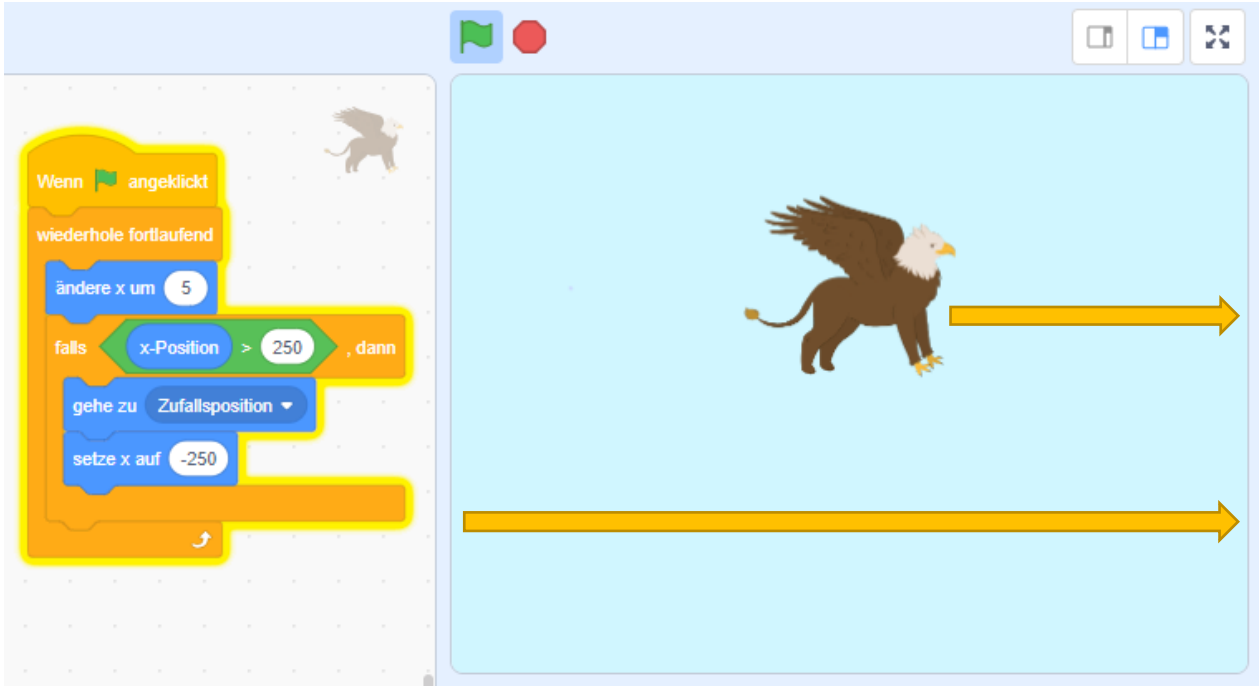
Erklärung:

Zwei Bausteine die jeweils die x-Koordinate oder die y-Koordinate auf einen bestimmten Wert setzen.

Der „gehe zu“ Baustein legt beide Werte fest. Was nun aber tun, wenn beispielsweise die x-Koordinate fix definiert sein muss, die y-Koordinate jedoch zufällig gewählt werden soll? Genau, man verwendet den „setze x“ Baustein, und das Problem ist gelöst.

Beispiel:

Der Greif, den wir hier sehen, fliegt durch die Luft. Er fliegt vom linken bis zum rechten Bildschirmrand. Ist er dort angekommen, hüpft er zurück zum Anfang, wählt sich einen neun Startpunkt aus und fliegt wieder los. Da Greife mehrere 1000 Jahre alt werden, geht das lange so.



Baustein:

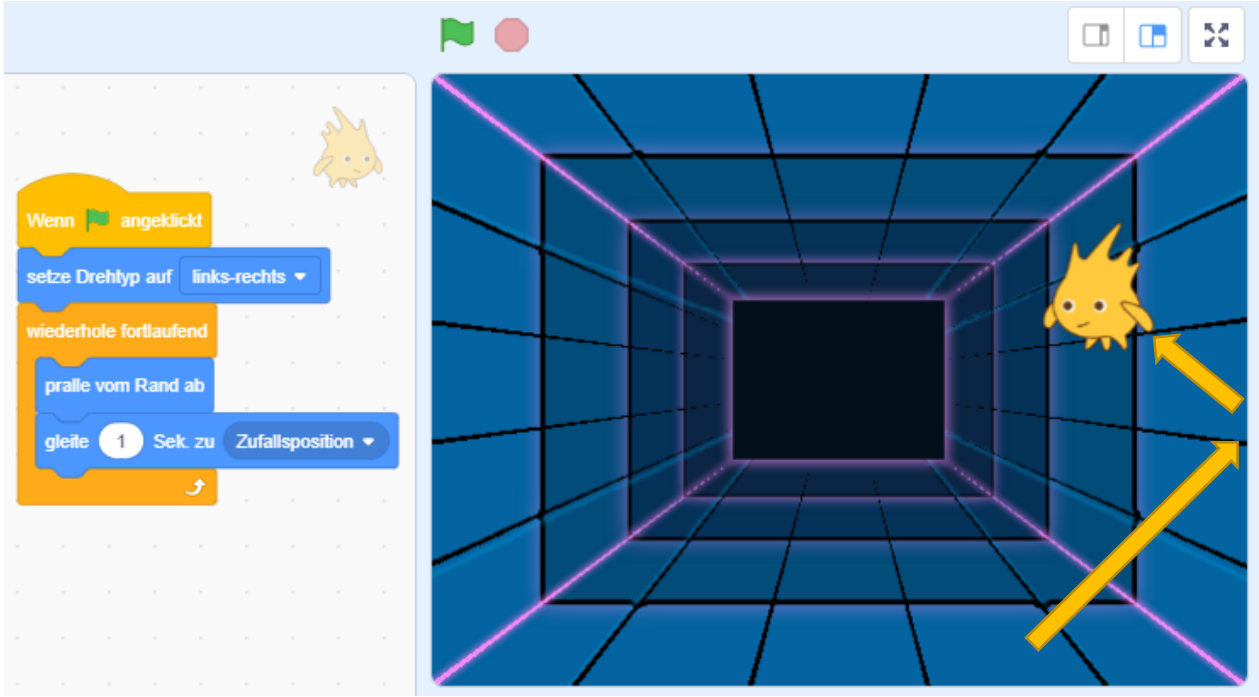
Erklärung:

Ein Baustein, der es einem Objekt nicht erlaubt, sich über den Rand hinaus zu bewegen. Stattdessen prallt dieses vom Rand ab und dreht sich um.

Je nachdem welche Rotationsachse ausgewählt wurde, dreht sich das Objekt in die entsprechende Richtung. Die Rotationsachse kann über den „setze Drehtyp“ Baustein ausgewählt werden.

Beispiel:

Unser oranger Freund Gobo befindet sich irgendwo weit weg von der Erde, in einer fremden Galaxie. Er schwebt wahllos in einem Neon Tunnel umher. Zu seinem Glück prallt er immer wieder vom Rand ab und kann so nicht in den endlosen Weiten des Universums verloren gehen.



Baustein:

Erklärung:

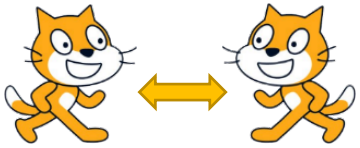
Der „setze Drehtyp“ Baustein, legt die Rotationsachse fest. Man kann zwischen drei Optionen auswählen:

- **links-rechts**
  Die Figur rotiert um seine eigene Y-Achse.
- **nicht drehen**
  Die Figur lässt sich nicht mehr drehen
- **rundherum**
  Die Figur rotiert um seine Z-Achse.

Beispiel:

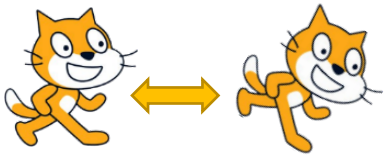
In den folgenden Bildern sehen sie die drei möglichen Arten von Rotation die Scratch anbietet:

links-rechts

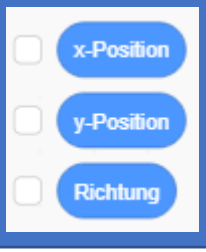


nicht drehen

rundherum



Baustein:



Erklärung:

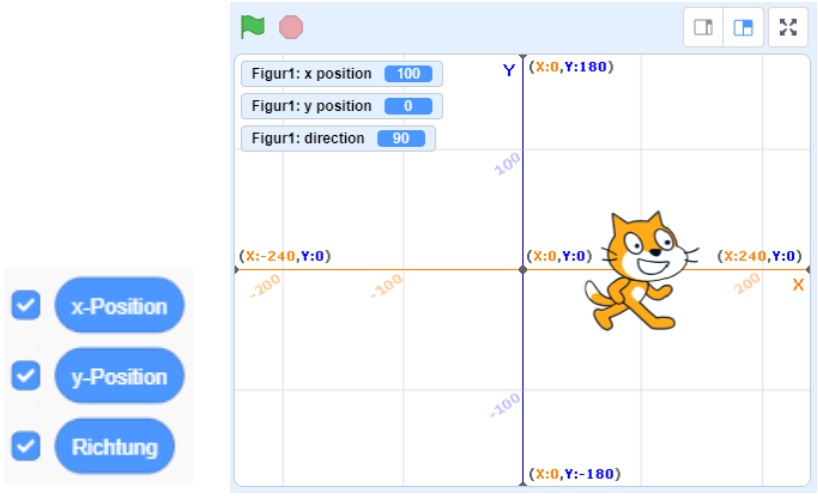
Variablen kann man sich wie eine Art Behälter, der Werte speichert vorstellen. Diese Werte können dann aufgerufen und verwendet werden. In diesem konkreten Fall wird die x-Position, die y-Position und die Richtung gespeichert. Wie im Beispiel zu sehen ist kann man diese Werte anzeigen lassen ober direkt in das Programm einbinden.

Hilfestellungen:

Variablen

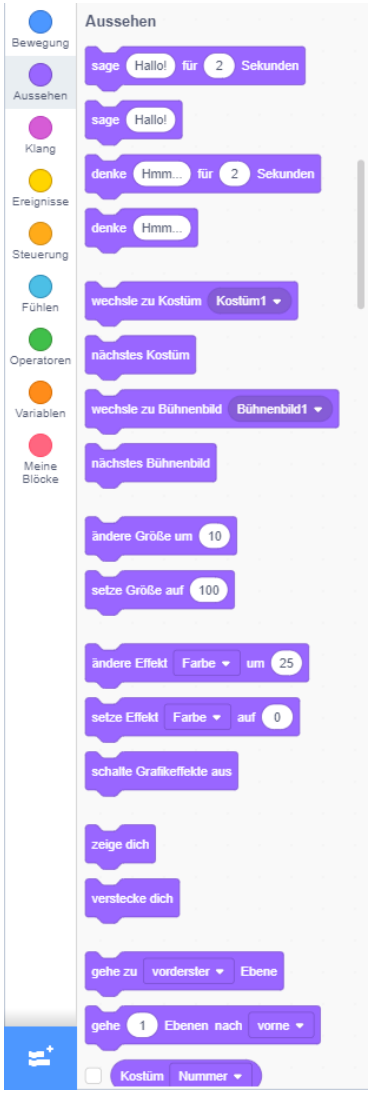
Beispiel:

Aktiviert man in der Bausteinsektion die Häkchen neben den Variablen, wie es unten vorgemacht ist, dann werden diese auf der Leinwand im linken oberen Eck angezeigt. Dort werden dann in Echtzeit die jeweiligen Daten ausgegeben.



Die Variablen können auch in andere Bausteine eingesetzt werden, um den aktuellen Wert direkt in das Programm miteinzubeziehen:

Bausteinkategorie: Aussehen

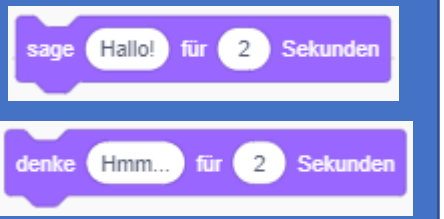


Übersicht:

Farbe:	Violett
Aufgabe:	Wenn in Scratch etwas angezeigt werden muss, verwendet man dazu die Bausteine dieser Kategorie. Egal, ob es um das Wechseln von Kostümen, um das Verstecken oder Anzeigen von Figuren, das Wechseln von Ebenen oder um das simple Anzeigen einer Sprechblase geht.
Anwendbar auf:	Figuren, Bühnenbilder
Sonstige Hinweise:	Nicht jeder Baustein dieser Kategorie ist auf Bühnenbilder und Figuren anwendbar. Ein Bühnenbild hat zum Beispiel keinen Zugriff auf die „sage“ oder „denke“ Bausteine.

Aussehen – sage / denke [Text eintippen] für [Zahl eintippen] Sekunden

Baustein:

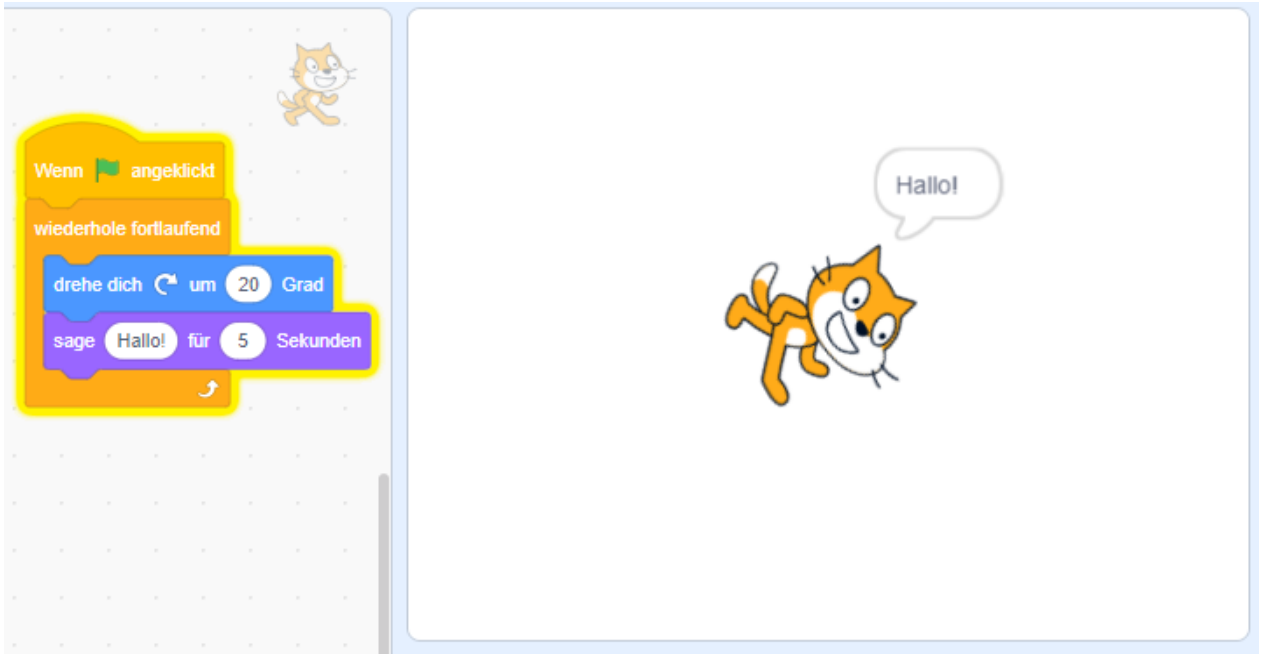


Erklärung:

Scratch meldet sich zu Wort. Dieser Baustein lässt eine Sprechblase erscheinen, die für eine bestimmte Zeit einen gewünschten Text ausgibt. Dabei ist zu beachten, dass Scratch, während er spricht, den aktuellen Codeblock für die angegebene Dauer unterbricht.

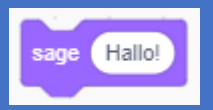
Beispiel:

Im folgenden Beispiel dreht sich Scratch fortlaufend um 20 Grad. Dieses Drehen im Kreis wird vom „sage“ Baustein unterbrochen, da Scratch pro Codeblock immer nur einen Baustein zur selben Zeit bearbeiten kann. Das Beispiel des nächsten Bausteins veranschaulicht den Unterschied nochmals.



Aussehen – sage [Text eintippen] // denke [Text eintippen]

Baustein:

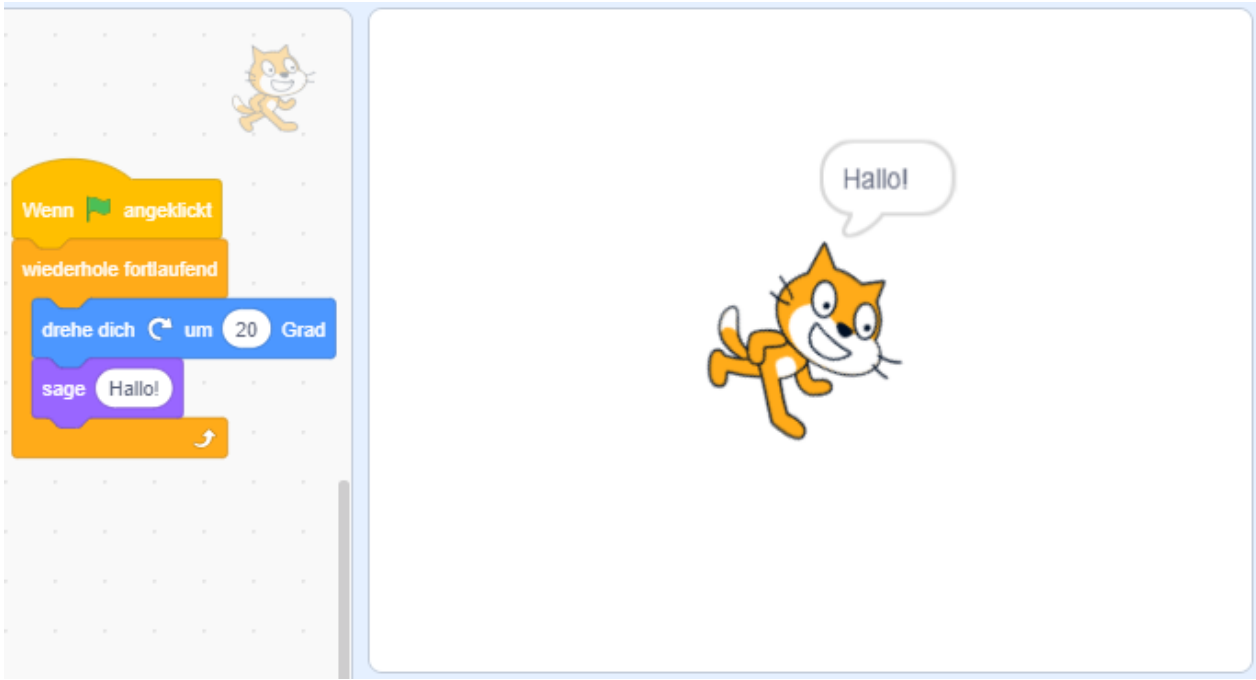


Erklärung:

Dieser Baustein ist ähnlich wie der vorherige Baustein, der auf der Seite 35 zu sehen ist. Der Unterschied dabei ist, dass dieser Baustein die Sprech-/ Gedankenblase für eine unbestimmte Zeit erscheinen lässt. Diese Sprechblase bleibt dann solange erhalten, bis sie entweder von einer anderen Sprechblase ersetzt wird oder das Programm per Klick auf das rote Stoppzeichen, rechts neben der grünen Fahne, gestoppt wird. Im Gegensatz zum vorherigen Baustein unterbricht dieser das Skript nicht, sondern alles läuft weiter.

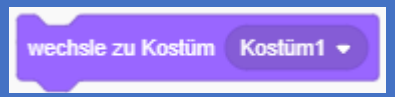
Beispiel:

Um den Unterschied zwischen den beiden Bausteinen „sage [Text eintippen] für [Zahl eintippen] Sekunden“ und „sage [Text eintippen]“ nochmals zu veranschaulichen, ist bei diesem Beispiel nur dieser einzelne Baustein ausgetauscht worden. Und siehe da: Die Katze dreht richtig durch! Probiere es einfach selbst aus!



Aussehen – wechsle zu Kostüm [beliebiges Kostüm auswählen]

Baustein:



Erklärung:

Nicht nur Scratch, nahezu jede Figur, kann zwischen einem oder mehreren Kostümen hin und her wechseln. Kostüme haben verschiedene Funktionen. Entweder kann man damit eine Animation erzeugen oder zwischen verschiedenen Figuren hin und her wechseln. Mit dem oben angeführten Baustein kann ein bestimmtes Kostüm ausgewählt werden.

Hilfestellungen:

Kostüm Funktion

Beispiel:

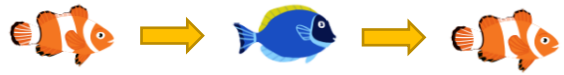
Dieses Beispiel geht nun genauer auf die zwei, in der Erklärung bereits angesprochenen, möglichen Funktionen der Kostümfunktion genauer ein. Ob Wechsel des Aussehens oder Erstellen einer kleinen Animation ist dabei abhängig von der ausgewählten Figur. Der Fisch beispielsweise wechselt sein Aussehen, die Katze dagegen sieht aus als würde sie laufen.

Möglichkeit 1:

Erstellen einer kleinen Animation.

Möglichkeit 2:

Auswahl aus einer Reihe von verschiedenen Figuren.



