Scratch,
dein Weg zum Programmieren

Erstelle deine eigenen Geschichten, Spiele und Animationen!

1. Edition November 2025 | Sven Waser | © iSTEP Verein

Inhaltsverzeichnis

Über dieses Skript

Willkommen zu deinem Self-Study-Tutorial für Scratch.

Dieser Leitfaden richtet sich an Personen, die keine Erfahrung mit Programmieren haben und sich mit dieser Kunst vertraut machen möchten. Wenn du neugierig bist und immer etwas Neues lernen möchten, ist dies das richtige Skript für dich.

Dieses Skript ist als Selbstlernleitfaden konzipiert, dessen Bearbeitung etwa 12 Stunden beansprucht. Du kannst das Skript in deinem eigenen Tempo und zu jeder beliebigen Zeit bearbeiten. Alle Übungen enthalten Lösungen, aber schau dir die Lösungen nicht an, bevor du nicht selbst ein funktionierendes Beispiel erstellt hast. Wenn deine Lösung nicht mit der Lösung des Autors übereinstimmt, mach dir keine Sorgen. Das Faszinierende an der Programmierung ist, dass es für ein einziges Problem tausende von Lösungen gibt. Wenn deine Lösung das Problem löst, ist deine Lösung perfekt! Wenn du dir bezüglich deiner Lösung unsicher bist oder eine Frage hast, dann wende dich sich bitte an den Betreuer. Die Kontaktdaten des Betreuers findest du weiter unten.

Am Ende jedes Abschnitts des Skripts findet sich ein separater Bereich, in dem du alle Fragen notieren kannst, die du beispielsweise in einer Lektion stellen möchtest. Plane dir genügend Zeit ein, um dieses Skript durchzuarbeiten und schreibe dir so viele Fragen wie möglich auf, damit Lektionen möglichst spannend und lehrreich werden.

Alle Lösungsdateien und die Datei für das Programmierprogramm Scratch sind in einem Paket zusammengefasst. Wenn du das gesamte Paket noch nicht heruntergeladen hast, tue dies jetzt, indem du diesem Link folgst: https://www.sven-waser.ch/scratch oder den QR-Code auf der rechten Seite scannst.

Jedes Kapitel wird mit Kontrollaufgaben beendet. Für jede Aufgabe existiert eine Lösungsdatei deren Dateiname im Paket aus der Kapitelnummer, gefolgt von einem Unterstrich und einer zweistelligen Übungsnummer besteht. Die Lösung für die erste Übung in Kapitel 1 hat beispielsweise den Dateinamen „1_01.sb3“.

Als Student weiss ich, dass es schwierig ist, sich alles zu merken, was in den vorherigen Kapiteln erklärt wurde. Deshalb habe ich den Anhang I erstellt, der eine Spickzettel mit den wichtigsten Funktionen der Scratch-Sprache enthält. Die Verwendung dieser Seite beim Lösen der Aufgaben oder während der Lektionen wird dir das Leben erheblich erleichtern und dir viel Zeit sparen.

Viel Spass beim Programmieren!

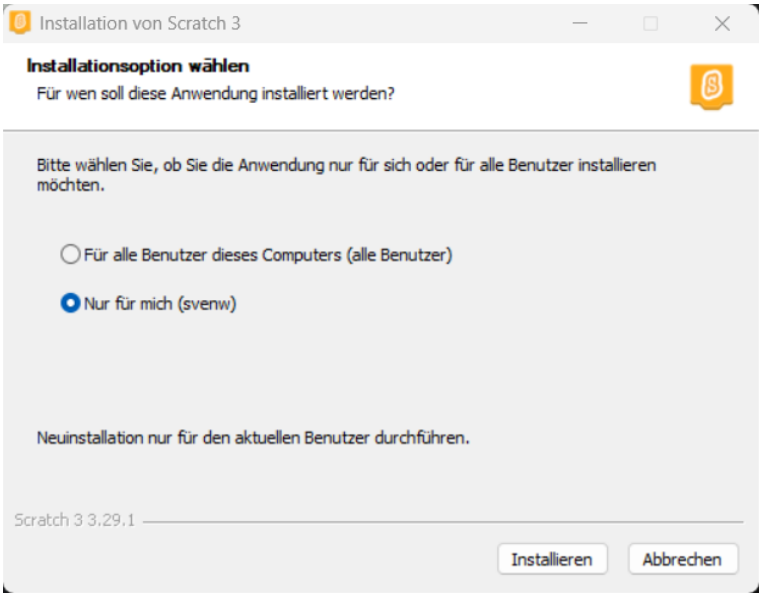
Kontaktinformationen:
Name: Sven Waser
E-Mail: sven.waser@istep.ch