Aussehen – nächstes Kostüm

Baustein:

Erklärung:

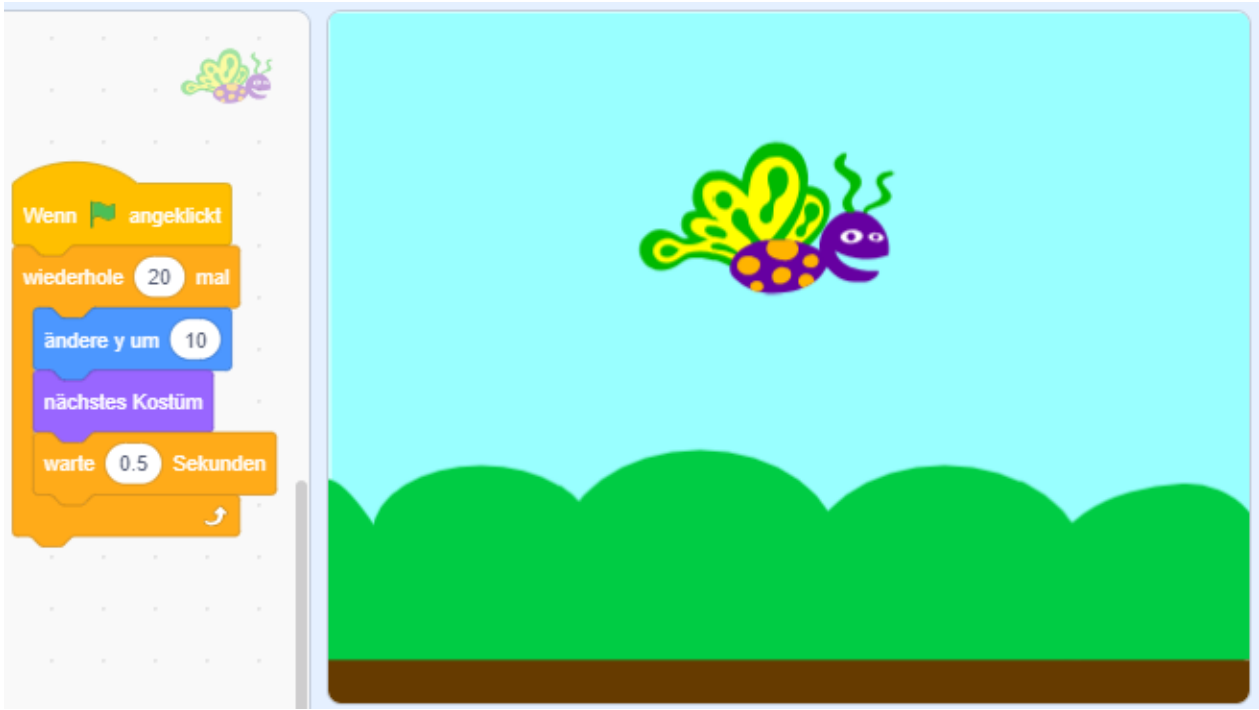
Mit diesem Baustein wird das nächste Kostüm in der Reihenfolge (siehe Hilfestellung) ausgewählt und in weiterer Folge ausgegeben. Dieser Baustein kommt zum Einsatz, wenn nicht genau definiert sein muss, welches Kostüm als nächstes aufgerufen wird oder es ohnehin nur zwei verschiedene Kostüme gibt und man das jeweils andere braucht.

Hilfestellungen:

Kostüm Funktion

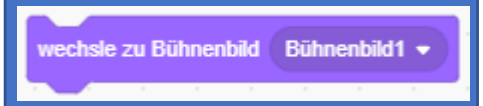
Beispiel:

Wie in der Erklärung bereits erwähnt, tritt hier genau dieser Fall ein. Die Figur „Butterfly 2“ hat nur 2 Kostüme. Mit diesen beiden Kostümen kann eine Animation erstellt werden, die den Schmetterling mit seinen Flügeln schlagen lässt.



Aussehen – wechsle zu Bühnenbild [Bühnenbild auswählen]

Baustein:



Erklärung:

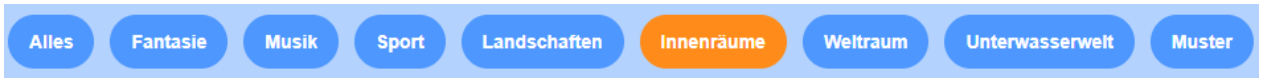
Es ist in Scratch möglich, mehrere Bühnenbilder (siehe Hilfestellung) zur gleichen Zeit geöffnet zu haben. Es wird zwar immer nur eines angezeigt, jedoch kann man zwischen mehreren geöffneten Bühnenbildern hin und her wechseln. Man kann beispielsweise eine Tag-Nacht Animation erzeugen oder den Anschein erwecken, dass eine Figur den Raum innerhalb eines Gebäudes wechselt.

Hilfestellungen:

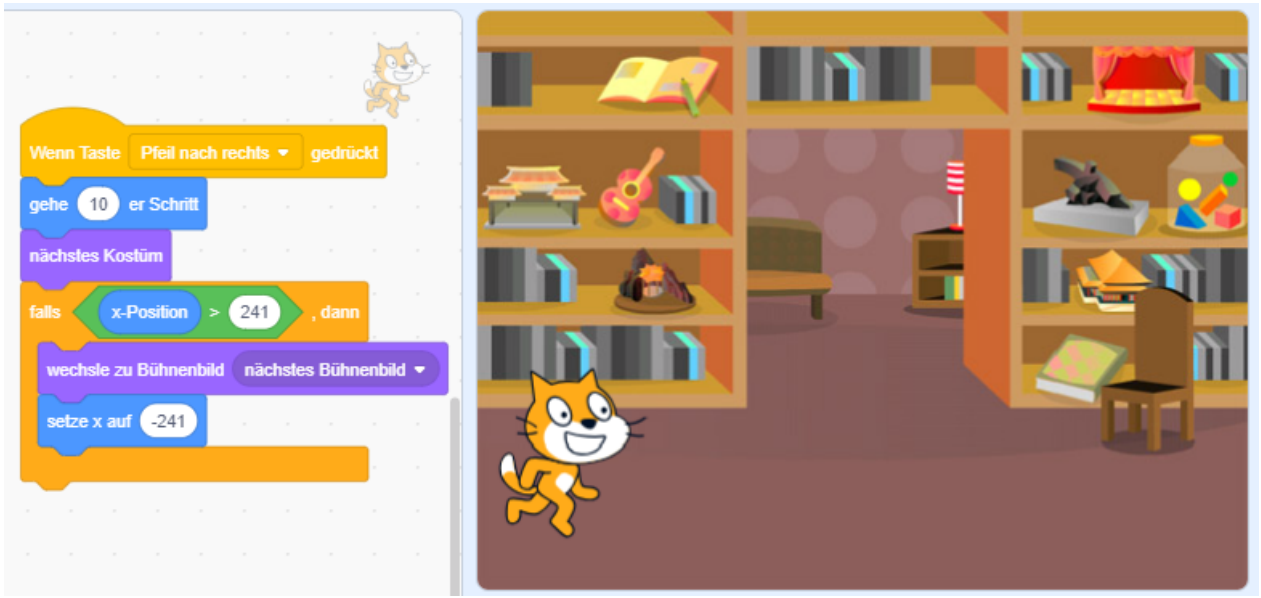
Bühnenbilder

Beispiel:

Scratch ist nun in seinem Haus! Öffne verschiedene Bühnenbilder aus der Kategorie „Innenräume“!



Mit der unten abgebildeten Steuerung, kann Scratch problemlos in den einzelnen Räumlichkeiten seines Anwesens umherlaufen. (Achtung: In diesem Beispiel kann Scratch ausschließlich nach rechts laufen!)



Baustein:

Erklärung:

Kommt dieser Baustein zum Einsatz, so wird automatisch, das nachfolgende Bühnenbild ausgewählt und aufgerufen.

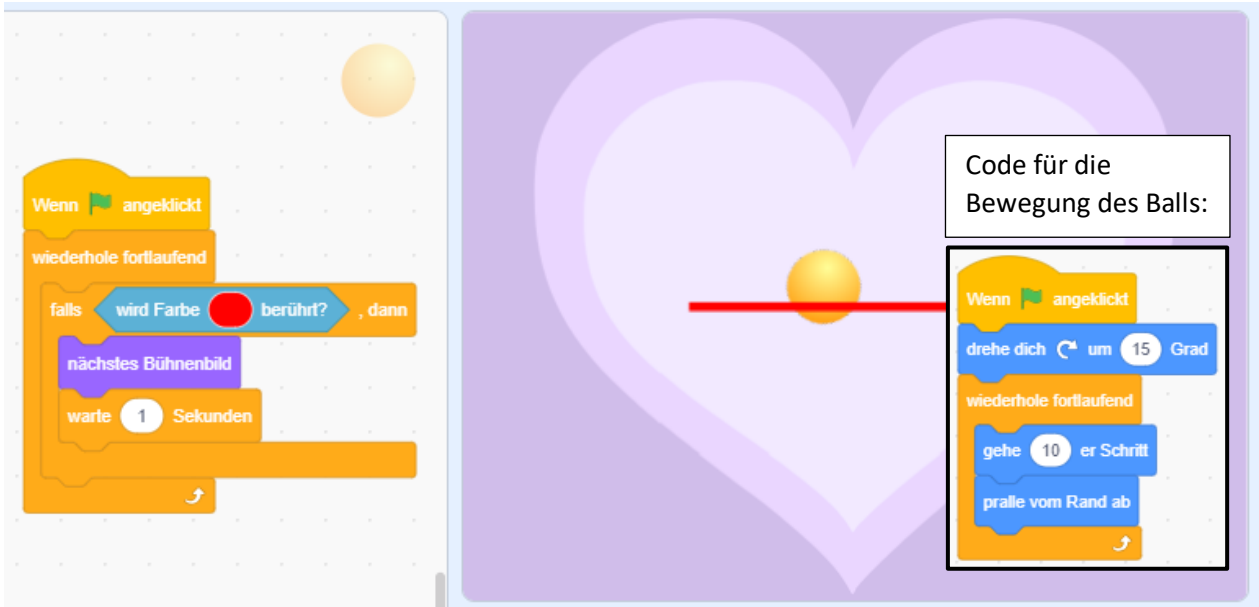
Im Dropdown Menü des zuvor erklärten Bausteins kann man ebenfalls die Funktion „nächstes Bühnenbild“ auswählen. Von der Funktion her kann der auf der aktuellen Folie erklärte Baustein nichts Neues, er spart jedoch ein wenig Zeit.

Hilfestellungen:

Bühnenbilder

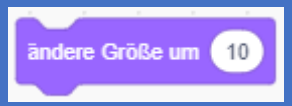
Beispiel:

Der Ball fliegt unermüdlich durch die Gegend. Jedes Mal, wenn er die rote Linie berührt, wechselt das Bühnenbild. Die Anweisung könnte so lauten: Füge dazu den Ball und die rote Linie als Figuren hinzu. Wähle eine unbestimmte Anzahl an beliebigen Bühnenbildern aus und los geht’s! (Beide im Bild zu sehenden Codeblöcke sind für die Figur „Ball“ zu programmieren.)



Aussehen – ändere Größe um [Zahl eintippen]

Baustein:



Erklärung:

Dieser Baustein ändert die Größe der aktuell ausgewählten Figur um einen beliebigen Wert. Eine positive Zahl macht die Figur größer, eine negative verkleinert sie. Manche Figuren in Scratch sind für gewisse Anwendungsfälle einfach zu groß. Es ist daher erforderlich, diese zu verkleinern.

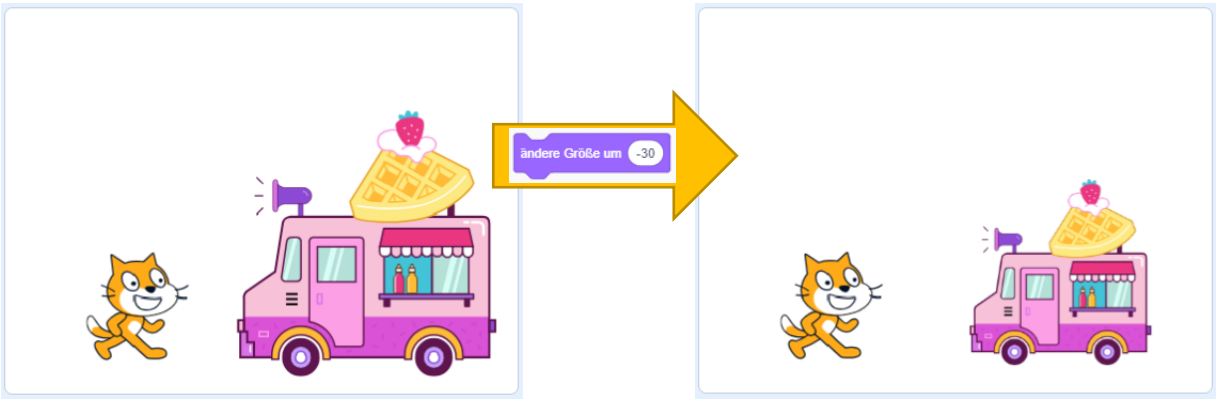
Im Beispiel wird das ebenfalls nochmals veranschaulicht.

Beispiel:

Im linken Bild ist ganz klar ersichtlich, dass die Figur „Foodtruck“ fast ein Viertel der gesamten Bühne in Anspruch nimmt. Will man diesen jetzt umherfahren lassen, ist dies mit seiner Standardgröße so gut wie nicht möglich. Daher muss man den Foodtruck etwas kleiner machen. Dies kann auf verschiedene Weisen passieren.

Direkt unter der Bühne findet man dieses Menü. Hier kann im Feld „Größe“ die Größe einer Figur angepasst werden.

Eine andere Möglichkeit die Größe einer Figur anzupassen, ist der Baustein „ändere Größe um [Zahl eintippen]“. Jedoch kann mit dem Baustein nicht die absolute Größe angegeben werden, sondern nur die Änderung der aktuellen Größe.



Aussehen – setze Größe auf [Zahl eintippen]

Baustein:

Erklärung:

Dieser Baustein setzt die absolute Größe einer Figur auf den angegebenen Wert.

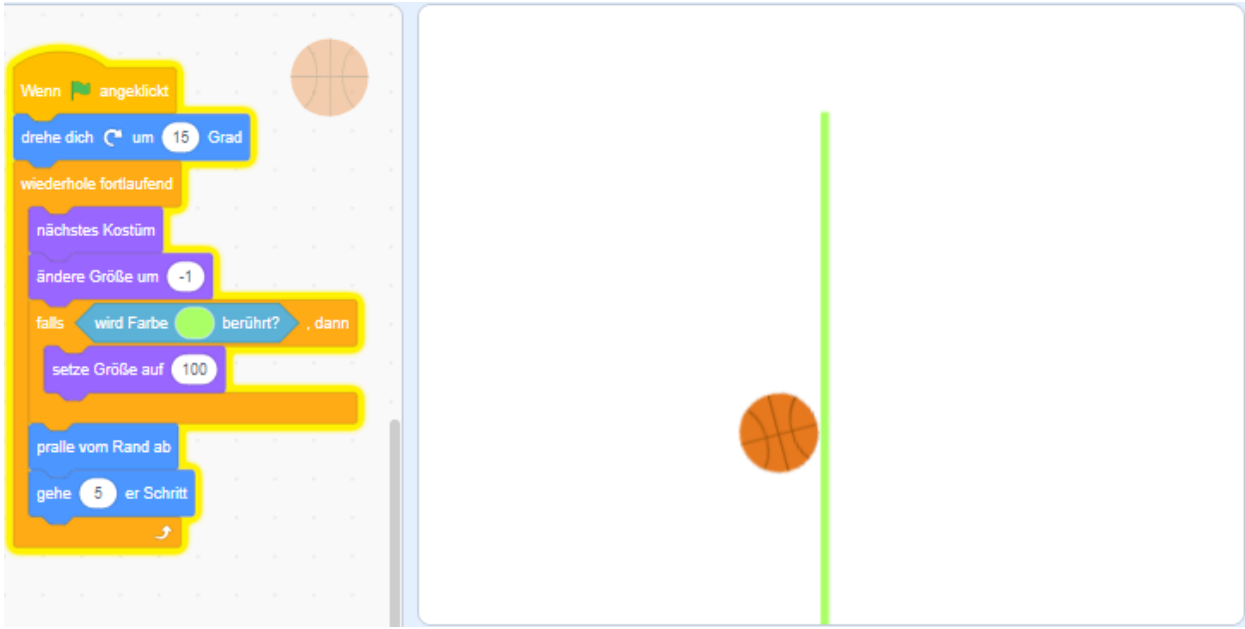
Was ist nun der Unterschied zwischen diesem Baustein und dem Menü direkt unter der Bühne?

Ganz einfach! Im Menü unter der Bühne muss man den Wert manuell eingeben. Der Baustein hingegen, kann in den Code mit eingebunden werden.

Beispiel:

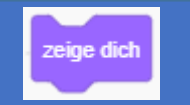
Ein Ball fliegt wahllos umher! Währenddessen schrumpft er stetig und wird erst wieder auf die normale Größe von (Wert: 100) zurückgesetzt, wenn er die grüne Linie berührt.

Wir bauen den unten abgebildeten Codeblock ganz einfach nach und sehen selbst!



Aussehen – zeige dich // verstecke dich

Baustein:



Erklärung:

Was tun, wenn man mehrere Figuren in einem Spiel oder etwas Ähnlichem braucht, man aber nicht alle Figuren gleich zu Beginn einsetzen möchte? Zu Anfang braucht man lediglich 2 Figuren, gegen Ende sollten es jedoch 4 sein?

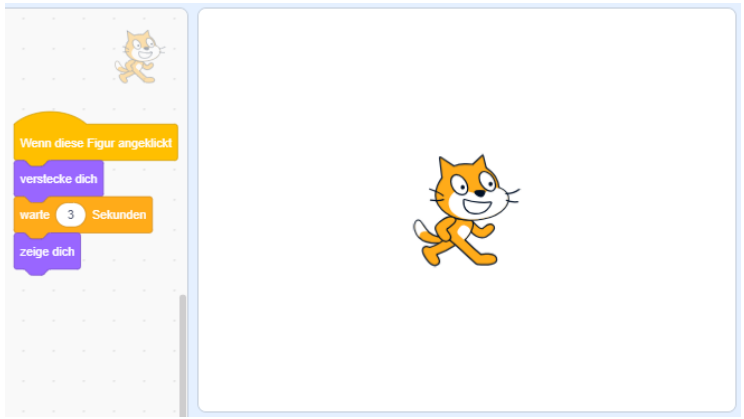
Manuell hinzufügen geht nicht! Hier kommen die beiden Bausteine „zeige dich“ und „verstecke dich“ zum Einsatz.

Beispiel:

Auch hier gibt es zwei verschiedene Möglichkeiten, eine Figur zu verstecken und sie wieder anzuzeigen.

Im Menü direkt unter der Bühne gibt es die Möglichkeit, eine Figur zu verstecken bzw. sie wieder anzuzeigen!

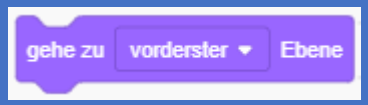
Andernfalls kann dies mit den Bausteinen „zeige dich“ und „verstecke dich“ erreicht werden.



Hier ein kurzes Beispiel zu den beiden Bausteinen, um ein besseres Verständnis, ihrer Funktionsweise zu erlangen.

Aussehen – gehe zu vorderster/hinterster Ebene

Baustein:



Erklärung:

Das Programm Scratch arbeitet mit Ebenen. Für ein besseres Verständnis kann man sich das wie bei einer Torte vorstellen. Diese besteht aus mehreren Schichten. Ganz unten ist der Boden der Torte, dann folgen die restlichen Schichten. Die Schichten des Kuchens haben eine bestimmte Reihenfolge und diese sollten keinesfalls vertauscht werden.

Ähnlich ist es in Scratch. Siehe Beispiel →

Hilfestellungen:

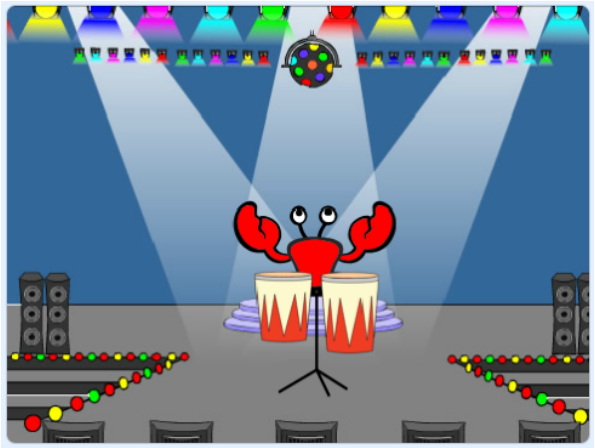
Was sind Ebenen?

Beispiel:

Die Krabbe versucht die Trommel zu spielen! Das ist jedoch leider nur möglich, wenn sie hinter der Trommel steht, ansonsten würde sie in die falsche Richtung schauen. Je nachdem, wie viele Figuren ein Programm beinhaltet, gibt es unterschiedlich viele Ebenen. Jede einzelne Figur hat eine eigene Ebene, auf der sie sich nur selbst befindet und sonst keine.

Die Krabbe steht vor der Trommel und schaut in die falsche Richtung.
→ So kann die Trommel NICHT gespielt werden.

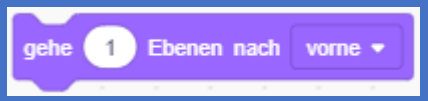
Die Krabbe steht hinter der Trommel und schaut somit in die richtige Richtung.
→ So kann die Trommel gespielt werden.



Mit dem auf dieser Seite erklärten Baustein kann genau dieses Problem behoben werden.
Man kann eine Figur so programmieren, dass sie sich auf der vordersten oder hintersten Ebene befindet.

Aussehen – gehe [Zahl eintippen] Ebenen nach vorne/hinten

Baustein:



Erklärung:

Wie auf der vorherigen Folie bereits erklärt, arbeitet Scratch mit sogenannten Ebenen. Jede Figur befindet sich auf einer eigenen Ebene. Dieser spezielle Baustein ermöglicht es einer Figur, um eine beliebige Anzahl von Ebenen nach vorne oder nach hinten zu springen.

Um diesen Baustein richtig anwenden zu können, muss man natürlich wissen, welche Figur auf welcher Ebene ist. Da diese Information aber nirgends abzulesen ist, kann dies relativ schwierig sein. Diese Information kann nur durch das

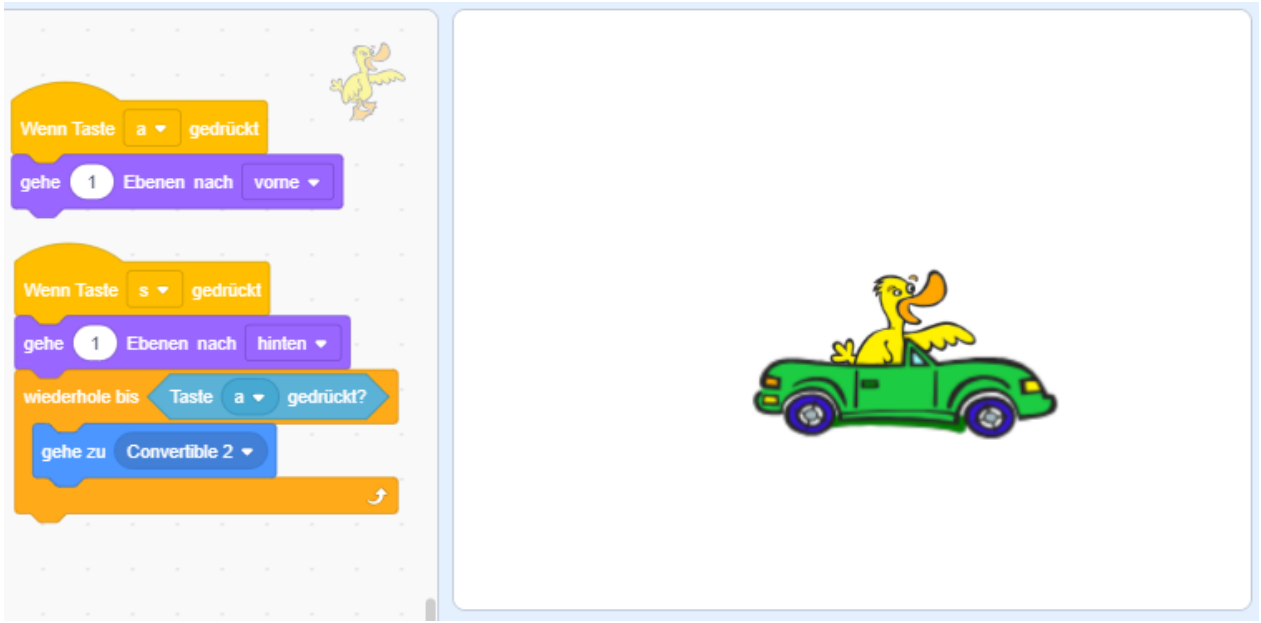
Hilfestellungen:

Was sind Ebenen?

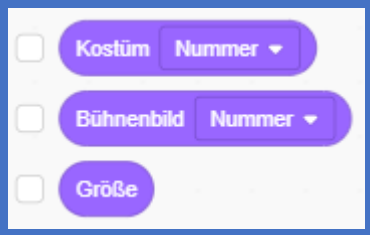
Beispiel:

Die Ente hat sich ein neues Auto gekauft. Wir können entscheiden, ob sie im Auto sitzt oder nicht.

Wir brauchen dazu die Ente und das grüne Auto. Mit den Tasten „a“ und „s“ kann die Ente dann in ihr Auto einsteigen. Damit sie auch ins Auto passt, ändern wir die Größe der Ente auf den Wert: 75.



Baustein:



Erklärung:

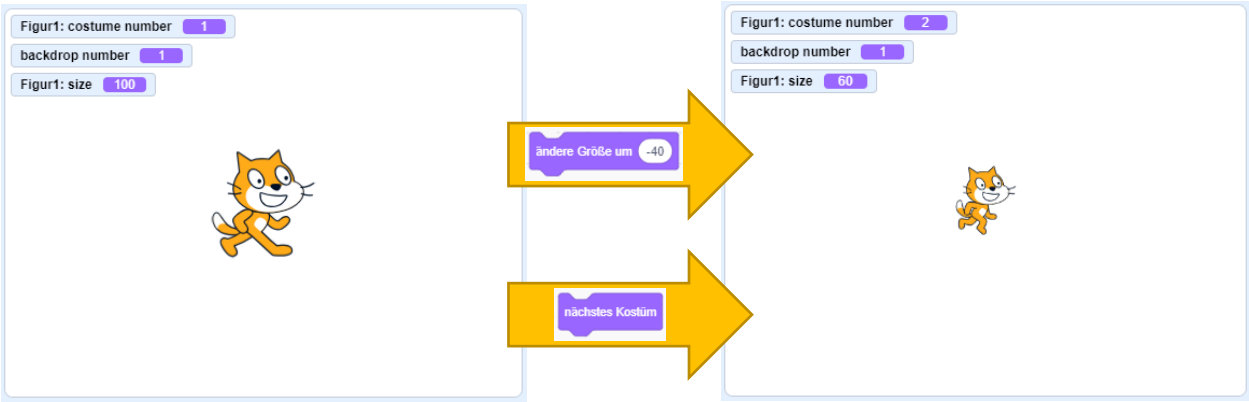
Variablen kann man sich wie eine Art Behälter, der Werte speichert, vorstellen. Diese Werte können dann aufgerufen und verwendet werden. In diesem konkreten Fall werden die x-Position, die y-Position und die Richtung gespeichert. Wie im Beispiel zu sehen ist, kann man diese Werte anzeigen lassen oder direkt in das Programm einbinden.

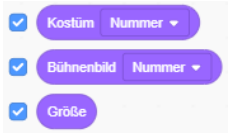
Hilfestellungen:

Variablen

Beispiel:

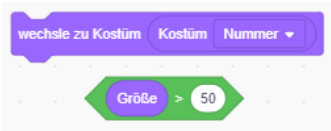
Aktiviert man in der Kategorie Aussehen die Häkchen neben den Variablen, wie es unten vorgemacht ist, dann werden diese auf der Leinwand in der linken oberen Ecke angezeigt. Dort werden dann in Echtzeit die jeweiligen Daten ausgegeben.



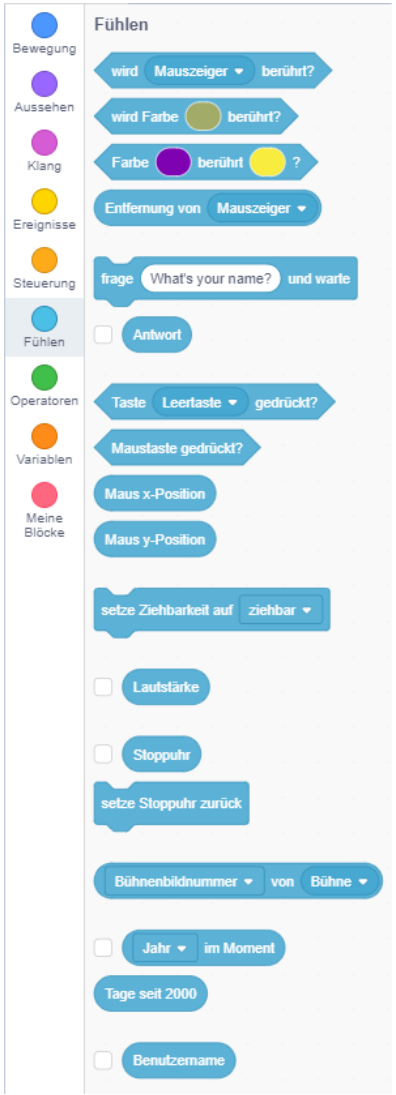


Hier wurden die Häkchen aktiviert, die Werte werden nun auf der Bühne angezeigt.

Die Variablen können auch in andere Bausteine eingesetzt werden. Um den aktuellen Wert direkt in das Programm miteinzubeziehen:



Bausteinkategorie: Fühlen



Übersicht:

Farbe: Hellblau

Aufgabe: Diese Kategorie von Bausteinen ist für das Wahrnehmen von bestimmten Dingen zuständig. Es kann abgefragt werden, ob eine Farbe berührt wird oder eine bestimmte Taste auf der Tastatur gedrückt ist. Beinahe alle Bausteine dieser Kategorie sind ausschließlich in andere Bausteine einsetzbar. Es wird ein bestimmter Wert abgefragt und dieser kann dann verwendet werden. Zusammenfassend geht es geht hauptsächlich um das Abfragen von bestimmten Dingen oder Werten. Das kann man auch daran erkennen, dass viele Bausteine dieser Kategorie ein Fragezeichen am Ende haben.

Anwendbar auf: Figuren, Bühnenbilder

Sonstige Hinweise: Nicht jeder Baustein dieser Kategorie ist auf Bühnenbilder und Figuren anwendbar. Bei manchen Bausteinen dieser Kategorie, wie zum Beispiel der Baustein „wird [Farbe] berührt“, ist es nicht sinnvoll, sie auf Bühnenbilder anzuwenden.

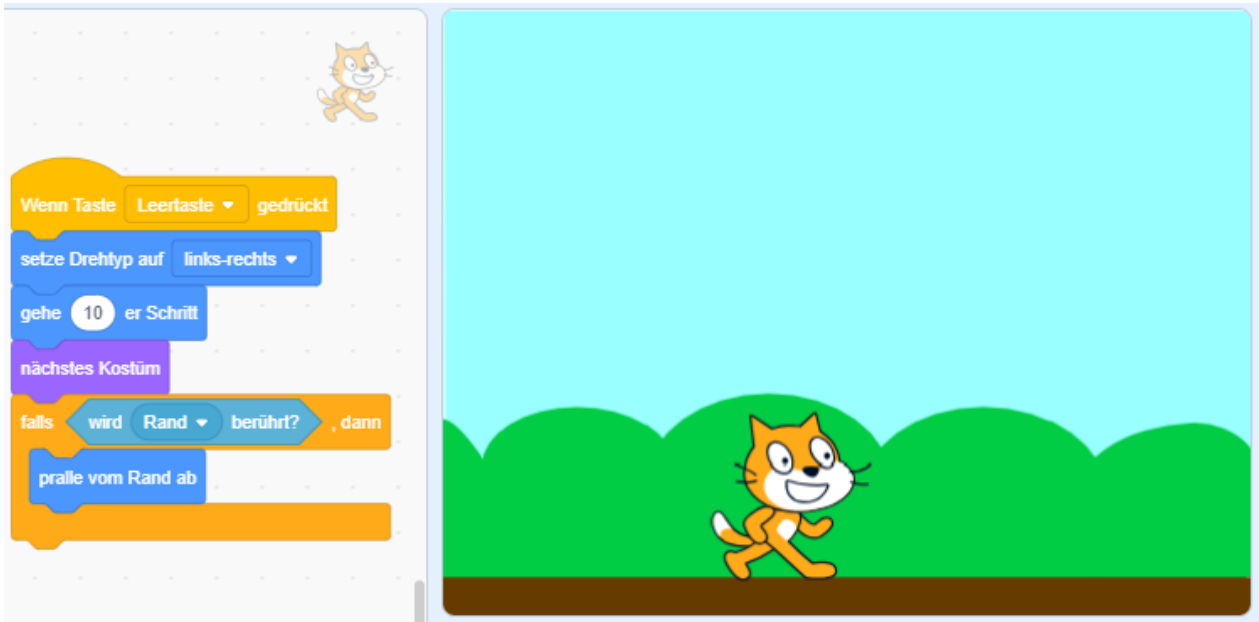
Baustein:

Erklärung:

Wie in der Beschreibung der Kategorie bereits erwähnt, geht es hier um das Erkennen von Zuständen. Dieser Baustein fragt ab, ob etwas berührt wird. Standardmäßig kann über ein Dropdown Menü zwischen dem Mauszeiger und dem Rand ausgewählt werden. Fügt man jedoch weitere Figuren hinzu, so stehen diese ebenfalls zur Auswahl.

Beispiel:

Scratch fragt sich, wann er den Rand berührt. Er läuft langsam und Schritt für Schritt nach vorne. Erreicht er den Rand, dreht er sich um und läuft solange in die andere Richtung, bis er am anderen Rand der Bühne angelangt ist.



Fühlen – wird Farbe [Farbe] berührt?

Baustein:

Erklärung:

Ob eine bestimmte Farbe von einer Figur berührt wird oder nicht, willst du wissen? Der oben abgebildete Baustein liefert dir die Antwort.

Mit einem Klick auf das farbige Feld kann die Farbe, nach der abgefragt werden soll, ganz einfach ausgewählt werden.

Wie man den Baustein jedoch genau anwendet, wird dir im Beispiel erklärt.

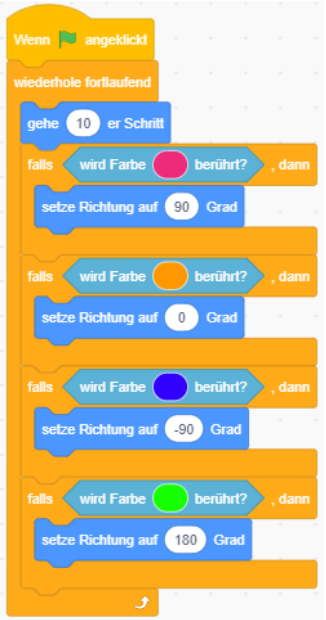
Hilfestellungen:

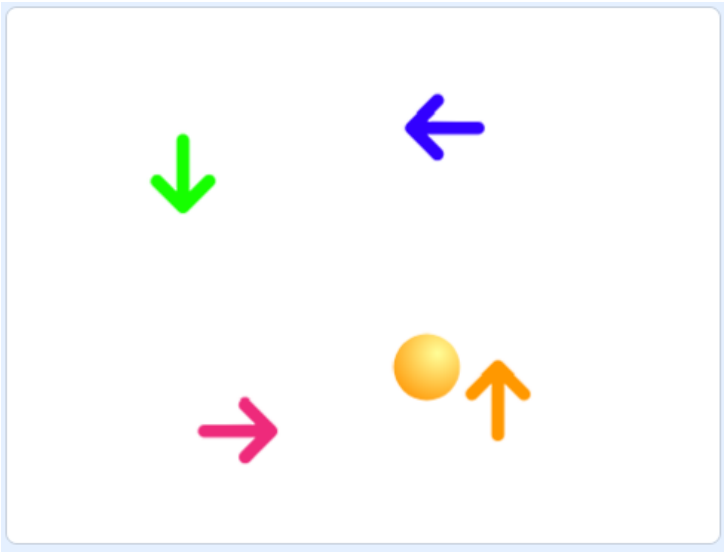
Kostüm Funktion

Beispiel:

Dieser Ball ändert jedes Mal die Richtung, wenn er einen dieser Pfeile berührt. Nur, woher weiß der Ball, wann er welchen Pfeil berührt? Ganz einfach, jeder Pfeil hat eine andere Farbe und diese kann mithilfe des „wird Farbe [Farbe] berührt?“ Bausteins vom Ball erkannt werden.

Du brauchst vier Mal die Figur „Arrow“ und einmal den „Ball“. Wechsle die Farbe der Pfeile mithilfe der Kostümfunktion. Los geht’s!





Baustein:

Erklärung:

Im Vergleich zum Baustein „wird Farbe [Farbe] berührt?“, überprüft dieser Baustein, ob eine Farbe, eine andere berührt.

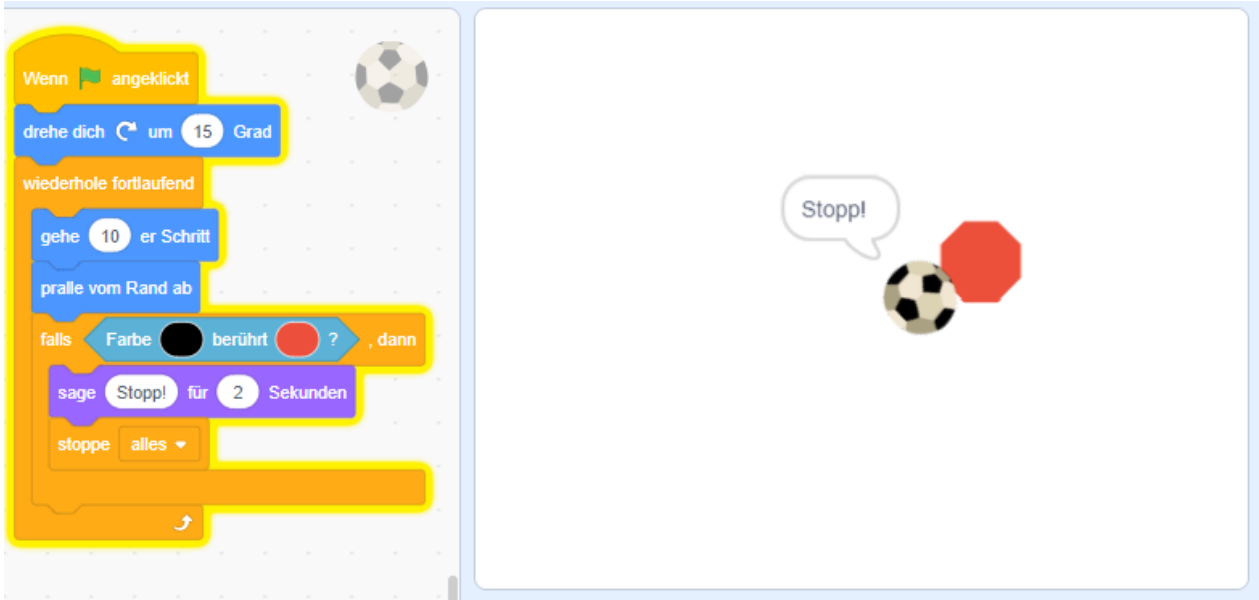
Beispielsweise:
Berührt die Farbe Rot die Farbe Blau?

Mit diesem Schritt kann etwas genauer abgefragt werden. Es ist zum Beispiel möglich, nach bestimmten Körperteilen einer Figur abzufragen. Scratch ist größtenteils orange, hat jedoch eine weiße Schwanzspitze.
Hier würde dann dieser Baustein zum Einsatz kommen.

Beispiel:

Der Ball fliegt durch die Gegend und soll stoppen, sobald eines seiner schwarzen Fünfecke, das rote Stoppschild berührt.

Du kannst das Stoppschild auch etwas kleiner oder größer machen – ganz so, wie du das gerne hättest.



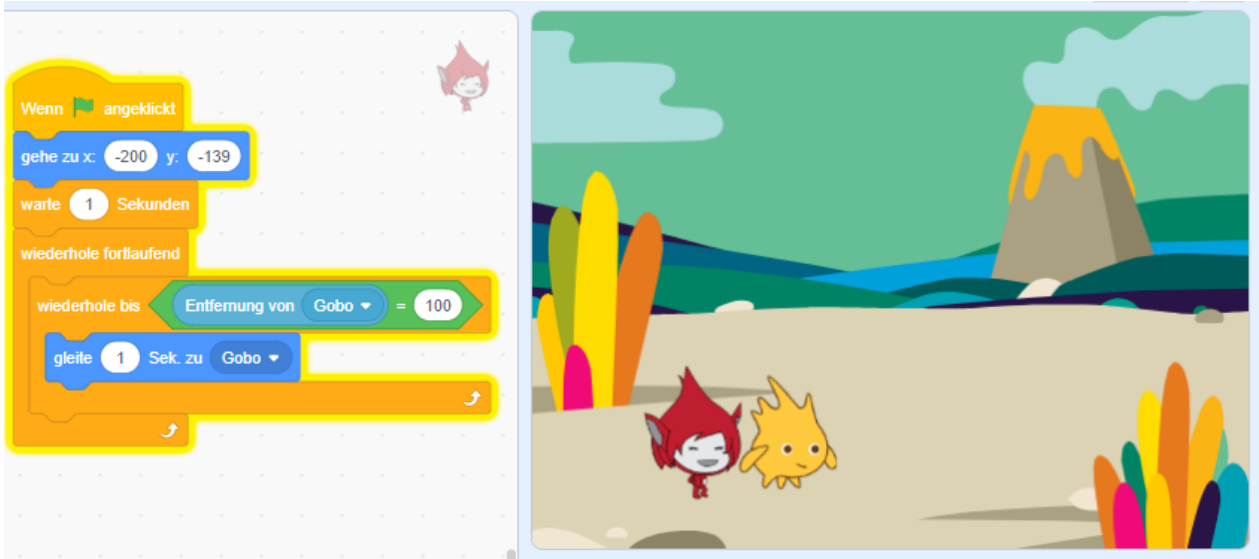
Baustein:

Erklärung:

Dieser Baustein gibt standardmäßig die Entfernung zwischen einer Figur und dem Mauszeiger als Wert zurück. Fügt man jedoch weitere Figuren hinzu, kann auch die Entfernung zwischen zwei oder mehreren Figuren als Wert zurückgegeben werden.

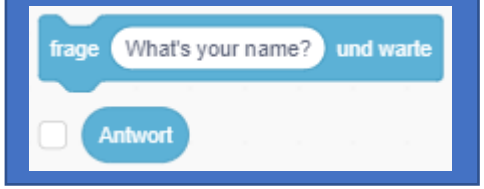
Beispiel:

Gobo und sein bester Freund Giga leben in einer Zeit, in der Herumlaufen alleine sehr gefährlich ist. Deswegen folgt Giga Gobo auf Schritt und Tritt während ihrer Reise zum Fuße des Vulkans.



Fühlen – frage [Text eintippen] und warte

Baustein:



Erklärung:

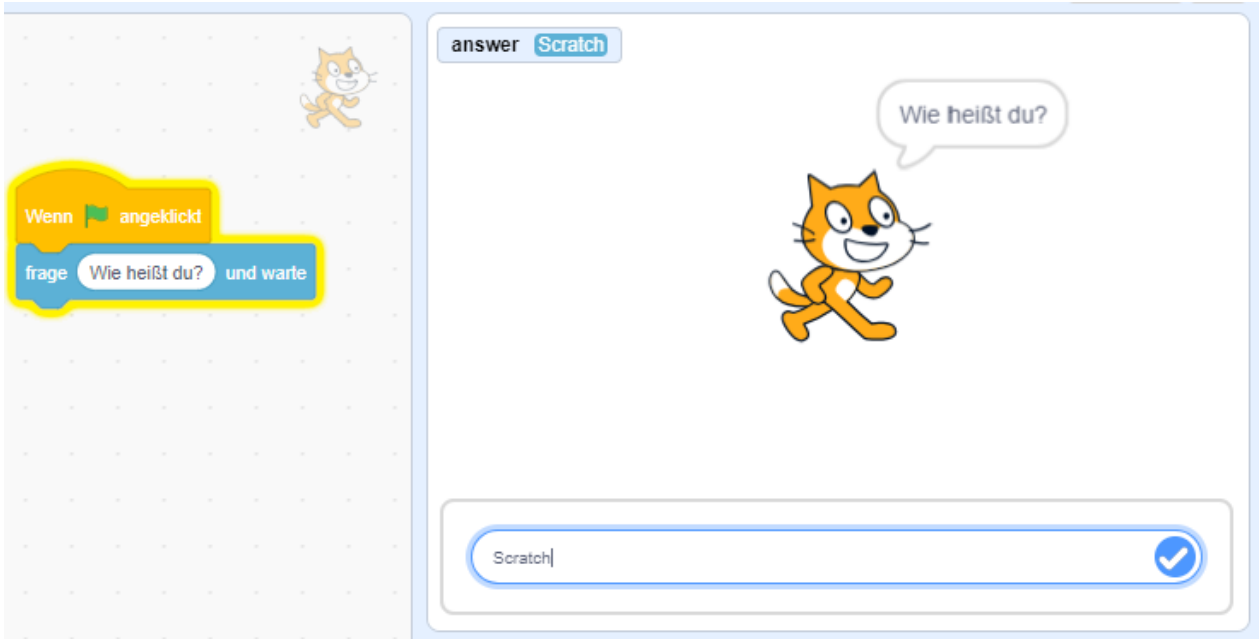
Dieser Baustein ist mehr Schmuck als Notwendigkeit. Mit diesem Baustein ist es möglich, der Spielerin oder dem Spieler eine Frage zu stellen. Diese kann dann beantwortet und die Antwort gespeichert werden. Ein paar Beispiele sind der Benutzername, das Alter oder das Geschlecht der Spielerin / des Spielers in Erfahrung zu bringen.
Die dazugehörige Variable „Antwort“ speichert dann den eingegebenen Wert und kann so auch abgerufen werden.