***Hinweis**: Dieses Skript wurde im Frühjahr 2024 für den iSTEP Verein entwickelt. Alle Dokumente wurden mit der Scratch Desktop Software Version 3.29.1 für Windows erstellt.*

1 Installiere Scratch

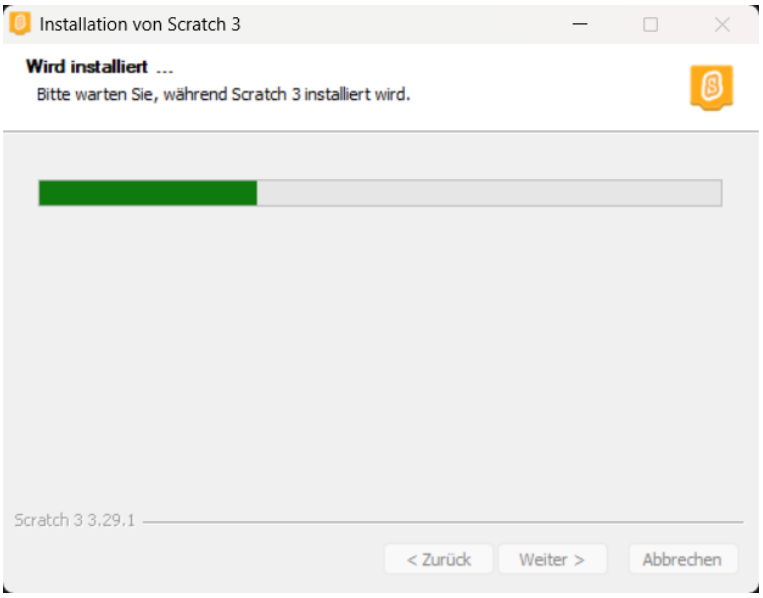
Lass uns mit Scratch beginnen. Bevor du beginnen kannst, musst du das Scratch-Programm installieren. Wenn du Scratch bereits auf deinem Computer installiert hast, kannst du dieses Kapitel überspringen und mit Kapitel 2 „Einführung in Scratch“ fortfahren. Wenn du Scratch nicht installieren kannst, kannst du auf die Online-Version von Scratch unter https://scratch.mit.edu/projects/editor/ zurückgreifen und mit Kapitel 2 „Einführung in Scratch“ weiterfahren, andernfalls folge den nachfolgenden Anweisungen zur Installation von Scratch. Du hast noch nie zuvor ein Programm installiert? Keine Sorge, es ist nicht so kompliziert, und es gibt immer ein erstes Mal. Diese Anleitung hilft dir dabei, alle notwendigen Schritte durchzuführen.

1. Um die Installation zu starten, musst du den Ordner mit dem Tutorial-Paket öffnen. Wenn du das Paket noch nicht heruntergeladen hast, findest du in der Einleitung einen Link und einen QR-Code, der zur Download-Seite führt.
2. Suche nun das Dokument “Scratch 3.29.1 Setup.exe” doppelklicke auf das Dokument, was die Dialogbox für die Installation öffnet,
3. Falls du dieses Fenster siehst, hast du den komplexen Teil der Installation überstanden.



Installationsfenster, in dem nach dem Installationsort gefragt wird.

4. Standardmässig sollte das Optionsfeld “Nur für mich” ausgewählt sein, andernfalls, wähle es aus. Die Information in den Klammern steht für deinen Benutzername, der für jeden der das Tutorial durchführt, unterschiedlich ist. In diesem Fall lautet der Benutzername svenw.
5. Nachdem du die richtige Option ausgewählt hast, klicke auf “Installieren”. Daraufhin sollte das Installationsfenster mit dem Fortschrittsbalken angezeigt werden.