Hilfestellungen:

Variablen

Beispiel:

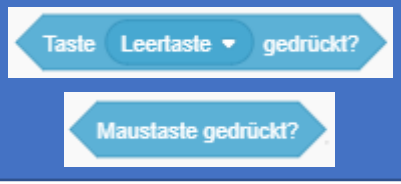
Scratch fragt uns, wie wir heißen. Wir schreiben unseren Namen in das Feld und klicken auf den Haken, um das zu bestätigen.



Wie alle Variablen kann man auch die „Antwort“ in andere Bausteine einsetzen und damit arbeiten:

Fühlen – Taste [Taste] gedrückt? // Maustaste gedrückt?

Baustein:



Erklärung:

Hier wird abgefragt, ob eine Taste, die nach Belieben gewählt werden kann, gedrückt ist.

Der andere Baustein hat ein und dieselbe Funktionsweise, fragt jedoch ab, ob die Maustaste gedrückt ist.

Im Hintergrund wird auf die Frage dann mit einer von zwei möglichen Antworten geantwortet.

Taste ist gedrückt:
→ Die Aussage ist wahr.

Taste ist nicht gedrückt:
→ Die Aussage ist falsch (unwahr).

Hilfestellungen:

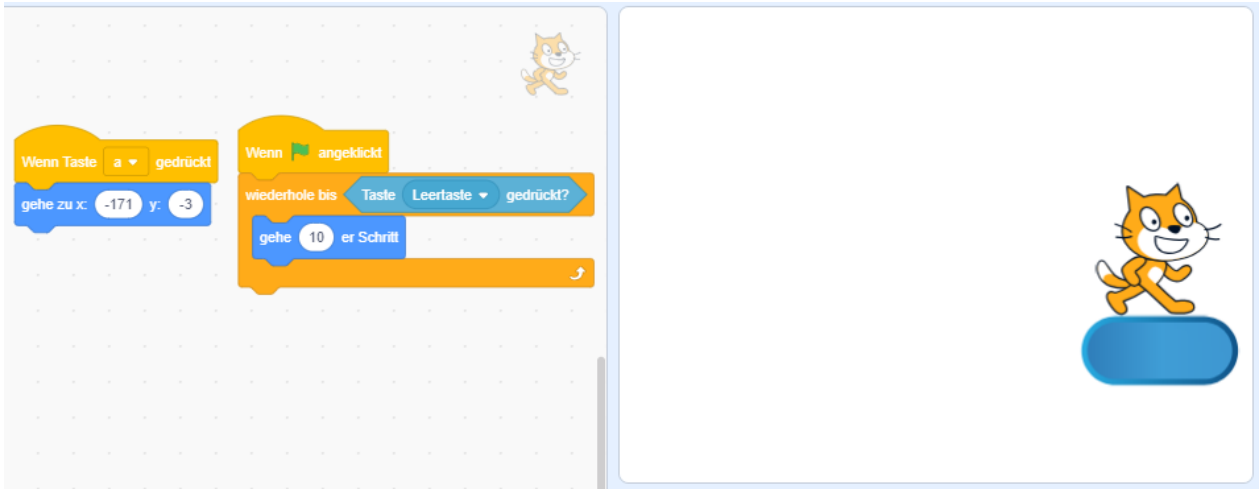
Boolesche Werte

Beispiel:

Scratch muss solange laufen, bis er auf der blauen Fläche steht. Es geht um unser Timing, wir müssen genau, wenn die Katze über der blauen Fläche steht, die Leertaste drücken, um ihn zu stoppen.

Tipp: Wir ändern die Geschwindigkeit der Katze, in dem wir die Zahl des „gehe [Zahl eintippen] er Schritt“ Bausteins erhöhen oder verkleinern. Das macht das Spiel dann einfacher oder etwas schwerer.

Mit einem Druck auf die Taste „a“ auf unserer Tastatur, setzen wir Scratch zurück an den Start.



Baustein:

Erklärung:

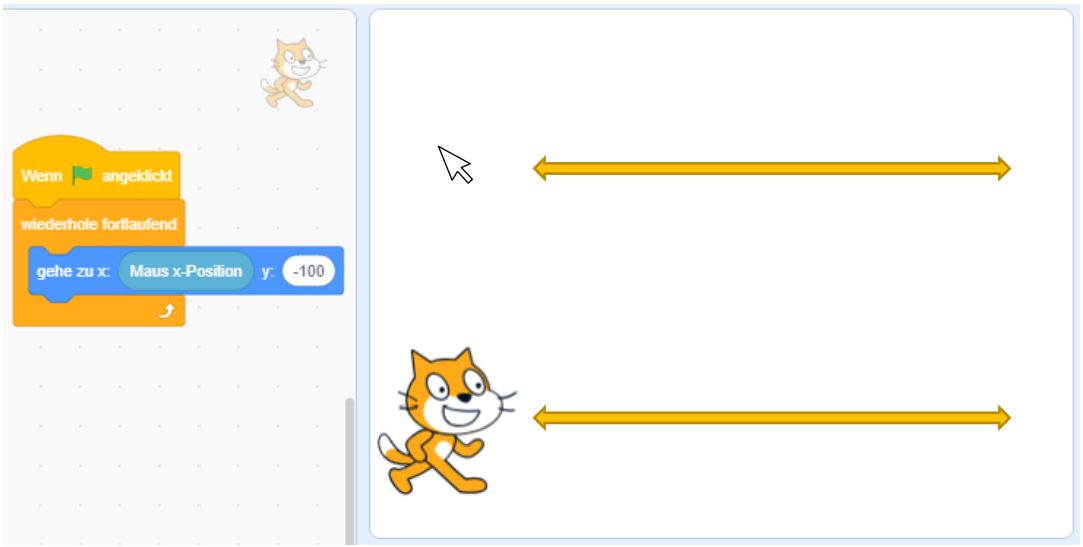
Diese beiden Variablen beziehen sich auf die aktuelle x- und y-Position des Mauscursors. Man muss jedoch beachten, dass diese beiden Variablen nicht über den Rand der Bühne hinaussehen können. Das bedeutet, es gibt einen maximalen und einen minimalen Wert, den diese Variablen zurückgeben können.

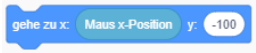
x-Position:
maximaler Wert: 240
minimaler Wert: -240

y-Position:
maximaler Wert: 180
minimaler Wert: -180

Beispiel:

In einem Spiel können diese beiden Variablen eine Möglichkeit zur Steuerung der Figur sein. Die y-Position wird nicht verändert, der x-Wert hingegen ändert sich mit der Bewegung der Maus.



Wie alle Variablen können auch „Maus x-Position“ und „Maus y-Position“ in andere Bausteine eingesetzt werden. Wie man oben im Beispiel auch schon sieht, kann dann damit gearbeitet werden: 

Anders als die meisten anderen Variablen können diese beiden jedoch nicht, über einen Klick auf das Häkchen, auf der Bühne angezeigt werden. Das Häkchen fehlt diesen Variablen.

Baustein:

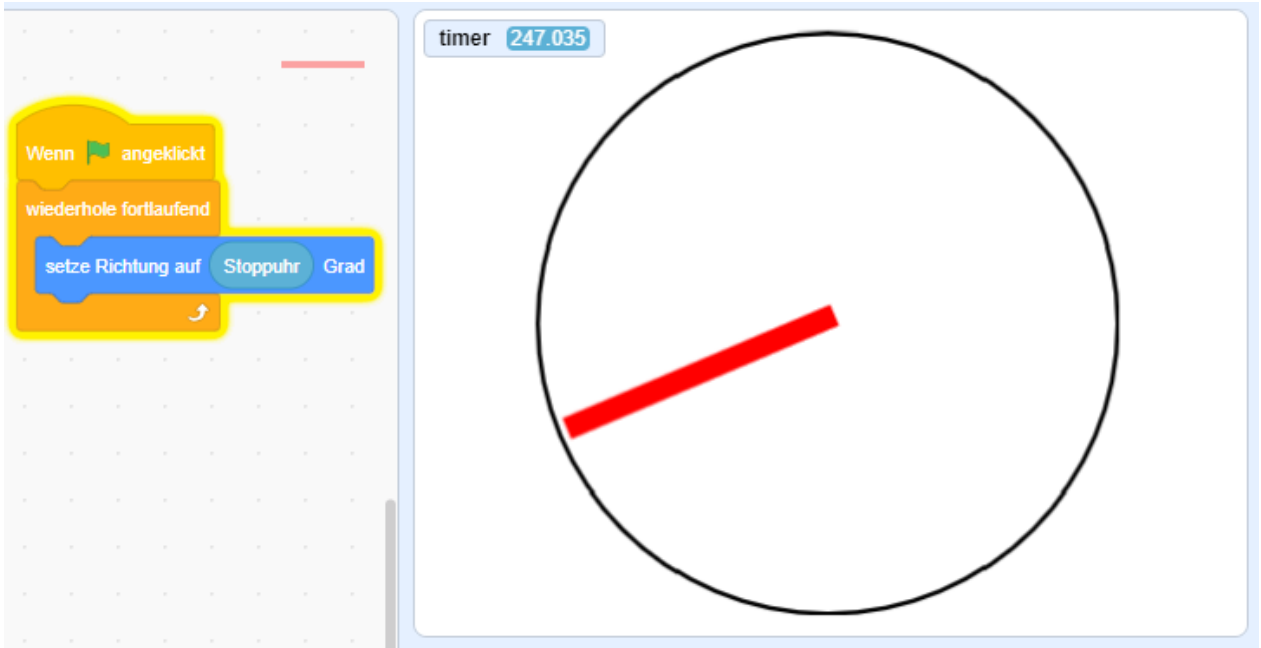
Erklärung:

Ein weiteres Variablen-Bausteinduo, das zusammengehört. Die Stoppuhr ist eine Variable, die ständig nach oben zählt. Sie macht nie pause und kann nur mit dem dazugehörigen Baustein „setze Stoppuhr zurück“ zurückgesetzt werden.
Die Stoppuhr kann mit einem Klick auf das danebenliegende Kästchen auf der Bühne angezeigt werden.
Das sieht dann so aus: 

Die Stoppuhr kann einerseits hilfreich zur Messung der Zeit sein, andererseits aber auch verwendet werden, um gegen die Zeit zu spielen.

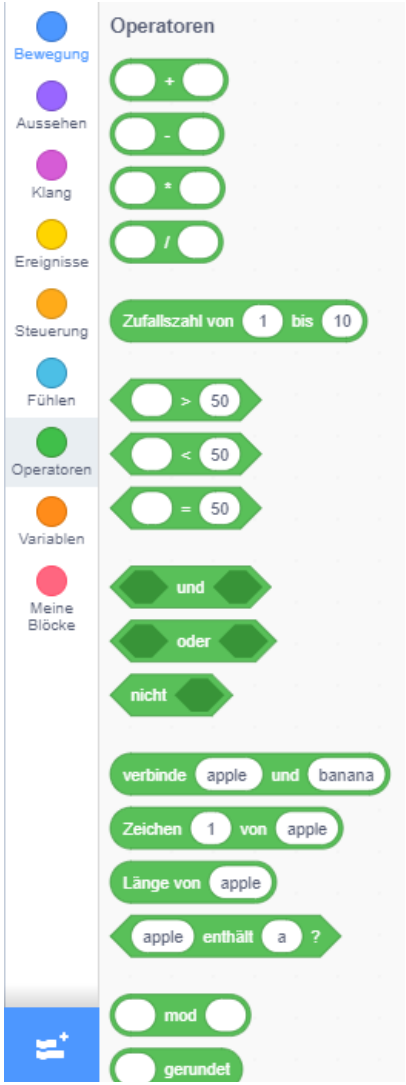
Beispiel:

Bei dieser Uhr stimmt etwas nicht! Wie kann es denn 247,035 Uhr sein?! Ganz einfach, diese Uhr zeigt nicht die Zeit an, sondern die Gradwerte innerhalb eines Kreises.



Die Variable Stoppuhr hat jedoch noch viel mehr Funktionen.
Man kann sie in Rechnungen miteinbeziehen, sie neustarten oder auch anzeigen lassen!

Bausteinkategorie: Operatoren



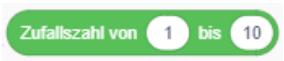
Übersicht:

Farbe: Grün

Aufgabe: In der Kategorie der mathematischen Operatoren dreht sich alles um das Ausrechnen von Dingen. Ob man nun zwei Werte addieren muss oder wissen will, ob ein Wert kleiner oder größer als ein anderer ist, dann ist man hier genau richtig.

Anwendbar auf: Figuren, Bühnenbilder

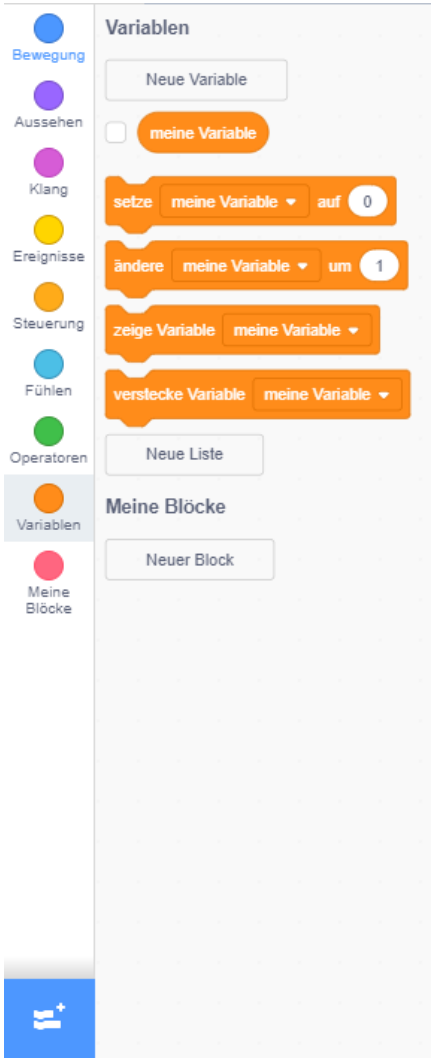
Sonstige Hinweise: Alle Bausteine sind sowohl auf Bühnenbilder als auch auf Figuren anwendbar. Jedoch macht es nicht immer Sinn, diese einzusetzen.

Bausteine: 	In diese Bausteine können entweder händisch oder per manueller Eingabe Zahlen geschrieben werden oder andere „abgerundete“ Bausteine (z.B.: Variablen, …) eingesetzt werden. So kann dann mit statischen, aber auch mit Live-Werten gearbeitet werden. Unabhängig davon, welche Art von Wert man in die dafür vorgesehenen Felder schreibt, werden diese dann addiert, subtrahiert, multipliziert oder dividiert. Die jeweilige Rechenoperation hängt vom gewählten Baustein ab.
Baustein: 	Das ist ein Baustein, der eine Zufallszahl innerhalb eines gewissen Wertebereichs ausgibt. So können beispielsweise zufällige Koordinaten ausgerechnet werden. In die dafür vorgesehenen Felder können über die Tastatur Werte eingetippt werden oder „abgerundete“ Bausteine (z.B.: Variablen, …) eingesetzt werden.
Bausteine: 	Größer, kleiner, gleich … das ist hier die Frage. Mit diesen drei Bausteinen können zwei Werte miteinander verglichen werden. Auch hier können wieder Live Werte (z.B.: Variablen, …) oder statische Werte eingesetzt werden.
Bausteine: 	Mithilfe dieser drei Bausteine können beispielsweise Bedingungen verknüpft werden. Angenommen, man will, dass ein bestimmtes Skript nur dann ausgeführt wird, wenn zwei Bedingungen zugleich zutreffen, dann verwendet man den „[Bedingung 1] und [Bedingung 2]“ Baustein.
Baustein: 	Dieser Baustein verbindet zwei Wörter oder Zahlen miteinander. Wie man in diesem Beispiel sieht, wird aus den beiden Wörtern „Apfel“ und „kuchen“ das kombinierte Wort „Apfelkuchen“. Dasselbe Prinzip gilt für Zahlen. Doch aufgepasst - dieser Baustein addiert die Zahlen nicht miteinander, er fügt sie zusammen. Das bedeutet, verbindet man „1“ mit „2“, erhält man nicht „3“, sondern „12“!
Baustein: 	Bei diesem Baustein kann man ein bestimmtes Zeichen eines Wortes oder einer Zahl in Erfahrung bringen. Wie man im Beispiel sieht, soll das erste Zeichen des Wortes „Scratch“ ausgegeben werden. Das ist hier der Buchstabe „S“ und wie man sieht, wird auch genau dieses Wort ausgegeben.

<table>
<tr>
<td>Baustein:
 
</td>
<td>Mit diesem Baustein kann man die Länge eines Wortes oder einer Zahl in Erfahrung bringen.
<table>
<tr><td>P</td><td>r</td><td>o</td><td>g</td><td>r</td><td>a</td><td>m</td><td>m</td><td>i</td><td>e</td><td>r</td><td>e</td><td>n</td></tr>
<tr><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td></tr>
</table>
Dasselbe Prinzip gilt, wie auch schon erwähnt für Zahlen. Die Zahl „100“ hat somit als Ergebnis „3“.</td>
</tr>
<tr>
<td>Baustein:
 
</td>
<td>Dieser Baustein prüft, ob ein bestimmtes Zeichen in einer Zahl oder in einem Wort enthalten ist.
Da das Wort „Suppe“ ein „e“ enthält, ist der Rückgabewert „true“ oder zu Deutsch „wahr“!</td>
</tr>
<tr>
<td>Baustein:
 
</td>
<td>Dieser Baustein rechnet zwei Werte miteinander Modulo. Das bedeutet, es wird ausgerechnet, wie oft der zweite Wert in den ersten Wert hineinpasst. Sollte dann ein Rest übrigbleiben, wird dieser ausgegeben.

„4“ passt „2“ Mal in „10“ hinein. Jedoch ist „2*4=8“. Das bedeutet, es bleibt „2“ als Rest übrig. Dieser Wert wird dann ausgegeben. Diese Rechenoperation rechnet den Rest aus.</td>
</tr>
<tr>
<td>Baustein:
 
</td>
<td>Hier wird der eingegebene Wert gerundet. Egal, ob es sich um einen statischen Wert oder um einen Live Wert handelt, dieser Baustein rundet die Zahl nach den Grundregeln des Rundens.</td>
</tr>
<tr>
<td>Baustein:
 
</td>
<td>Alle relevanten, noch nicht erwähnten mathematischen Operationen werden in diesem Baustein vereint. Die Funktionen, die dieser Baustein anbietet, lauten:
Betrag, abrunden, aufrunden, Wurzel, Sinus, Cosinus, Tangens, Arcus Sinus, Arcus Cosinus, Arcus Tangens, Logarithmus Naturalis, Logarithmus, e^, 10^;
Der letzte Baustein der Kategorie Operatoren ist somit vielseitig einsetzbar.</td>
</tr>
</table>

Bausteinkategorie: Variablen



Übersicht:

Farbe: Orange

Aufgabe: Es gibt bereits sehr viele vordefinierte Variablen in Scratch, die standardmäßig zur Verfügung stehen. Jedoch könnten niemals alle Variablen von den Herstellern schon vordefiniert werden. Manche Variablen kommen viel zu selten zum Einsatz, andere wiederum sind viel zu speziell und somit schlicht und einfach unmöglich vordefinierbar.

Hierzu gibt es die Kategorie der Variablen, in der selbst Variablen definiert werden können. Diese selbst definierten Variablen können dann auch bearbeitet und eingesetzt werden.

Anwendbar auf: Figuren, Bühnenbilder

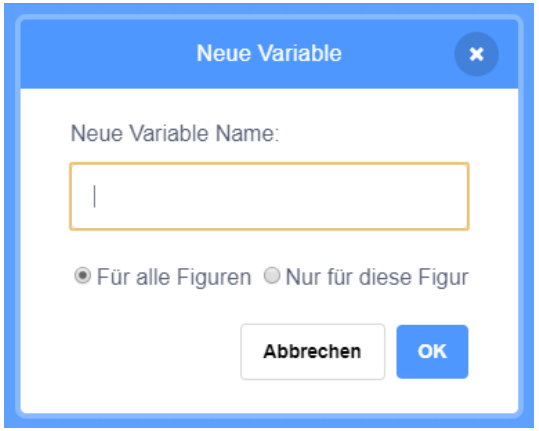
Sonstige Hinweise: Alle Bausteine sind sowohl auf Bühnenbilder als auch auf Figuren anwendbar. Jedoch macht es nicht immer Sinn, diese einzusetzen.

Erklärung:

Auf diesem Button kann man eine neue Variable erstellen. Man klickt einfach drauf und folgt der Anleitung.

Anleitung:

Nachdem wir auf den Button „Neue Variable“ geklickt haben, öffnet sich dieses Dialogfeld.



Schritt 1: Wir geben einen Namen für unsere neue Variable ein.

Schritt 2: Dann wählen wir aus, für den die Variable zugänglich ist.

Schritt 3: Anschließend klicken wir auf OK.