Installationsfenster bei der Installation von Scratch

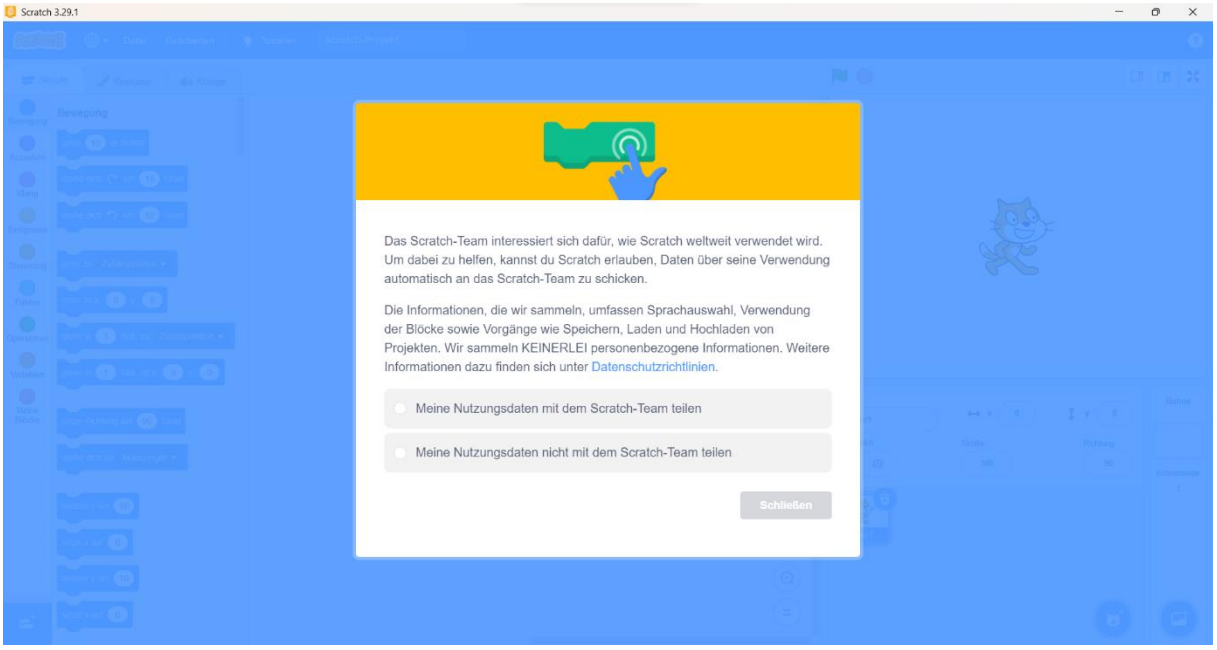
6. Warte, bis der Fortschrittsbalken vollständig gefüllt ist, und klicke auf den Knopf “Weiter”, sobald dieser Verfügbar ist.
7. Du hast es geschafft. Klicke nun auf “Fertigstellen”, um die Installation abzuschliessen und das Scratch Programm zu starten.

Installationsfenster bei erfolgreicher Installation

Herzlichen Glückwunsch, du hast Scratch erfolgreich installiert. Auf deinem Desktop findest du ein Symbol, mit dem du den Scratch-Editor erneut öffnen kannst. Lösche dieses Symbol nicht, da es sonst schwierig wird, den Scratch-Editor erneut zu öffnen. Aber keine Sorge, du kannst deinen Computer jederzeit neu starten oder herunterfahren oder andere Programme verwenden.

1.1 Scratch zum ersten Mal verwenden

Nachdem du auf „Finish“ geklickt hast, startet den Scratch-Editor. Wenn du Scratch zum ersten Mal verwendest, wirst du nach der Verwendung deiner Nutzungsdaten gefragt. Wenn du damit einverstanden bist, deine Daten mit dem Scratch-Team zu teilen, darunter die Sprachauswahl, die Verwendung von Blöcken und einige weitere nicht personenbezogene Daten, dann wähle die erste Option. Wenn nicht, wähle die zweite Option. Wenn du mehr über die Datenschutzrichtlinie von Scratch erfahren möchtest, klicken auf den vorhandenen Link. Wenn du sich nicht sicher bist, welche Option du wählen sollst, wählen die zweite Option.



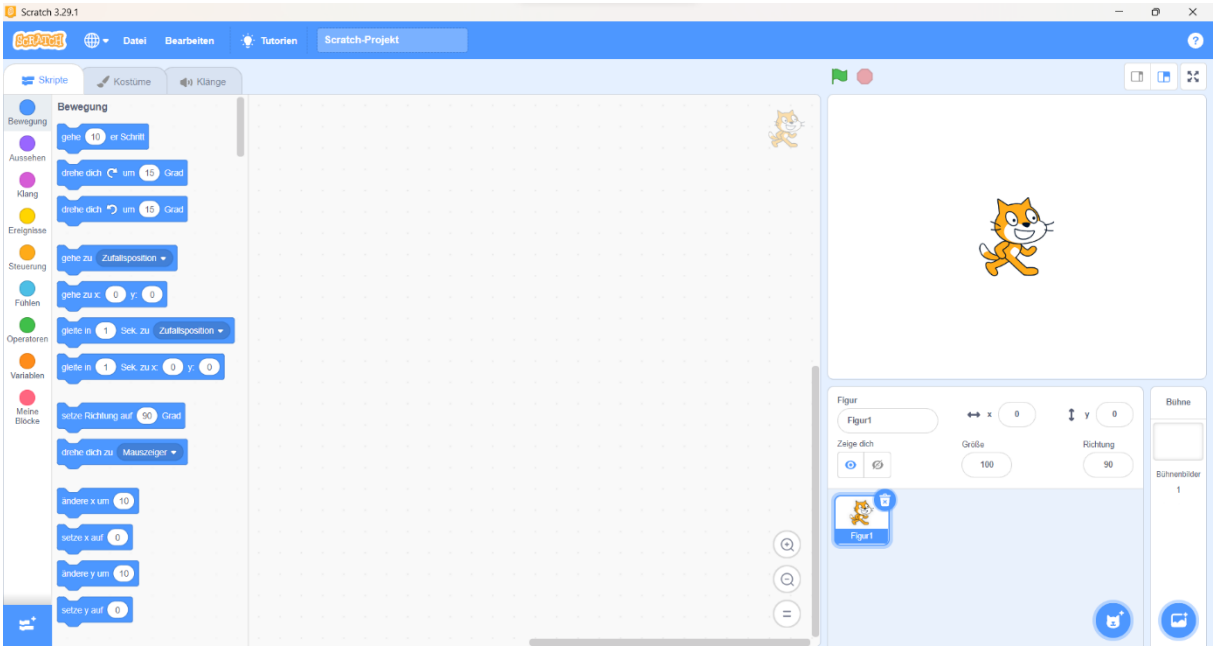
Scratch bittet um die Erlaubnis zur Erhebung anonymisierter Daten.

Falls du Probleme bei der Installation von Scratch hast, dann wende ich bitte an den Betreuer. Die Kontaktdaten findest du in der Einleitung. Wenn du Fragen zum erfolgreichen Installationsprozess hast, dann schreib diese in das dafür vorgesehene Feld und fahre mit dem nächsten Kapitel fort.

1.2 Notizen

2 Einführung in Scratch

In diesem Kapitel lernst du die verschiedenen Programmabschnitte des Scratch-Programmiereditors kennen sowie eine Einführung in einige neue Konzepte erhalten. Verwende in den kommenden Erklärungen das untenstehende Bild, um den jeweiligen Abschnitt zu finden, oder öffne deinen Scratch-Editor. Lass uns gemeinsam die Welt von Scratch Entdecken!



Übersicht über den Scratch-Editor.

2.1 Kopfzeile

Die Kopfzeile des Editors befindet sich oben im Fenster und enthält die Projekteinstellungen. Du musst dir nicht alle diese Funktionen und ihre Position merken, aber es ist wichtig zu wissen, dass es sie gibt. Ein Programmierer folgt dem Prinzip, zu wissen, wo er das Wissen finden kann.

Die Funktionen der Kopfzeile sind unten von links nach rechts aufgelistet.

Sprachauswahl: Nach dem Scratch-Logo befindet sich ein Symbol, welches einem Globus mit Längen- und Breitengraden ähnelt. Klicke auf dieses Symbol, um die Sprache auszuwählen.

Neu, Speichern und Laden: Das Element “Datei” bietet folgende Operationen:
***Neu*:** Ersetzt dein aktuelles Projekt mit einem neuen leeren Projekt
***Von deinem Computer hochladen*:** Öffnet eine Dialogbox zum Importieren abgespeicherter Projekte. Diese Option kann verwendet werden, um die eigene Lösung mit der Lösung des Autors zu vergleichen.
***Auf deinem Computer speichern*:** Öffnet eine Dialogbox, um den Speicherort deines Projekts zu bestimmen. Bevor du ein Projekt Speicher kannst, musst du diesem einen Namen geben, mehr dazu weiter unten.

Bearbeiten*:* Das Element “Bearbeiten” bietet Operationen zum Wiederherstellen von Änderungen oder das Aktivieren des Turbo-Modus.
Wiederherstellen: Stellt auf den Ausganszustand vor der letzten Änderung zurück.
Turbo-Modus einschalten: Schaltet den Turbo-Modus ein oder aus. Wir werden den Turbo-Modus nicht verwenden. Fals du dich für den

Turbo-Modus interessierst findest du die Dokumentation dazu online im Scratch-Wiki (https://de.scratch-wiki.info/wiki/Turbo-Modus).

Tutorials: Der Punkt „Tutorial“ bietet eine Reihe einfacher Tutorials, um sich mit Scratch vertraut zu machen. Probieren einige davon aus, nachdem du dieses Skript bearbeitet hast.

Projektname: Der Punkt „Scratch-Projekt“ ist ein Eingabefeld, in das du den Namen deines Projekts eintragen kannst.

Über die Software: Das Fragezeichen-Symbol auf der rechten Seite bietet Hilfe zu Scratch.
About: Öffnet ein Fenster mit den Version-Details deines Scratch-Programms. Wenn die Version 3.29.1 installiert ist, hast du die richtige Version installiert.
Privacy Policy: Öffnet ein Fenster mit der Datenschutzerklärung von Scratch
Data Settings. Öffnet ein Dialogfeld, in dem du die Dateneinstellung ändern kannst, welche du bei der ersten Verwendung von Scratch vorgenommen hast.