Baustein:

Erklärung:

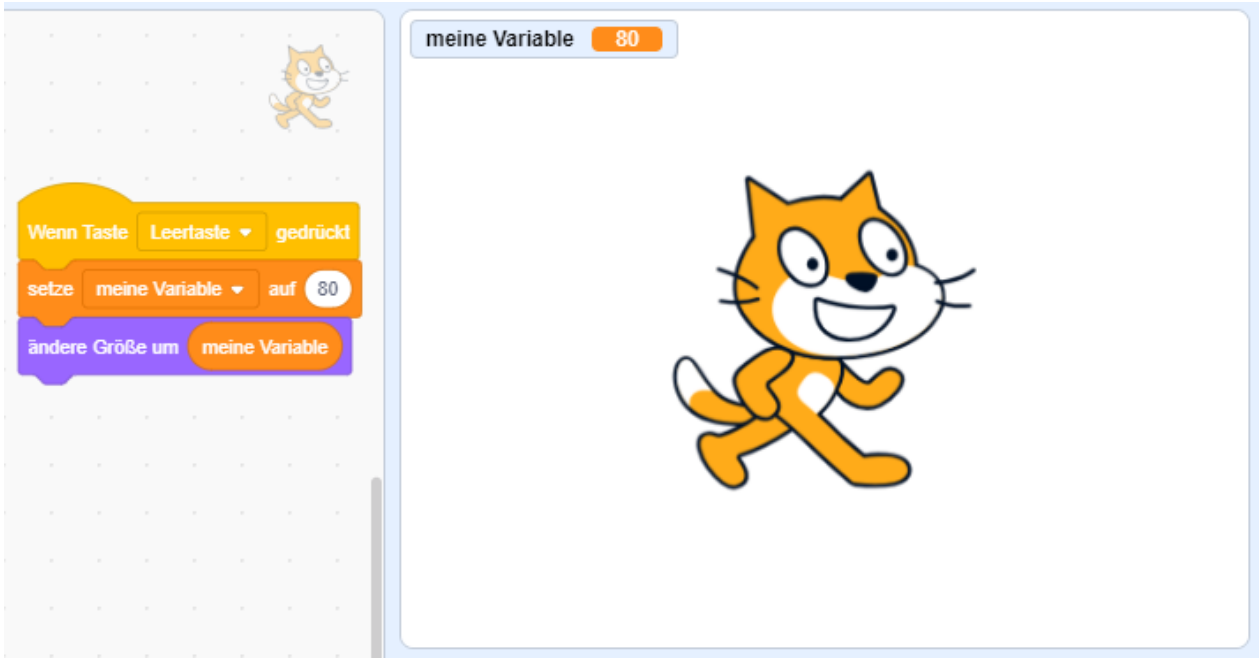
Eine vorgefertigte Variable, die keine genaue Aufgabe hat. Sie kann einfach als leere Variable verwendet werden. Sie kann umbenannt und gelöscht werden.

Diese Variable ist nicht zu verwechseln mit der Variablen, die man selbst erzeugt hat. Diese Variable ist standardmäßig schon im Programm.

Man kann dieser Variable einen Wert zuteilen und mit ihr arbeiten.

Beispiel:

Zuerst muss man der Variable einen Wert zuweisen, dann kann man mit diesem Wert machen, was man will. In diesem Fall wird die Größe von Scratch um den Wert der Variable verändert.



Variablen – setze [Variable] auf [Zahl eintippen/Wort eintippen]

Baustein:

Erklärung:

Mit diesem Baustein kann man eine Variable bearbeiten. Man kann ihr einen Wert zuweisen, sie umbenennen oder gar löschen.

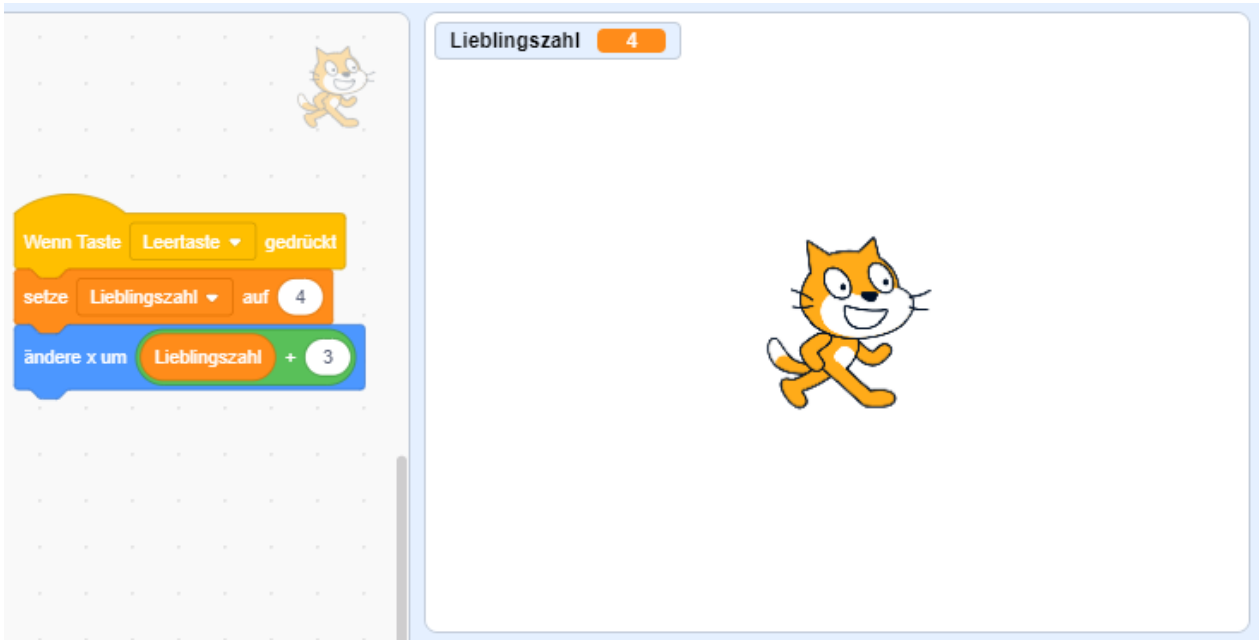
Diese weiteren Funktionen verbergen sich im Dropdown Menü dieses Bausteins. Man klappt es auf, um die volle Funktion des Bausteins zu erkunden.

Dieser Baustein kann im Umgang mit Variablen sehr hilfreich sein.

Beispiel:

Zuerst wurde eine Variable „Lieblingszahl“ erzeugt. Dieser wurde dann der Wert 4 zugewiesen. Nun soll sich Scratch immer um den Wert der Variable „Lieblingszahl“ + 3 in x-Richtung bewegen.

Den Wert unserer Variablen können wir selbst bestimmen und nach Belieben verändern.



Variablen – ändere [Variable] um [Zahl eintippen]

Baustein:

Erklärung:

Ein Baustein der den Wert einer Variablen verändern kann.

Man wählt dafür die Variable, die man mithilfe des Dropdown Menüs verändern will, aus und gibt den Wert ein, um den die Variable verändert werden soll.

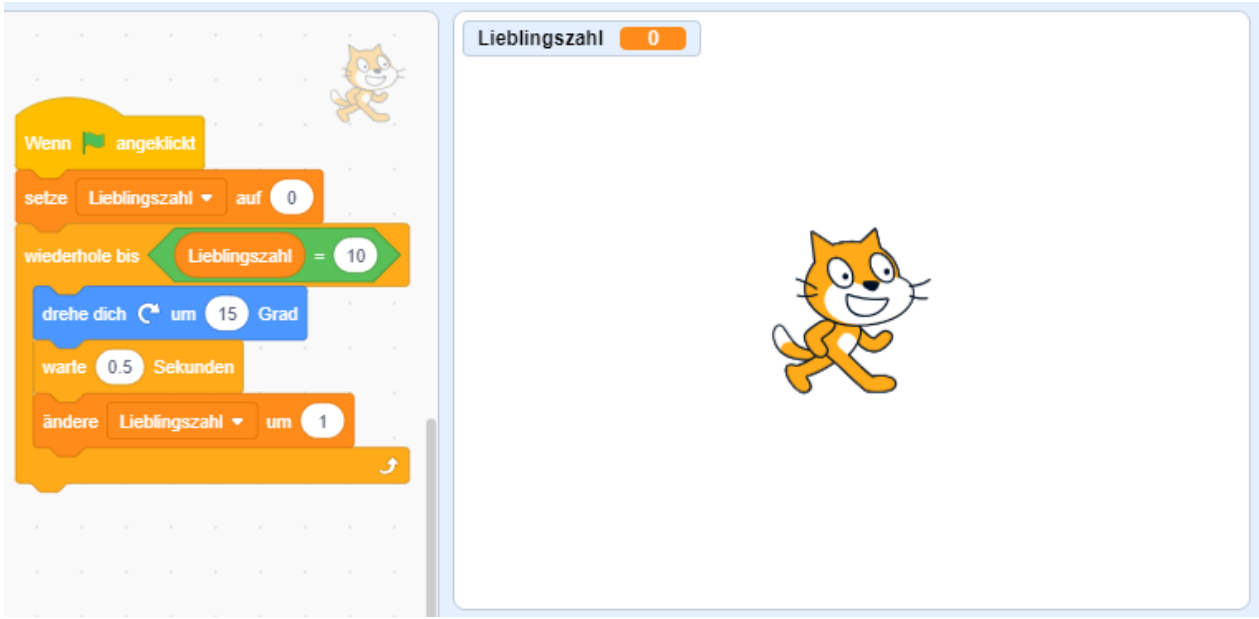
Du kannst den Wert deiner Variablen jeder Zeit manuell ändern. Mithilfe dieses Bausteins jedoch kannst du eine solche Änderung mit in den Programmcode miteinbeziehen.

Beispiel:

Wir arbeiten immer noch mit der Variablen „Lieblingszahl“. Zu Beginn wird diese auf „0“ gesetzt.

Scratch soll sich um „15“ Grad drehen, kurz warten und dann die Variable „Lieblingszahl“ um „1“ erhöhen. Das soll solange geschehen, bis sie einen Wert von „10“ erreicht hat.

Los geht’s! Wir probieren es jetzt einfach selbst aus!



Variablen – zeige Variable [Variable] // verstecke Variable [Variable]

Baustein:

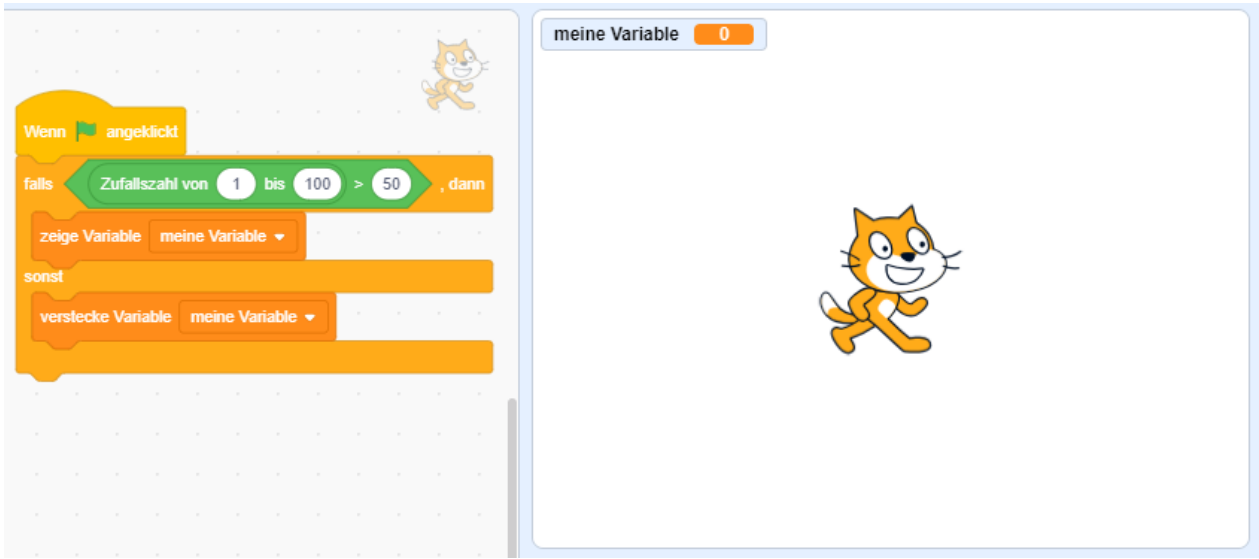
Erklärung:

Diese zwei Bausteine können die Variable anzeigen oder verstecken, ohne das Häkchen anklicken zu müssen.

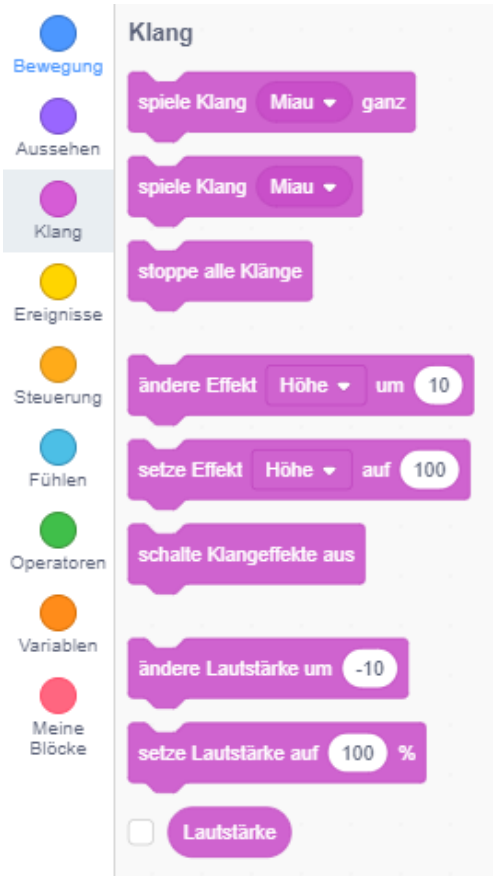
Manuell können Variablen immer ein- oder ausgeblendet werden. Doch so kann diese Funktion mit in den Programmcode einbezogen werden.

Beispiel:

Der Zufall entscheidet, ob die Variable angezeigt wird oder nicht! Ist die Zufallszahl größer als „50“, so wird die Variable angezeigt, ist sie kleiner, so wird sie versteckt.



Bausteinkategorie: Klang



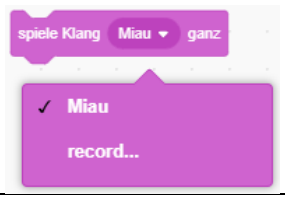
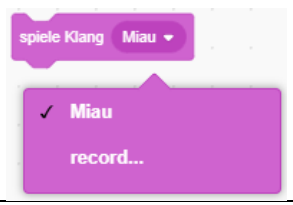
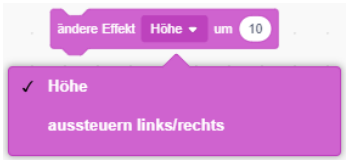
Übersicht:

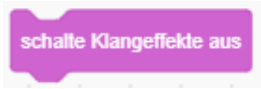
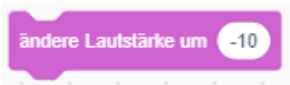
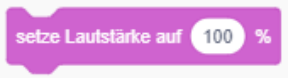
Farbe: Pink

Aufgabe: Die Klangbausteine geben einen Ton über die Lautsprecher des Endgerätes aus. Man kann eigene Töne aufnehmen und diese ausgeben lassen, man kann jedoch auch schon vorgefertigte Klänge ausgeben.

Anwendbar auf: Figuren, Bühnenbilder

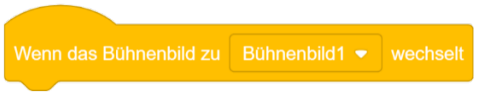
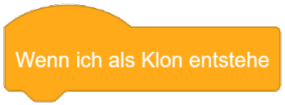
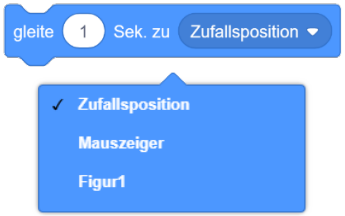
Sonstige Hinweise: Alle Bausteine sind sowohl auf Bühnenbilder als auch auf Figuren anwendbar. Jedoch macht es nicht immer Sinn, diese auch einzusetzen.

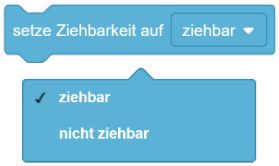
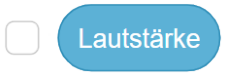
Baustein: 	Mit diesem Baustein wird ein Klang ganz abgespielt. Das bedeutet, es wird solange gewartet, bis der Klang ganz zu Ende gespielt ist. Solange also der Klang ertönt, können die darunterliegenden Bausteine nicht ausgeführt werden. Es ist über das Dropdown Menü auch möglich, einen eigenen Klang aufzunehmen. Das setzt voraus, dass das Endgerät, auf dem man Scratch verwendet über ein Mikrofon verfügt.
Baustein: 	Dieser Baustein spielt den Klang im Hintergrund ab. Das heißt, auch darunterliegende Bausteine können ausgeführt werden, während der Klang abgespielt wird. Auch bei diesem Baustein ist es über das Dropdown Menü möglich, einen eigenen Klang aufzunehmen. Das setzt voraus, dass das Endgerät, auf dem man Scratch verwendet über ein Mikrofon verfügt.
Baustein: 	Alle Klänge werden mit diesem Baustein gestoppt, auch wenn sie noch nicht zu Ende gespielt sind. Die Klänge verstummen mit sofortiger Wirkung.
Baustein: 	Dieser Baustein kann einen Effekt auf einen Klang legen. Der Basiswert des Klangs wird jedoch nur verändert und nicht neu gesetzt. Es kann die Höhe eines Tones verändert werden. Eine negative Zahl macht den Ton tiefer, eine positive macht in höher. (Der Wertebereich ist in beide Richtungen unbegrenzt.) • Ein Ton kann auch nach links oder rechts ausgesteuert werden. Hierfür werden entweder Kopfhörer oder mindestens zwei Lautsprecher benötigt. Ein Ton wird dann links oder rechts lauter ausgegeben. Eine negative Zahl verlagert den Ton nach links, eine positive Zahl verlagert den Ton nach rechts. (Der Wertebereich ist in beide Richtungen unbegrenzt.)
Baustein: 	Dieser Baustein kann einen Effekt auf einen Klang legen. Der Basiswert des Klangs wird ganz neu gesetzt Es kann die Höhe eines Tones verändert werden. Eine negative Zahl macht den Ton tiefer, eine positive macht in höher. (Der Wertebereich ist in beide Richtungen unbegrenzt.) • Ein Ton kann auch nach links oder rechts ausgesteuert werden. Hierfür werden entweder Kopfhörer oder mindestens zwei Lautsprecher benötigt. Ein Ton wird dann links oder rechts lauter ausgegeben. Eine negative Zahl verlagert den Ton nach links, eine positive Zahl verlagert den Ton nach rechts. (Der Wertebereich ist in beide Richtungen unbegrenzt.)

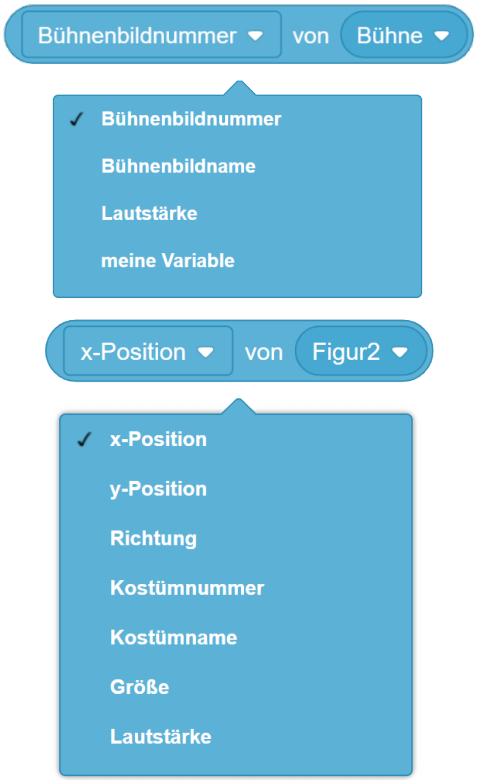
Baustein: 	Alle Klangeffekte werden mit sofortiger Wirkung ausgeschaltet. Der Ton spielt dann wieder auf seinen Grundeinstellungen.
Baustein: 	Die Ausgabelautstärke wird um den eingetippten Wert verändert. Man kann die Lautstärke auch manuell einstellen. Auf diese Weise kann jedoch eine Änderung der Lautstärke in den Programmcode miteinbezogen werden.
Baustein: 	Die Lautstärke wird neu festgelegt. Man beachte, dass die Zahl in Prozent anzugeben ist. Die Lautstärke kann auch manuell festgelegt werden. So kann das neue Festlegen der Lautstärke mit in den Programmcode einbezogen werden.
Variable: 	Diese Variable speichert den Wert der aktuellen Lautstärke. Dieser Wert kann dann in die dafür vorgesehenen „abgerundeten“ Felder eingesetzt und abgerufen werden. Mit einem Klick auf das Häkchen kann die Variable auf der Bühne angezeigt werden.

Sonstige Bausteine

Hier werden Bausteine aus verschiedenen Kategorien erklärt, die sehr selten und in besonderen Fällen angewendet werden. Diese kommen im Aufgabenhandbuch für Schüler nicht vor.

	Sobald das im Baustein ausgewählte Bühnenbild erscheint, aktiviert sich dieser Ereignisbaustein und startet das Programm, wo sich unter dem Ereignisbaustein befindet.
	Mit „sende **Nachricht** an alle“ können Nachrichten an alle Figuren versendet werden. Der Ereignisbaustein „Wenn ich Nachricht1 empfange“ ist der Empfänger dieser Nachricht. Wird in diesem Fall Nachricht1 gesendet, empfängt diese der Ereignisbaustein „Wenn ich Nachricht1 empfange“ und aktiviert sich und das Programm, wo sich unter dem Baustein befindet.
	Dieser Baustein erzeugt eine Kopie der ausgewählten Figur. Die für die originale Figur programmierten Blöcke wirken dann auch beim Klon.
	Sobald ein Klon von der ausgewählten Figur entsteht, aktiviert sich dieser Ereignisbaustein und startet das Programm, wo sich unter dem Ereignisbaustein befindet.
	Löscht alle Klone der ausgewählten Figur. Kann nur am Ende eines Programmblocks gesetzt werden.
	Lässt die ausgewählte Figur in einer bestimmten Zeit zu einer Zufallsposition, zum Mauszeiger oder zu einer anderen Figur gleiten.