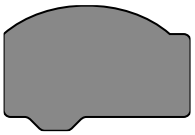
2.2 Blockpalette

Lass uns nun zum Abschnitt „Blockpalette“ gehen, der sich auf der linken Seite der Scratch-Software befindet. Wie bereits erwähnt, ist Scratch eine blockbasierte Programmiersprache. Das bedeutet, dass jeder Befehl zum Ausführen eines Programms einen eigenen Block hat. Scratch verfügt über 6 verschiedene Blocktypen, die sich durch ihre Form unterscheiden. Um die Blockcodierungstechnik zu verstehen, ist es wichtig, die Unterschiede zwischen den Blocktypen zu kennen und zu wissen, wo welcher Blocktyp verwendet wird.

2.2.1 Blockformen

2.2.1.1 Kopf-Blöcke

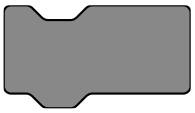
Ein Kopfblock befindet sich immer am Anfang eines Skripts. Alle Befehle eines Skripts müssen sich unter einem Kopfblock befinden, was mit deinem Körper verglichen werden kann. Es ist nicht möglich, Blöcke über dem Kopfblock hinzuzufügen. Scratch kennt 6 Hutblöcke, von denen du die meisten in den nächsten Kapiteln kennenlernen wirst.



Kopf-Block

2.2.1.2 Stapel-Blöcke

Der Stapelblock ist der am häufigsten verwendete Block. In Scratch gibt es über 60 Stapelblöcke. Wie der Name schon sagt, kann der Stapelblock unter oder über einen Block gelegt werden, der eine Erhebung oder Vertiefung aufweist. Der Stapelblock enthält die Magie der Programmierung, indem er einen bestimmten Befehl ausführt.



Stapel-Block

2.2.1.3 Wahrheits-Blöcke

Der Wahrheitsblock ist ein Bedingungsblock, der entweder wahr oder falsch darstellt (Beispiel: Ist heute dein Geburtstag? Entweder wahr oder falsch). Der Wahrheitsblock sieht aus wie eine längliche Sechseckform und kann in Löcher mit derselben Form eingesetzt werden. Ein Wahrheitsblock kann nicht als Stapelblock verwendet werden und muss in einen anderen Block eingesetzt werden.

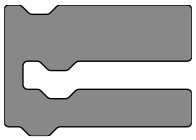
Wahrheits-Block

2.2.1.4 Wert-Blöcke

Der Werteblock repräsentiert einen Wert. Die Werteblöcke werden für Berechnungen (Addition, Subtraktion, Multiplikation und Division) und Variablenwerte während der Laufzeit verwendet. Der Werteblock hat abgerundete Ecken und kann in Löcher mit derselben Form eingefügt werden.

Wert-Block

2.2.1.5 Klammer-Blöcke



Klammer-Block

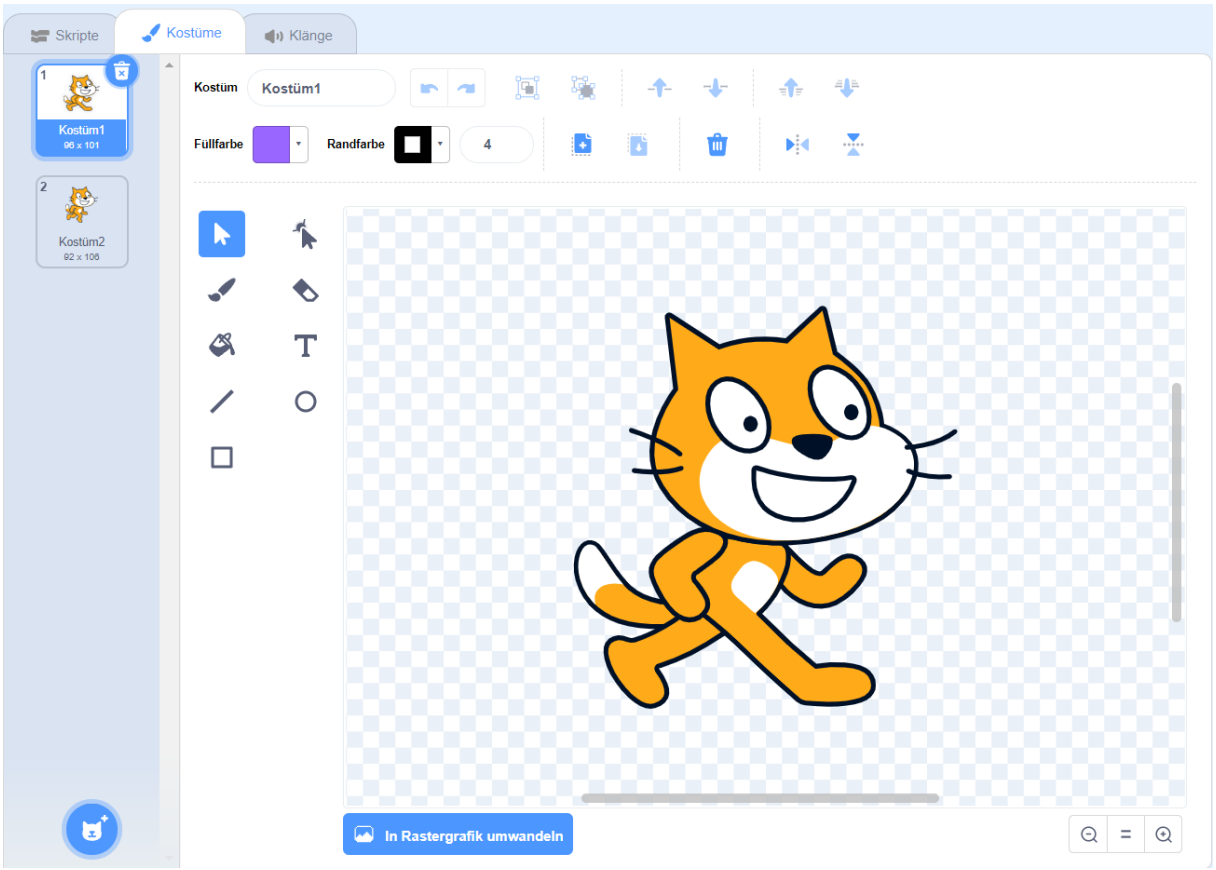
Klammer-Blöcke sind wahrscheinlich die wichtigsten Blöcke in Scratch. Klammer-Blöcke werden für Schleifen und Bedingungen verwendet. Das erforderliche Skript für eine Schleife oder die Bedingung ist im „Mund“ des Klammer-Blocks enthalten.

2.2.1.6 Abschluss-Blöcke

Abschluss-Block

Abschluss-Blöcke werden kaum verwendet, bieten jedoch einige Sonderfunktionen, die wir im Kapitel „Ereignisse und Steuerelemente” näher betrachten werden. Abschluss-Blöcke befinden sich am Ende von Skripten, wo nichts angehängt werden kann. Sie sind im Grunde das Gegenteil von Kop-Blöcken.

2.3 Malprogramm



Scratchs eingebauter Maleditor

Das Malprogramm unterscheidet Scratch von vielen anderen Programmier-Editoren und bietet grundlegende Bildbearbeitungsoptionen. Dieser Editor wird für Kostüme, die Darstellung Ihrer Figuren und Hintergründe verwendet.

Der Editor unterstützt Vektorgrafiken und Bitmap-Grafiken, die in das jeweils andere Format konvertiert werden können. Der Unterschied zwischen diesen beiden Formaten ist einfach. Stelle dir einen roten Kreis in der Mitte des Bildes vor. Bei einer Vektorgrafik speicherst du die Form, die Position und die Farbe der Figur. Bei einer Bitmap-Grafik speicherst du für jedes Pixel (Quadrat auf deinem Bildschirm) dessen Farbe. Wenn du alle Pixel in der richtigen Reihenfolge in ein Raster einfügst, erhältst du das Bild. Vektorgrafiken werden für Zeichnungen verwendet, während Bitmap-Grafiken für Fotos verwendet werden. Wenn du eine neue Figur, Kostüm oder einen neuen Hintergrund erstellen, überlege dir immer, welches Grafikformat am besten geeignet ist. Wichtig ist jedoch, dass du kein falsches Grafikformat wählen kannst, sondern nur ein passenderes oder unpassenderes.