	Fügt visuelle Effekte zur ausgewählten Figur hinzu. Der Baustein „setze“ setzt den Effektwert auf einen fixen Wert während „ändere“ zum Effektwert addiert oder subtrahiert. Manche Effekte haben einen bestimmten Wertebereich. Falls z.B. ein maximaler Wert (100) überschritten wird, wird beim niedrigsten Wert weitergezählt (-100). Zum Ausschalten einer dieser Effekte setzt man den Wert auf 0 oder wenn man alle gleichzeitig ausschalten will, führt man den „schalte Grafikeffekte aus“ Baustein aus. • Farbe: ändert die Farbe der ausgewählten Figur. Wertebereich des Effektwerts liegt zwischen -100 und 100. • Fischauge: bei positivem Wert verzerrt sich das Aussehen der Figur von innen nach außen, bei negativem Wert von innen weiter nach innen • Wirbel: verzerrt die Figur bei positivem Wert gegen den Uhrzeigersinn und bei negativem Wert im Uhrzeigersinn • Pixel: reduziert bei höherem Wert die Auflösung der Figur • Mosaik: teilt die Figur gleichmäßig auf • Helligkeit: kann die Figur heller oder dunkler erscheinen lassen • Durchsichtigkeit: stellt die Transparenz der Figur in Prozent ein
	Schaltet alle Grafikeffekte der ausgewählten Figur wie Farbe, Fischauge, Wirbel usw. aus.
	Definiert, ob die ausgewählte Figur ziehbar oder nicht ziehbar ist.
	Dieser Baustein aus der Fühlen Kategorie zeigt den Wert der Mikrofonlautstärke an. Leicht verwechselbar mit dem Lautstärke Baustein aus der Kategorie Klang.



Bei diesem Baustein können verschiedene Werte wie Bühnenbildnummer, Bühnenbildname, Lautstärke der Bühne oder sonstige für die Bühne selbst definierte Variablen als Zahl oder als String angezeigt werden. Beim rechten Dropdown Menü dieses Bausteins kann man auch andere Figuren auswählen und deren Werte anzeigen lassen. Die Form des Bausteins zeigt auch, dass man diesen Baustein mit ihrem Zahlenwert auch in andere Rechenoperationen einfügen kann (siehe Bausteinkategorie Operatoren).

	Zeigt das Jahr, den Monat, das Datum (Tag), den Wochentag, die Stunde, die Minute oder die Sekunde des jetzigen Zeitpunkts an.
	Zählt die Tage seit dem Beginn des Jahres 2000 mit zwölf Nachkommastellen an.
	Falls man den Scratch Editor online im Webbrowser benutzt und angemeldet ist, kann der Benutzername angezeigt werden.

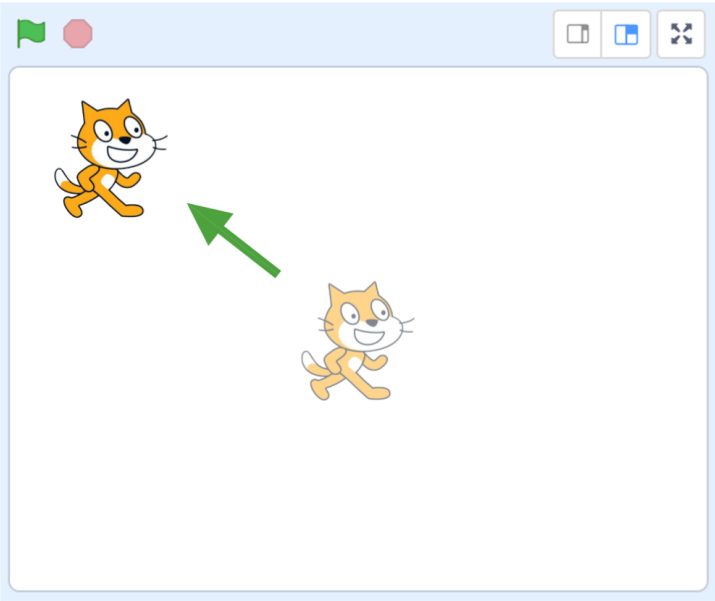
Kapitel 02 – Lösungen und Erklärungen zu den Aufgaben und Beispielen aus dem Aufgabenhandbuch

Im Folgenden Kapitel werden die Beispiele und die Aufgaben aus dem Aufgabenhandbuch ausführlich erklärt und in gelöster Form präsentiert.

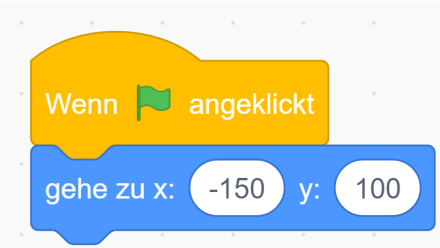
Beispiel 1

Erklärung:

Im ersten Beispiel wird dem Kind eine der drei Bewegungsarten vorgestellt, das Springen zu bestimmten Koordinaten. Die Figur soll in die linke obere Ecke gehen. Die richtigen Koordinaten sind in der Aufgabenstellung gegeben: x=-150 y=100. Diese werden in den „gehe zu x: y:“ Block eingegeben. Wird nun die grüne Flagge angeklickt springt die Figur sofort zu den Koordinaten. Der „gehe zu x: 0 y: 0“ Baustein kann verwendet werden, um zum Beispiel ein Spiel zurückzusetzen (Bei einem Rennen können alle Teilnehmer jederzeit mit einem Klick zur Startposition platziert werden). Falls Koordinaten nicht gegeben sind kann man die Katze auf eine gewünschte Stelle bewegen mit der Maus. Rechts unter der Bühne werden dann die Koordinaten angezeigt (Wenn die Figur ausgewählt ist). Diese Koordinaten können beispielsweise in den „gehe zu x: y:“ Block eingegeben werden.



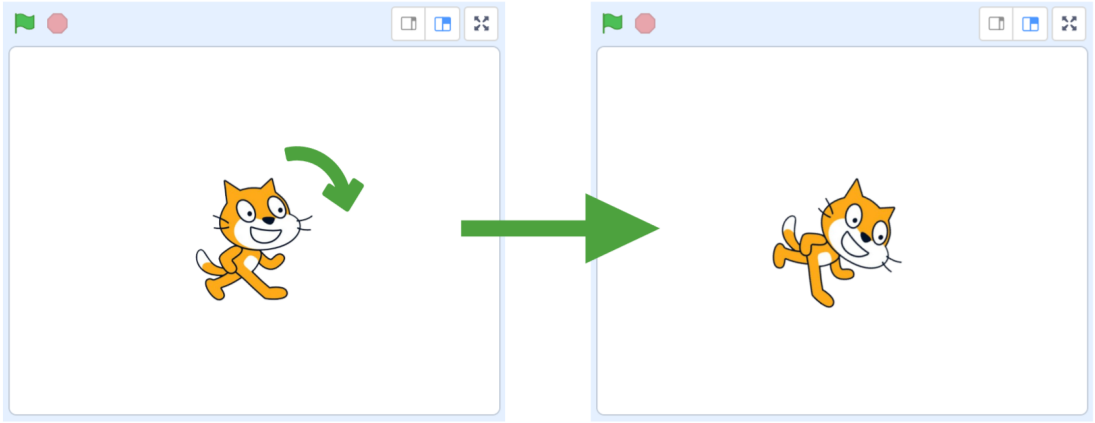
Lösung



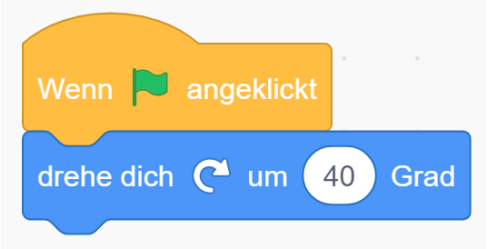
Beispiel 2

Erklärung:

Im Zweiten Beispiel lernt das Kind die Funktion des „drehe dich um 15 Grad“ Bausteins. Mit diesem Baustein kann man eine Figur im Uhrzeigersinn oder gegen den Uhrzeigersinn, um die eigene Achse drehen lassen (für beide Drehrichtungen gibt es einen eigenen Baustein). Der „drehe dich“ Baustein kann in Verbindung mit einem „wiederhole fortlaufend“ oder „wiederhole x-mal“ Baustein zu einer regelmäßigen Drehung führen, für Programme, bei denen sich etwas eine gewisse Zeit drehen soll.



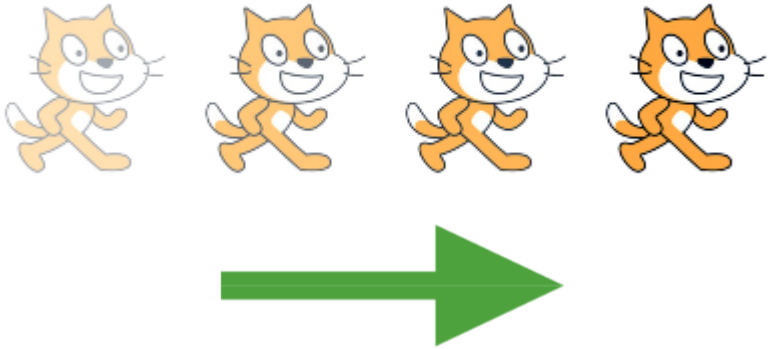
Lösung



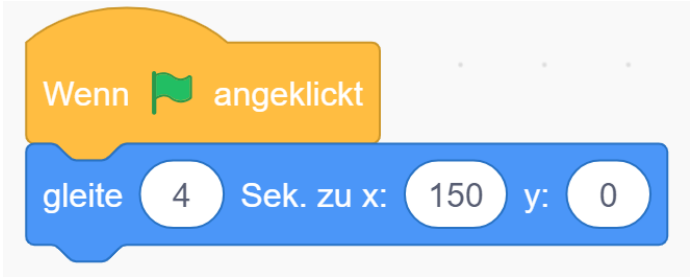
Beispiel 3

Erklärung:

Im dritten Beispiel lernt das Kind die zweite Art der Bewegung: das Gleiten. Wie beim „gehe zu x: y:“ Baustein kann man die gewünschten Koordinaten eingeben. Die Figur wird die Bewegung nicht sofort ausführen. Man kann bestimmen wie lange die Bewegung dauern soll.



Lösung

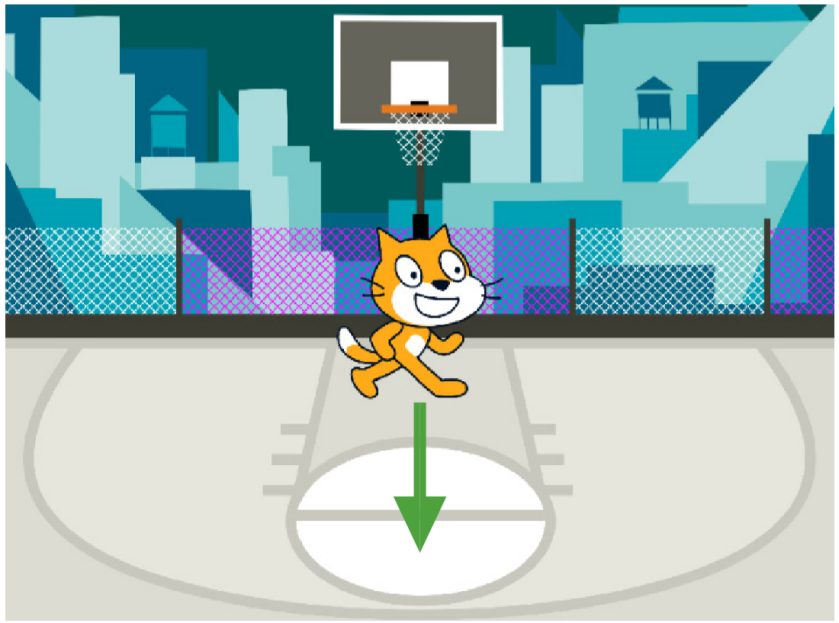


Beispiel 4

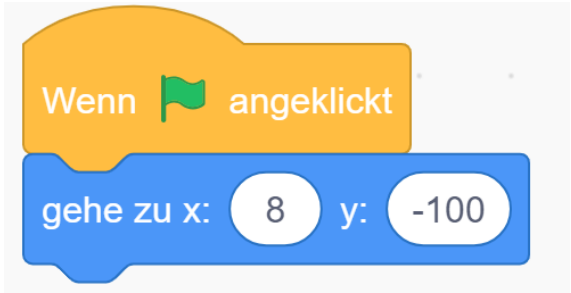
Erklärung:

In Beispiel 4 lernt man wie man das Bühnenbild ändert. Das Bühnenbild ist der Hintergrund eines Programmes, das kann ein Fußballfeld sein, das Weltall, eine Rennstrecke und vieles mehr. Mit einem schönen Hintergrund sieht ein Programm gleich besser aus, aber auch Figuren können mit dem Hintergrund interagieren. Man könnte zum Beispiel ein Programm erstellen, bei dem eine Figur etwas sagt, wenn sie Wasser (die Farbe Blau) im Hintergrund berührt.

Die nächste Herausforderung ist es, die richtigen Koordinaten des Kreises herauszufinden. Die Katze kann dafür mit der Maus zur gewünschten Position bewegt werden, dadurch ändern sich ihre Koordinaten, die man daraufhin in den „gehe zu x: y:“ Baustein einfügen kann.



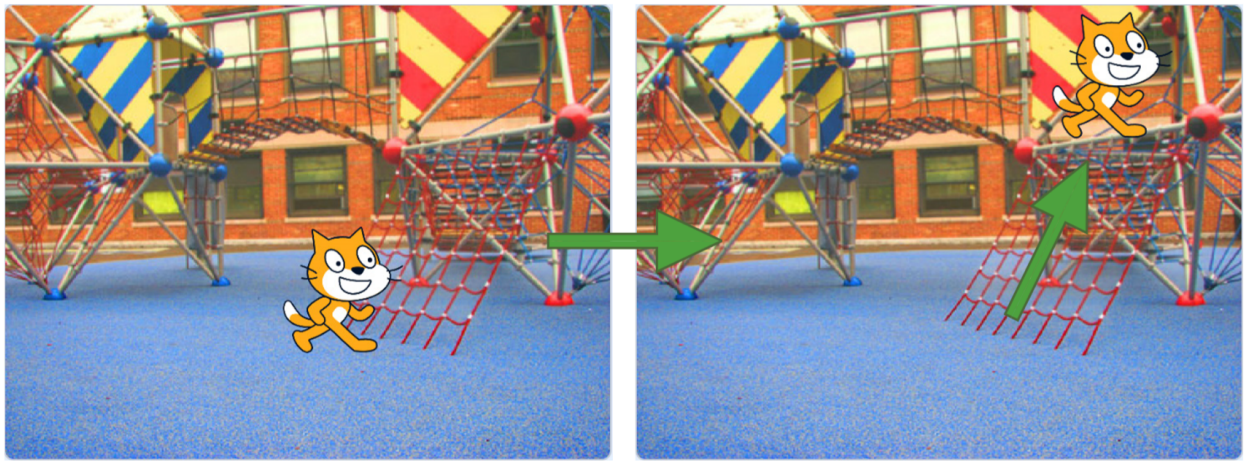
Lösung



Beispiel 5

Erklärung:

In diesem Beispiel werden das Gleiten und die Gehe zu x Bewegung kombiniert. Zuerst soll sich die Katze direkt an die Startposition bewegen, danach soll sie das Kletternetz in 3 Sekunden hochgleiten. Hier ist es sehr wichtig, dass das Kind den Unterschied zwischen den zwei Bewegungsarten kennt (gleiten und gehen). Die Katze soll auf dem Boden starten, also geht sie sofort dahin. Danach gleitet sie „langsam“ (3 Sekunden) das Kletternetz hinauf.

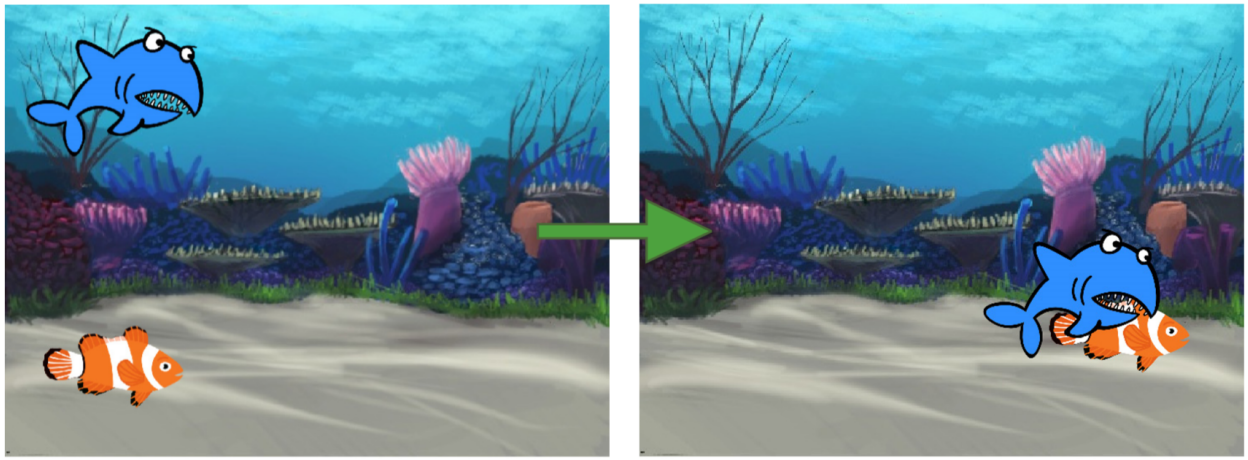


Lösung

Beispiel 6

Erklärung:

Hier werden zum ersten Mal zwei Figuren gleichzeitig verwendet. Um zwischen den beiden Figuren zu wechseln im Menü „Figuren“ auf die gewünschte Figur klicken. Beide Figuren müssen links im Bild starten (Startpunkt mit „gehe zu x: y:“ definieren) und danach sollen sie nach rechts gleiten, dass es so aussieht als ob sie schwimmen würden. Die Koordinaten und die gleit-dauer können in diesem Beispiel relativ frei gewählt werden.

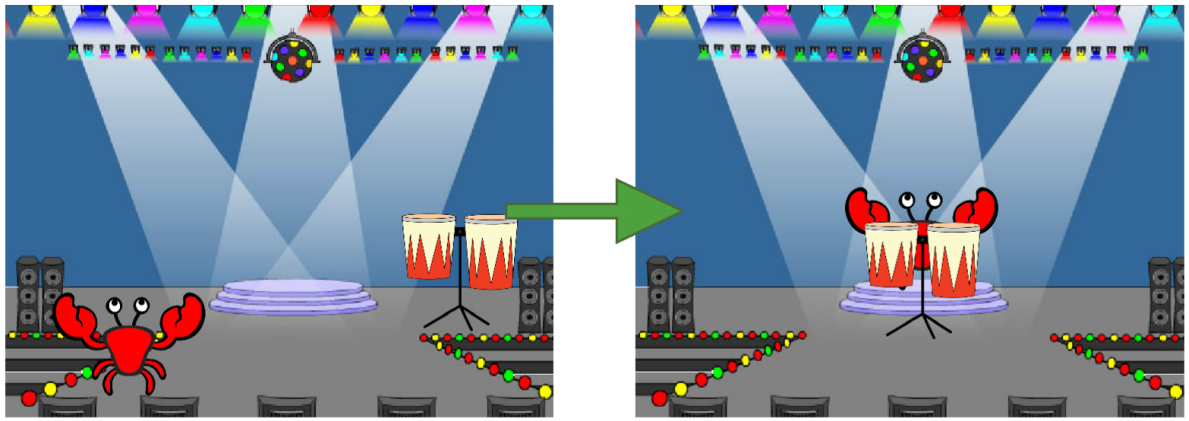


Lösung (zwei verschiede Figuren!)

Beispiel 7

Erklärung:

Im siebten Beispiel wird der Baustein „warte 1 Sekunden“ vorgestellt. Beiden Figuren wird zuerst ein Startpunkt gegeben mit dem „gehe zu x: y:“ Baustein. Danach soll die Krabbe zur Trommel gleiten (Baustein „gleite in 1 Sek. Zu x: y:) und sie „abholen“. Währenddessen wartet die Trommel 1 Sekunde mithilfe des neuen Bausteins „warte 1 Sekunden“. Der letzte Schritt ist für beide Figuren gleich, sie sollen beide in Mitte der Bühne gleiten (Baustein „gleite in 1 Sek. Zu x: y:). Tipp: Die Trommel wartet solange, wie die Krabbe braucht, um zur Trommel zu gleiten.



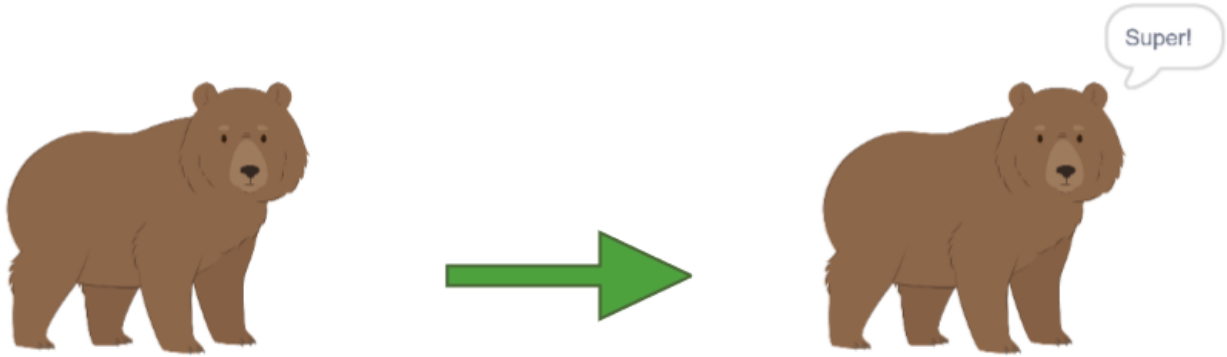
Lösung Krabbe

Lösung Trommel

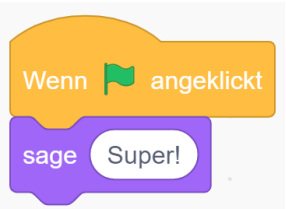
Beispiel 8

Erklärung:

Hier wird zum ersten Mal die Bausteinkategorie „Aussehen“ verwendet (bis jetzt gab es in den Beispielen nur „Ereignisse“ und „Bewegung“). Da der Baustein „sage“ nicht einen Ton macht, sondern nur eine Sprechblase, ist er in der Kategorie aussehen.



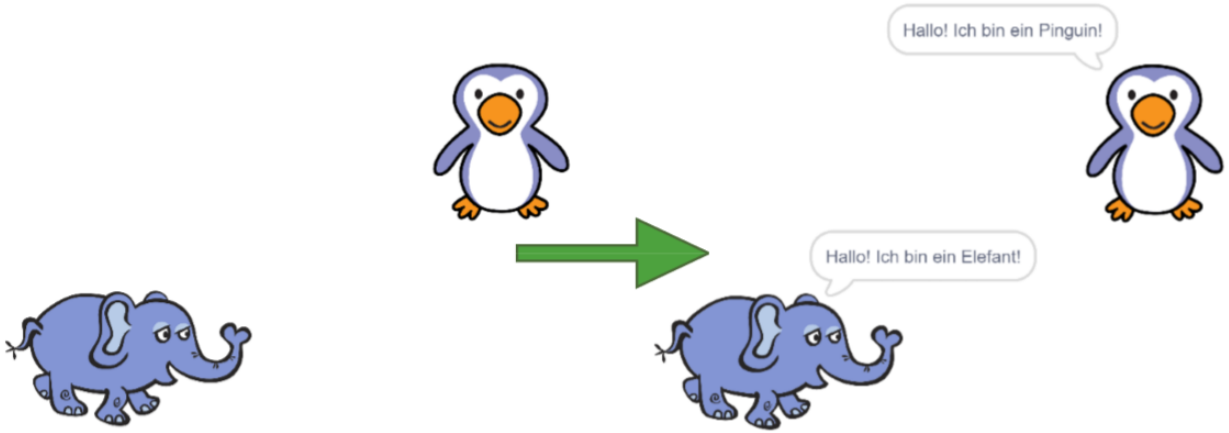
Lösung



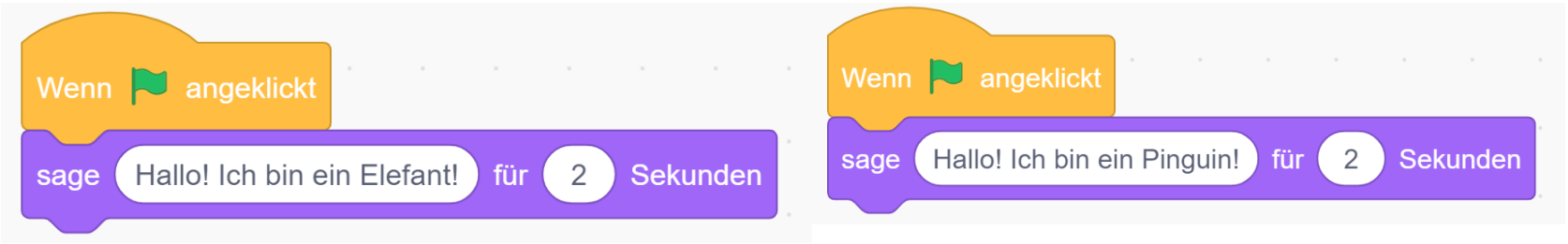
Beispiel 9

Erklärung:

Hier wird wieder die Bausteinkategorie „Aussehen“ verwendet, da beide Figuren etwas sagen sollen. Da der Baustein „sage“ nicht einen Ton macht, sondern nur eine Sprechblase, ist er in der Kategorie aussehen. Es wird auch eine leicht andere Version des „sage Hallo!“ Baustein verwendet, bei dem man bestimmen kann wie lange die Sprechblase erscheinen soll. In diesem Beispiel ist es wieder wichtig vor dem Programmieren einer Figur, im Menü für Figuren auf die richtige zu klicken, bevor man sie programmiert.



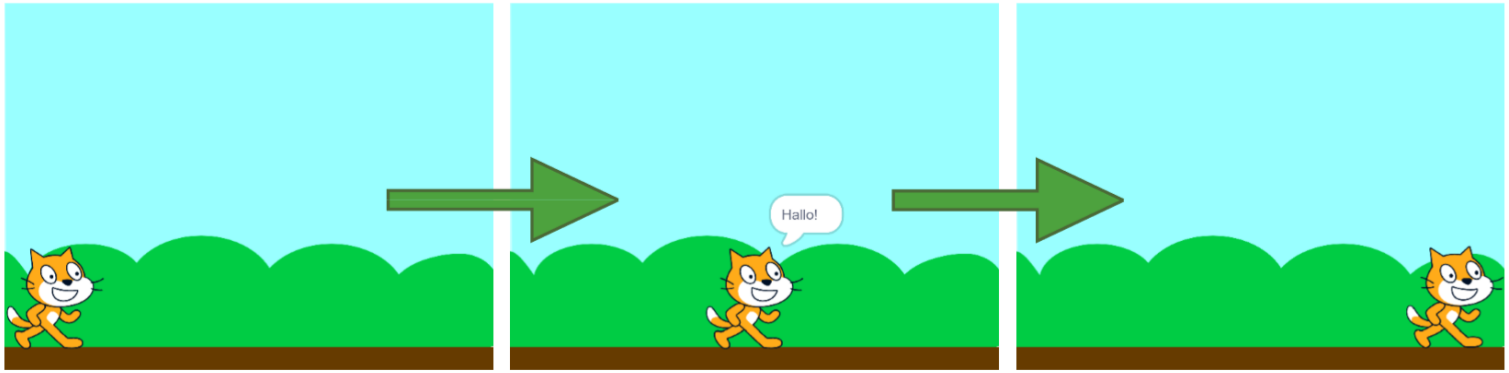
Lösung



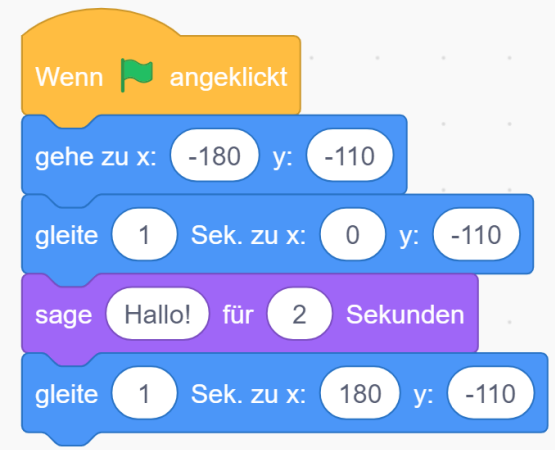
Beispiel 10

Erklärung:

In diesem Beispiel werden verschiedene Arten von Bewegungen aneinandergehängt. Der Startpunkt wird mithilfe des „gehe zu x: y:“ Bausteins definiert. Danach lauft die Katze mithilfe des „gleite zu x: 0 y: 0“ Bausteins in die Mitte des Bildes, um daraufhin 2 Sekunden lang „Hallo!“ zu sagen (Baustein „sage Hallo! Für 2 Sekunden“). Und am Ende soll die Katze nach rechts laufen („gleite zu x: 0 y: 0“). Wichtig ist, dass der Unterschied zwischen „gehe zu“ und „gleite“ verstanden worden ist.



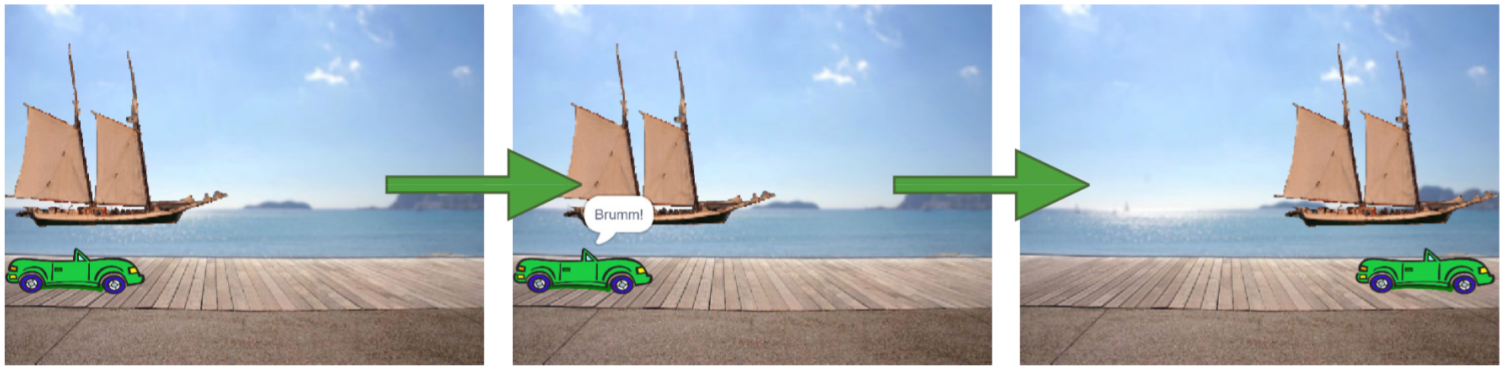
Lösung



Beispiel 11

Erklärung:

Um dieses kleine Rennen programmieren zu können, müssen die Kinder viele Dinge aus den vorigen Beispielen benutzen. Beide Figuren werden auf dieselbe Weise programmiert, der einzige Unterschied ist, dass das Auto als erstes 2 Sekunden „Brumm!“ sagt („sage Brumm! Für 2 Sekunden“ Baustein). Der Startpunkt beider Figuren ist links („gehe zu x: y:“ Baustein). Beide fahren/segeln mithilfe des „gleite in 1 Sek. Zu x: y:“ nach rechts zu ihrem Ziel. Damit das Rennen bis zum Schluss spannend bleibt, programmiert man das Schiff so, damit es 2 Sekunden länger braucht (das Auto startet dafür später, um „Brumm!“ zu sagen).



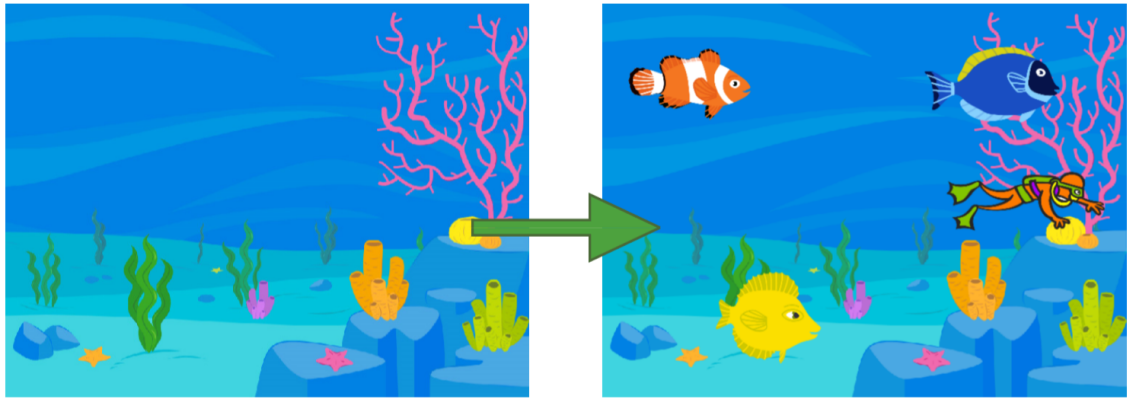
Lösung Auto

Lösung Boot

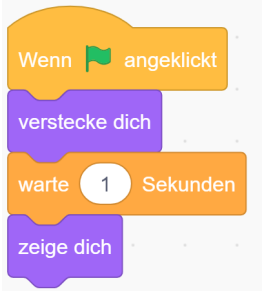
Beispiel 12

Erklärung:

In diesem Beispiel kann sich das Kind ein bisschen kreativ austoben. Für den Hintergrund wird eine Unterwasserwelt gewählt. Ein Taucher und ein paar Fische werden als Figuren auch gebraucht. Der „einzige“ Fisch in der Figuren-Bibliothek ist der Clownfisch. Diese Figur besitzt jedoch vier Kostüme, die alle andere Fische darstellen. Will man also beispielsweise einen Paletten-Doktorfisch, muss man zuerst einen Clownfisch auswählen und daraufhin im Menü für Kostüme auf das richtige Kostüm wechseln. Die eigentliche dieses Beispiels besteht darin, die Bausteine „verstecke dich“, „zeige dich“ und „warte 1 Sekunde“ kennenzulernen. Das Kind kann danach selbst entscheiden wie oft und wie lange sich die Figuren verstecken sollen.



Lösungsvorschläge



Beispiel 13

Erklärung:

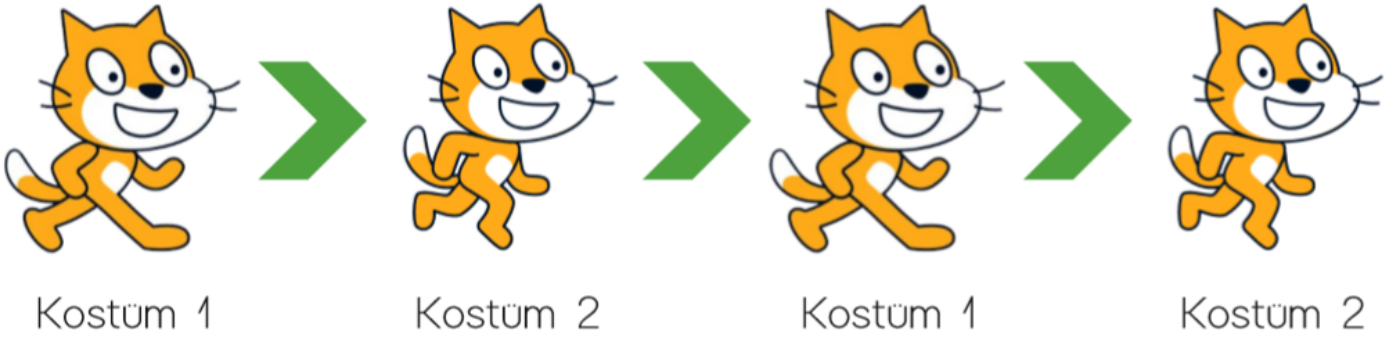
In diesem Beispiel werden 4 Figuren verwendet, mit einem Party-Hintergrund. Jede Figur soll nacheinander zur Party erscheinen. Das heißt jede Figur versteckt sich anfangs unterschiedlich lange. Für jede Figur werden unterschiedliche Zahlen beim „warte 1 Sekunden“ Baustein verwendet. Dieses Beispiel dient dazu eine bessere Orientierung zu bekommen, wenn man mehreren Figuren gleichzeitig arbeitet.

Lösung

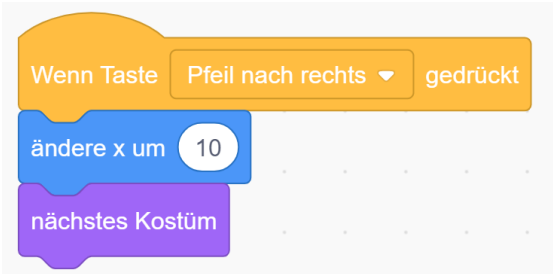
Beispiel 14

Erklärung:

In diesem Beispiel lernt man, wie man mithilfe von Kostümen, eine Lauf-Animation erstellen kann. Bei jedem Schritt wird das Kostüm geändert, damit die Katze den Anschein macht, Schritt für Schritt nach rechts zu gehen. Dabei wird auch zum ersten Mal ein neuer Ereignis-Baustein verwendet. Mit „Wenn Leertaste gedrückt“ kann man jede beliebige Taste verwenden, um etwas zu starten (Vorher hat man das Programm immer mit der grünen Flagge gestartet). Jeder Tastendruck soll das Kostüm wechseln und die X-Koordinaten der Figur um 10 ändern.



Lösung



Beispiel 15

Erklärung

Die Katze soll mithilfe der Pfeiltasten nach links und nach rechts laufen können („Wenn Taste X gedrückt wird“ und„ändere x um x“ Baustein). Bei jedem Schritt wird gleichzeitig auch das Kostüm gewechselt, damit man sieht wie die Katze Schritte macht („nächstes Kostüm“ Baustein).

Lösung

Beispiel 16

Erklärung:

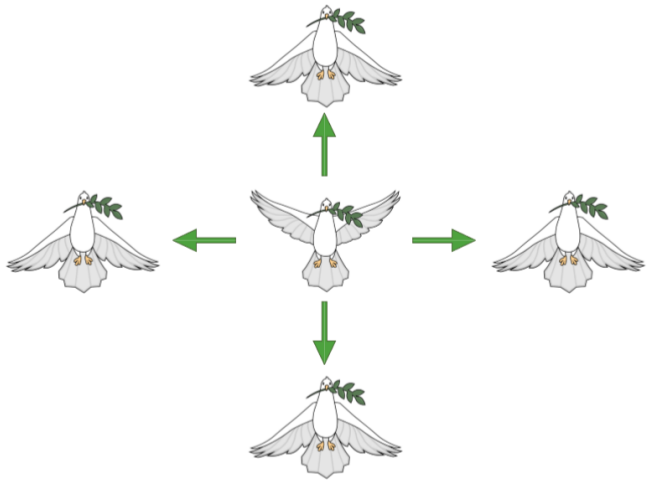
Der Vogel soll nach unten und oben fliegen können. Statt einem Schritt, macht der Vogel jedes Mal, wenn das Kostüm gewechselt wird ein Flügelschlag. Um den Vogel zu steuern werden die Bausteine „Wenn Taste X gedrückt“ und „ändere y um X“ verwendet. Für jeden Flügelschlag wird der Baustein „nächstes Kostüm verwendet.

Lösung

Beispiel 17

Erklärung:

In diesem Beispiel werden Beispiel 15 und 16 vereint. Mit den Pfeiltasten soll man nach oben, unten, links und rechts fliegen können. Fliegen kann man mit den Bausteinen „Wenn Taste X gedrückt wird“, „ändere x um X“ und „ändere y um X“. Derr Flügelschlag wird mit „nächstes Kostüm“ programmiert.

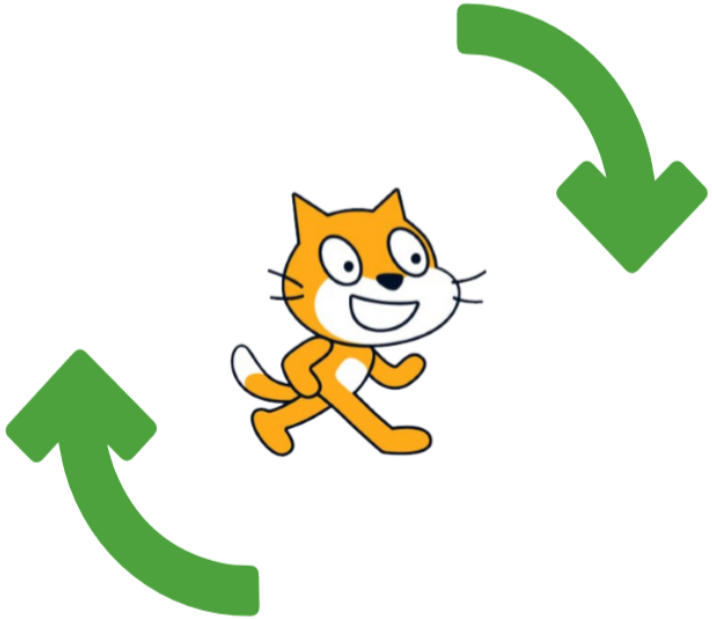


Lösung

Beispiel 18

Erklärung:

Hier wird der Baustein „wiederhole fortlaufend“ vorgestellt. Alles was sich in diesem Baustein befindet wiederholt sich, bis es von etwas anderem gestoppt wird, beispielsweise der Baustein „stoppe alles“ oder in diesem Fall der rote Knopf neben der grünen Flagge, mit dem man immer das Programm stoppen kann.

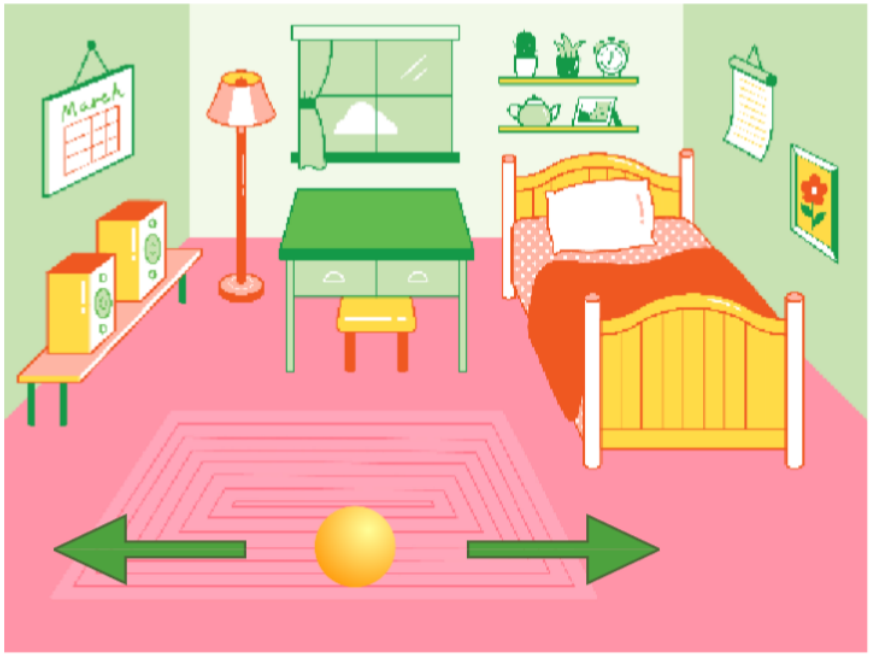


Lösung

Beispiel 19

Erklärung:

In diesem Beispiel soll der Ball fortlaufend einen 10er Schritt machen („gehe 10er Schritt“). Damit er nicht aus dem Bild rollt, benutzt man den neuen Baustein „pralle vom Rand ab“.

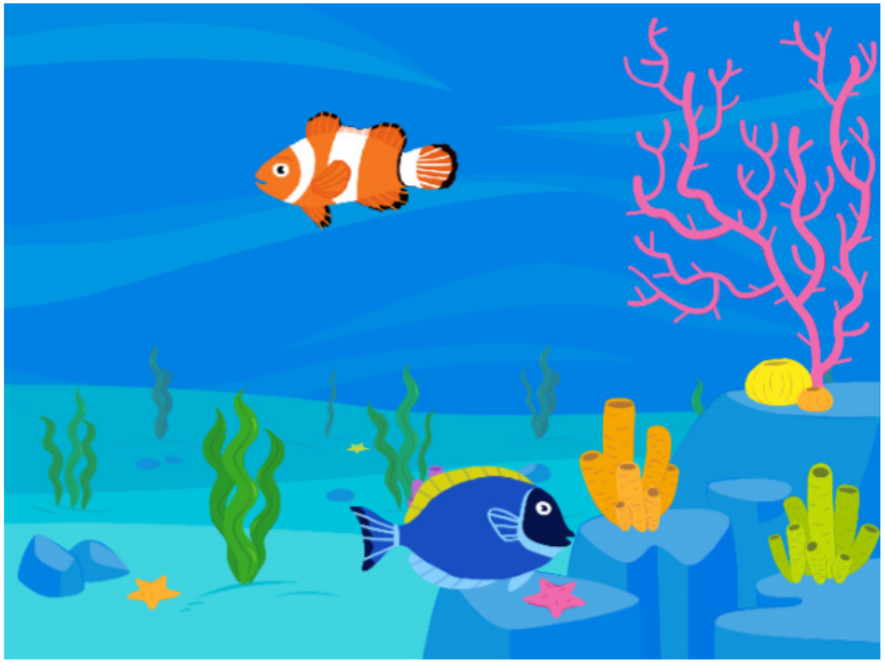


Lösung

Beispiel 20

Erklärung:

In diesem Beispiel wird man die Fische selbst spiegeln, um sie hin und her schwimmen zu lassen. Beide Fische werden gleich programmiert, nur Koordinaten unterscheiden sich. Beiden Fischen gibt man einen Startpunkt (mit „gehe zu x: y:“). Um sicher zu gehen das der Fisch zu Beginn in die richtige Richtung schaut, wechselt man zum ersten Kostüm (das Kostüm, das nach rechts gerichtet ist). Mit dem „wiederhole fortlaufend“ und den beiden „gleite 1 Sek. Zu x: y:“ Bausteinen programmiert man das die Fische von links nach rechts schwimmen und dann wieder von rechts nach links. Bevor sich ein Fisch wieder umdreht und zurückschwimmt, muss er zuerst sein Kostüm wechseln. Ein zweites Kostüm muss selbst gemacht werden indem man es dupliziert und spiegelt.