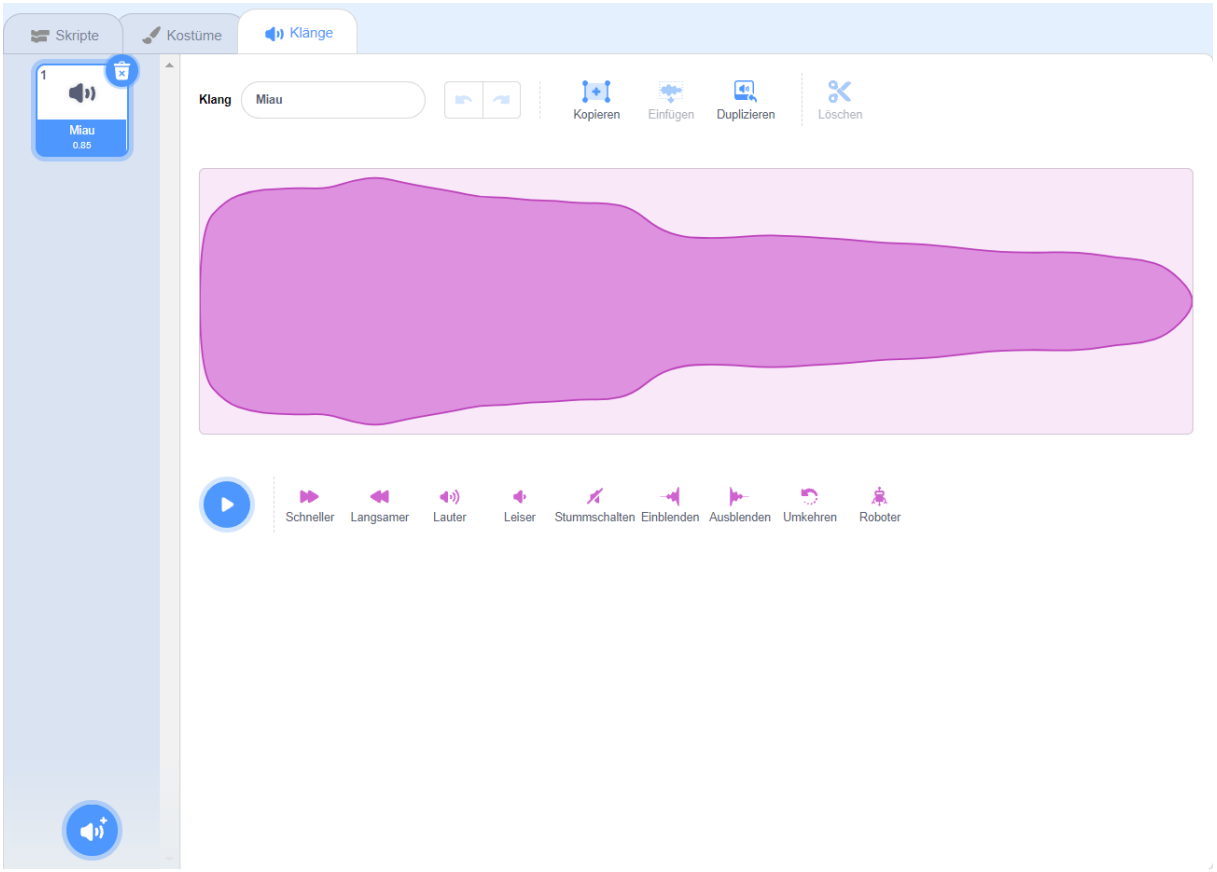
Im linken Navigationsbereich des Editors kannst du das Kostüm auswählen, dass du bearbeiten möchtest. Ein oder mehrere Kostüme ergeben eine Figur. Wenn du ein Kostüm hinzufügen möchtest, musst du den Mauszeiger über das Katzen-Symbol am unteren Rand des Navigationsbereiches bewegen. Du kannst nun ein Bild als Kostüm hochladen oder ein Kostüm von Grund auf neu zeichnen. Wenn du ein vordefiniertes Kostüm verwenden möchten, musst du auf die Lupe klicken. Wenn du auf das Überraschungssymbol klickst, wird ein zufälliges Kostüm aus den vordefinierten Kostümen ausgewählt. Um ein Kostüm zu löschen, wählst du das Kostüm aus und klickst auf den Papierkorb in der oberen rechten Ecke. Wenn du dich mit Kostümen noch nicht so gut auskennst, mache dir keine Sorgen. In Kapitel 7 werden wir uns Kostüme und Hintergründe, die ebenfalls im Malprogramm bearbeitet werden können, genauer ansehen.

Wähle ein Kostüm

Es gibt viele Bildbearbeitungsoptionen, auf die wir hier nicht näher eingehen werden. Wenn du eine dieser Optionen verwenden möchtest, kannst du mit der Maus über das Symbol fahren, um eine QuickInfo anzuzeigen, oder sie im Scratch-Wiki nachschlagen.

Dies ist nur eine kurze Beschreibung des Malprogramms. Wenn du dich für dieses Thema interessierst, findest du ausführliche Informationen im Scratch-Wiki (https://de.scratch-wiki.info/wiki/Malprogramm). Aber verbringe nicht zu viel Zeit mit dem Lesen der Dokumentation, denn du lernest durch Ausprobieren.

2.4 Klang Editor



Scratchs eingebauter Klang Editor

Der Klang-Editor kann durch Klicken auf die Registerkarte neben der Registerkarte „Kostüm“ geöffnet werden. Der Editor bietet Optionen zum Bearbeiten und Remixen deiner Klänge. Der Editor ist in einen Klangbereich auf der linken Seite, in dem du die zu bearbeitende Audiodatei auswählen kannst, und den Bearbeitungsbereich in der Mitte, in dem du die Audiodatei remixen und bearbeiten kannst, unterteilt.