Lösung

1. *Aufgabe

In der ersten Aufgabe sieht das Kind wie Bausteine richtig „zusammengesteckt“ werden. Die richtigen Bausteine müssen ausgewählt werden und sie müssen richtig zusammengefügt werden, dass die Katze kleine Schritte machen kann.

2. Aufgabe

Hier wird zum ersten Mal eine Figur programmiert, die ganz klassisch wie in einem Computerspiel steuerbar ist. Wenn sich bei den Koordinaten den X-Wert um 10 erhöht, bewegt sich die Katze um 10 Pixel nach rechts. Wenn der X-Wert sich um 10 verringert, bewegt sich die Katze 10 Pixel nach links. Das soll man mit den zwei Pfeiltasten programmieren.

Koordinaten

Die Bühne, auf der sich die Figuren in Scratch bewegen können, ist wie ein großes Schachbrett, bei dem Jeder Pixel bestimmte Koordinaten hat. Es gibt insgesamt 480 Pixel auf der X-Achse und 360 Pixel auf der Y-Achse. Das heißt, die obere Rechte Ecke hat die Koordinaten (240|180), und die untere, linke Ecke (-240|-180). Wenn sich eine Figur mit dem „gehe zu x: 0 y: 0“ Baustein fortbewegt, wird immer der Mittelpunkt der Figur verwendet.

Kostüme

Kostüme in Scratch können vieles sein. Eine Figur kann zum Beispiel 2 Kostüme haben, eines in der sie steht und eines während sie sitzt. Eine Figur kann aber auch ein Kostüm haben in der sie den rechten Fuß vorne hat, und beim anderen Kostüm hat sie den linken Fuß vorne. Wenn man bei dieser Figur, bei jedem Schritt, das Kostüm wechselt, sieht sie so aus, als würde sie gehen.

3. Aufgabe

Hier wird der „wiederhole X mal“ Baustein vorgestellt. Die Katze soll sich 36-mal um 10 Grad drehen, weil man nicht 36 „drehe dich um 10 Grad“ Bausteine will, nimmt man einen „wiederhole X mal“ Baustein und einen „drehe dich um X Grad“ Baustein.

Oftmals wenn man in Scratch ein Spiel programmieren will, müssen Figuren „fühlen“ können. Man kann zum Beispiel die Ziellinie rot machen, und falls eine Figur rot berührt, weiß diese Figur das sie im Ziel ist.

1. Aufgabe

In dieser Aufgabe muss die Katze fühlen, wann sie im Ziel ist. Diese Aufgabe ist auch die Basis für ein Rennen, das auf Seite 59 programmiert wird. Es soll programmiert werden, dass man 10er Schritte machen kann, bis die Katze das rote Ziel berührt. Dann soll sie Ziel! Sagen.

Kapitel 03 – Grundlagen der Programmierung

Die Grundlagen der Programmierung bieten weiterführendes Wissen über die Programmierung an. Für besonderes Interessierte Eltern und Volksschullehrer / Volksschullehrerinnen wird in diesem Kapitel die Textbasierte Programmierung angeschnitten.

If Abfrage:

If Abfragen kann man in Scratch spielerisch mit den Bausteinen „falls < > dann“ und „falls < > dann, sonst“ erlernen. Mit diesen Bausteinen kann man Programmieren was eine Figur machen soll falls etwas Bestimmtes passiert. Beispiel: Falls die Katze etwas Rotes berührt, soll sie sich um 180° drehen. Im Grunde genau gleich wie bei einer Textbasierten Programmiersprache. Aus „If“ und „else“ werden hier „falls“ und „sonst“.

Schleife:

Eine Schleife in der Programmierung wiederholt eine Reihe von Befehlen. In Scratch kann man seine eigenen Schleifen so definieren das sie sich entweder für immer wiederholen oder man gibt an wie oft sich die Schleife wiederholt. „Wiederhole 10-mal“ und „Wiederhole fortlaufend“ sind die zwei Blöcke dafür.

Algorithmus:

Ein Algorithmus ist eine Handelsvorschrift zur Lösung eines Problems. Das heißt, dass es ein Problem gibt, und es werden klare Regeln aufgestellt wie dieses Problem in einzelnen Schritten gelöst wird. Ein relativ einfacher Algorithmus wäre zum Beispiel, wenn man in einem Labyrinth ist biegt man immer rechts ab und irgendwann wird man das ende des Labyrinthes erreichen. Natürlich gibt es für den Menschen effektivere Lösungen aus einem Labyrinth zu entkommen, das ist die Kunst beim Programmieren, die einfachste, effektivste Lösung für ein Problem zu finden.

Debuggen:

Bugs sind in der Informatik Fehler im Programmcode. Das Debuggen ist das entfernen dieser Bugs. Beim Arbeiten mit Programmiersprachen, wird es immer Fehler geben, egal ob ein Semikolon[1] vergessen wird oder ein Baustein in Scratch. Solche Bugs können ganz einfach entfernt werden, man muss sie nur finden. Solche Fehler werden Kinder beim lernen mit Scratch immer wieder machen, und das ist gut so denn man lernt aus ihnen.

Koordinatensystem:

Bei Kindern im Volksschulalter werden die meisten noch nicht wissen was ein Koordinatensystem ist. In Scratch kann man sich auf einem Koordinatensystem (X-Achse von -240 bis 240, Y-Achse von -180 bis 180) bewegen. Falls die Katze 10 Pixel nach rechts gehen soll, kann man den Block „ändere x um 10“ verwenden.

Objektorientierte Programmierung:

Einige Grundlagen der Objektorientierten Programmierung werden die Kinder durch dieses Handbuch auch lernen.

- Objekt

  Objekte sind in Scratch die Figuren. Sie können Eigenschaften und Methoden haben

[1] Strichpunkt

- Eigenschaften

  Eigenschaften der Objekte (Figuren) wären zum Beispiel Farbe und Größe

- Methoden

  Methoden der Objekte wären zum Beispiel „gehe 10er Schritt“ und „sage Hello!“

Kapitel 04 – Hilfestellungen

Dieses Kapitel befasst sich mit Erklärungen und Funktionen von Scratch.

Sie sollen dem Benutzer / Der Benutzerin dieses Handbuchs dabei helfen, sich in Scratch zurecht zu finden.

Das Kapitel beinhaltet:

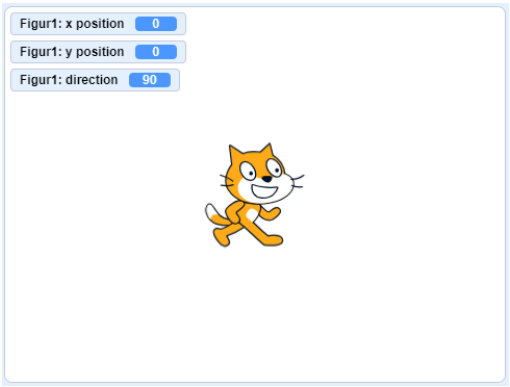
- Variablen
- Kostümfunktion
- Welche Figur ist ausgewählt
- Bühnenbilder
- Was sind Ebenen
- Boolesche Werte

Variablen

Man kann sich Variablen wie Platzhalter für Werte, die laufend überschrieben werden können, vorstellen.

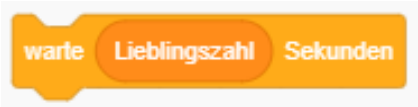
Mit einem Klick auf das Häkchen direkt neben den Variablen können diese auf der Bühne ein- oder ausgeblendet werden. So kann man unter anderem Auskunft über den aktuellen Wert der Variable bekommen.

Variablen, wie „x-Position“, „y-Position“, „Richtung“ oder „Größe“, übermitteln Live Werte einer Figur. So kann man stets mit aktuellen Werten arbeiten. Sollten sich diese Werte verändern, so werden auch die Werte der Variablen verändert.



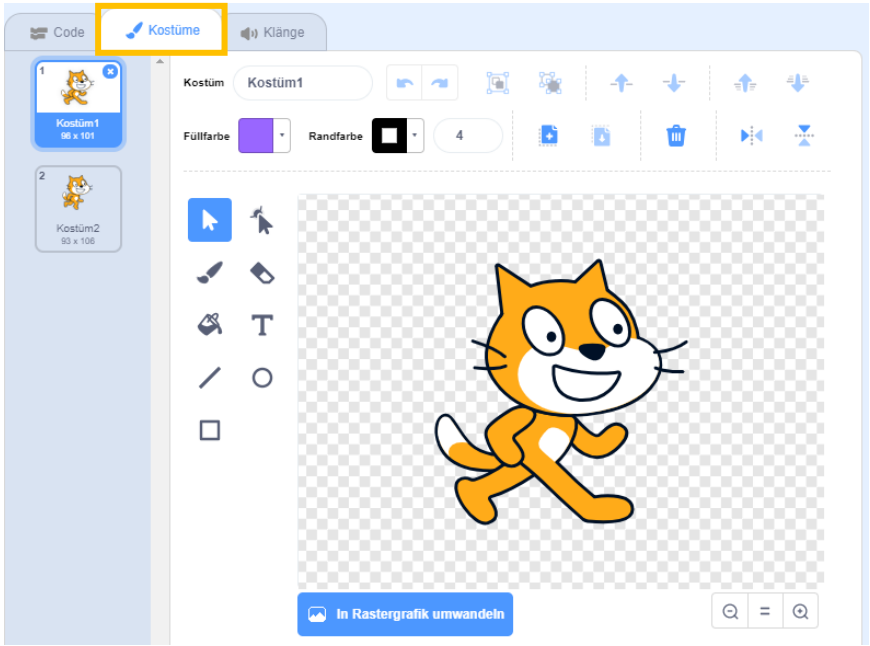
Variablen können auch selbst erstellt werden. Diese, selbsterstellten, Variablen sind dann grundsätzlich leer. Ihnen können jedoch Werte zugewiesen werden, mit denen dann gearbeitet werden kann.

Variablen können in Scratch per *Drag and Drop* auch in andere Bausteine eingesetzt werden. Der Baustein, in den die Variable eingesetzt wurde, arbeitet dann automatisch mit dem Wert der Variable.



Kostüm Funktion

Die Kostümfunktion bietet eine Reihe unterschiedlicher Möglichkeiten an. Um die Optionen dieser Funktion nutzen zu können, klickt man im Reiter auf „Kostüme“. Sollte dieser Reiter nicht vorhanden sein, muss man zuerst überprüfen, ob man überhaupt eine Figur ausgewählt hat.

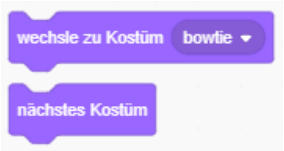


Hier können Figuren bearbeitet werden. Es kann ihre Farbe verändert werden, die Größe und die Form angepasst werden. Des Weiteren kann hier gezeichnet werden. Man wählt dazu den Pinsel aus und eine Farbe, die einem gefällt. Dann legt man los!

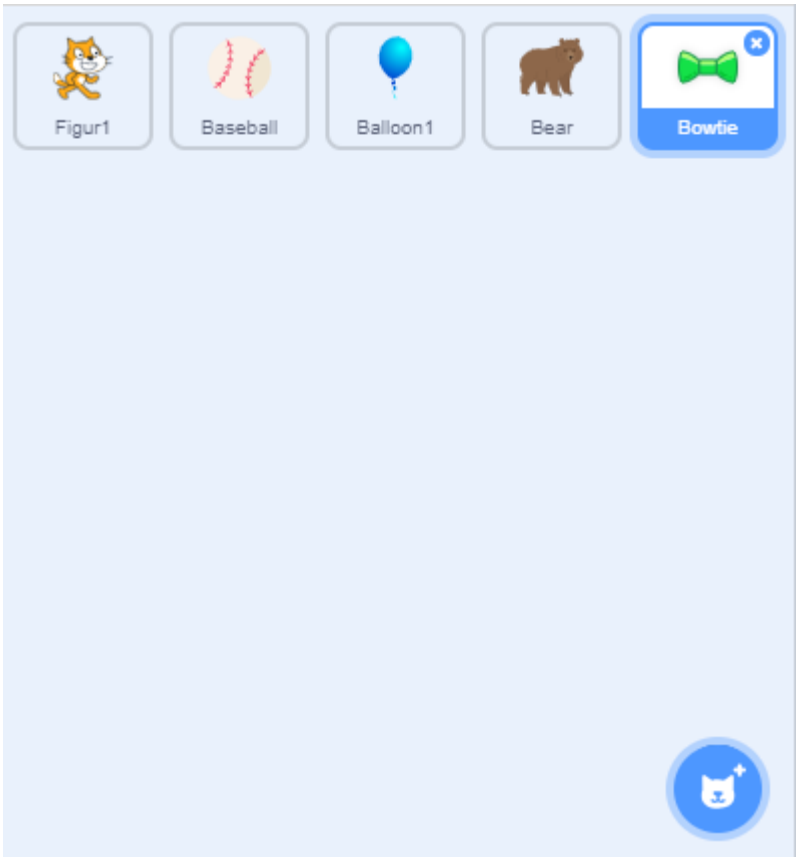
Figuren können gespiegelt (umgedreht) werden:

Neben den ganzen grafischen Optionen können auch Animationen erzeugt werden. Scratch kann zum Beispiel mit Hilfe seines Kostüms eine Laufanimation hervorrufen.

Diese beiden Bausteine werden dir helfen, mit der Kostümfunktion umzugehen:



Welche Figur ist ausgewählt?



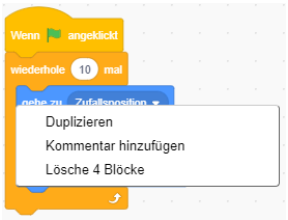
Welche Figur aktuell ausgewählt ist, sieht man direkt unter der Bühne. Die blau umrandete Figur ist die aktuell ausgewählte Figur.

Man muss beachten, dass man immer nur die ausgewählte Figur bearbeiten oder programmieren kann!

<u>Hier eine kleine Hilfestellung im Umgang mit mehreren Figuren:</u>

Wenn man nach einiger Zeit bemerkt, dass man eine falsche Figur programmiert hat, muss man nicht von vorne beginnen.

Durch einen Rechtsklick mit der Maus öffnet sich das folgende Menü:

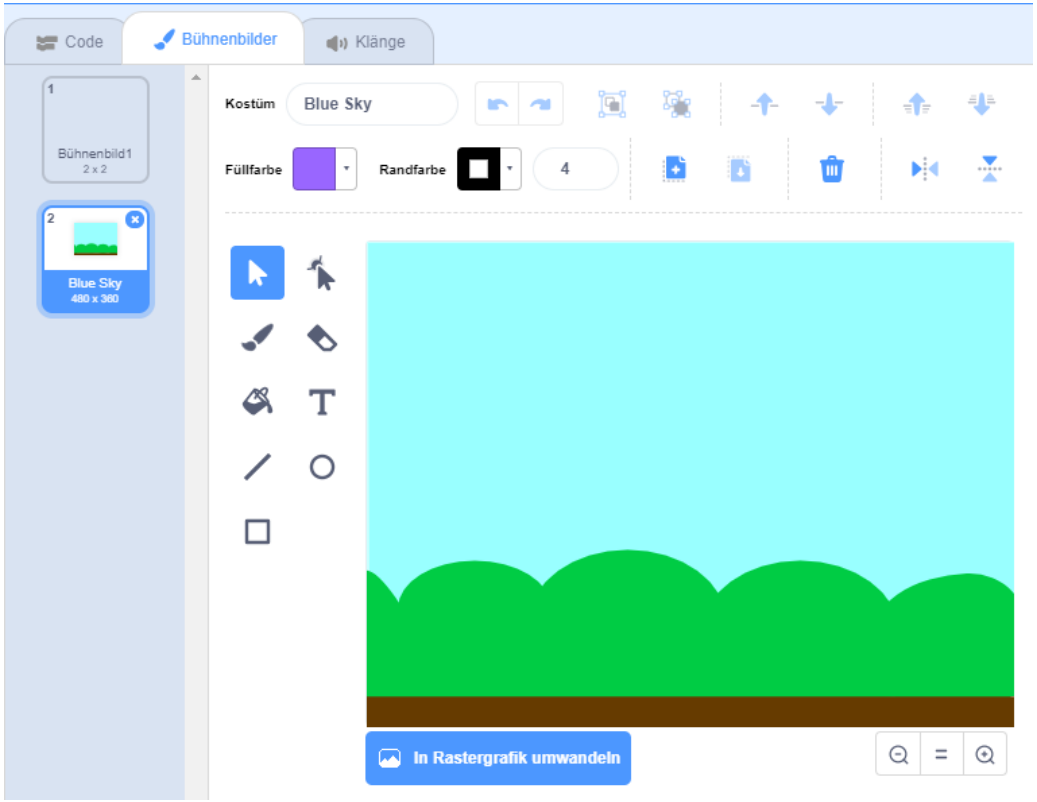


Man wählt „Duplizieren“ und klickt auf die Figur, auf die man den Code programmieren wollte. Jetzt befindet sich der Code genau dort, wo er sein sollte.

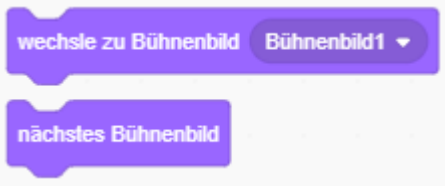
Bühnenbilder

Um diese Option überhaupt öffnen zu können, muss man sicher gehen, dass man die Bühnenbilder ausgewählt hat.

Ähnlich wie du Kostümfunktion kann man auch die Bühnenbilder bemalen, drehen und einfärben wie man es braucht. Es kann Text hinzugefügt werden oder der Name des Bildes verändert werden.



Wie man sieht, können mehrere Bühnenbilder zur selben Zeit geöffnet sein. Sofern mehr als ein Bühnenbild geöffnet ist, kann die Figur mit Hilfe dieser Bausteine zwischen diesen Bühnenbildern bewegen:

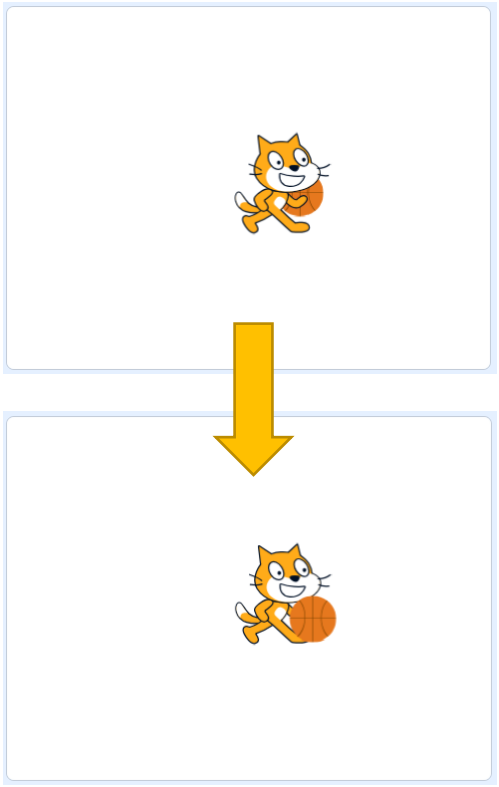


Beispielsweise kann so die Animation erzeugt werden, dass eine Figur den Raum wechselt oder sich die Jahreszeiten verändern.

Was sind Ebenen?

Die unterste Ebene ist immer das Bühnenbild! Ebenen kommen nur dann zum Einsatz, wenn mehr als eine Figur verwendet wird.

Man kann sich Ebenen wie die einzelnen Schichten eines Kuchens vorstellen.



Scratch soll mit dem Ball spielen.

Jedoch ist der Ball auf einer tieferen Ebene wie Scratch selbst. So ist der Ball kaum zu erkennen und die Erstellung einer Animation oder eines Spiels macht wenig Sinn.

Legt man den Ball nun auf eine höhere Ebene, so ist er klar und deutlich zu erkennen. Scratch kann so problemlos mit dem Ball spielen.

Diese Bausteine helfen dir im Umgang mit Ebenen:

Boolesche Werte

Boolesche Werte können nur zwei Zustände haben.

Entweder „Wahr“ oder „Falsch“. In den meisten Fällen sind Boolesche Werte, Rückgabewerte von Berechnungen oder Ähnlichem.

In Scratch findet man diese Art von Rückgabewerten zumeist in der Bausteinkategorie „Operatoren“.

Hier ein Beispiel:

Diese Abfrage überprüft, ob das Wort „Apfel“ den Buchstaben „A“ enthält. Da dem so ist, ist die Aussage wahr und der Rückgabewert lautet „true“.

Diese Abfrage überprüft, ob das Wort „Apfel“ den Buchstaben „H“ enthält. Da dem nicht so ist, ist die Aussage falsch und der Rückgabewert lautet „false“.

Es gibt keine andere Antwort auf diese Art von Abfrage. Das heißt, man kann sich immer darauf verlassen, dass man eine von beiden Antworten als Ergebnis zurückbekommt.

Träger

Vorarlberg unser Land

WKO Wirtschaftskammer Vorarlberg

CODE 4 KIDS .AT

Bildungspartner

PH Vorarlberg

Bildungsdirektion Vorarlberg

HTL Dornbirn

Partner

DORNBIRN

Marktgemeinde Lustenau

HOHENEMS inspiriert!