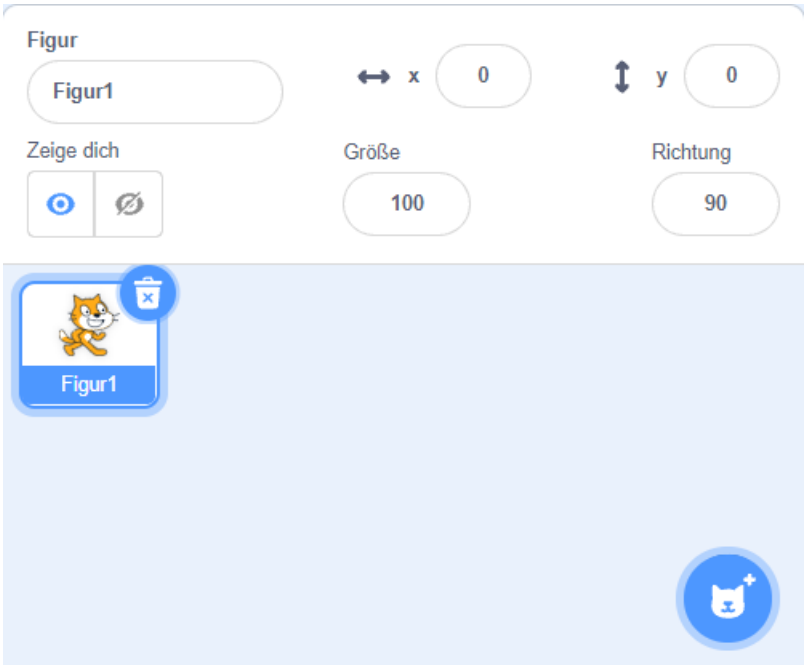
Wozu kann ein Klang-Editor bei der Entwicklung eines Spiels verwendet werden? Fast jedes Spiel hat Hintergrundmusik und spezielle Sounds, die beim Klicken auf eine Schaltfläche abgespielt werden. Mit dem Klang-Editor kannst du deine Sounds einfach und schnell modifizieren, um ein einzigartiges und unverwechselbares Spielerlebnis zu schaffen.

Wie im Malprogramm kannst du deine eigenen Audiodateien importieren, laden oder aufnehmen, indem du auf die Lautsprechertaste unten im Soundfenster klickst. Scratch bietet eine Reihe von Klängen, die in den meisten Fällen passend sind. Um einen Klang zu löschen, wähle ihn aus und klicke rechts oben auf das Papierkorb-Symbol.

Wir werden den Klang-Editor kaum verwenden. Wenn du dich damit vertraut machen möchtest, besuch das Scratch-Wiki (https://de.scratch-wiki.info/wiki/Klangeditor), wo alle Funktionen ausführlich beschrieben sind.

2.5 Figureneigenschaften

Die Figureneigenschaften enthalten alle Figuren. Figuren sind Charakteren welche in dem Spiel, der Story oder der Animation verwendet werden.



Figureneigenschaften

Wie du sehen kannst, kannst du deinen Figuren einen Namen geben. Wähle jeweils einen aussagekräftigen Namen, der die Funktion der Figur beschreibt. Wenn du beispielsweise eine Katze hast, die sich von links nach rechts bewegt, wäre „Katze bewegt sich von links nach rechts“ ein guter Name. In den Figureneigenschaften kannst du auch die Standardposition der Figur mit der x- und y-Achse festlegen. Wir werden uns das Koordinatensystem später genauer ansehen. Du kannst zudem festlegen, ob die Figur auf der Bühne sichtbar sein soll oder nicht, und du kannst die Größe der Figur sowie die Richtung, in die es zeigt, definieren. Weitere Details hierzu findest du im nächsten Kapitel.

Um eine Figur zu erstellen, klicke auf das Katzen-Symbol unten rechts im Fenster und wähle die gewünschte Option aus, z. B. das Hinzufügen eines Kostüms. Wenn du auf die Figur klickst und dann zur Registerkarte „Kostüme“ wechselst, kannst du alle Kostüme der Figur anzeigen sowie Kostüme hinzufügen, entfernen oder ändern.

Klicke auf den Papierkorb in der oberen rechten Ecke der **Figur**-Darstellung, um d**ie Figur** einschließlich aller Kostüme und Skripte zu entfernen.

2.6 Bühne

Die Bühne befindet sich in der oberen rechten Ecke der Scratch-Oberfläche und zeigt das grafische Ergebnis deiner Skripte an. Mit der grünen Flagge kannst du dein Programm starten und mit dem roten Knopf beenden. In der oberen rechten Ecke kannst du zudem die Größe der Bühne bestimmen (klein, groß (Standard) oder Vollbild).

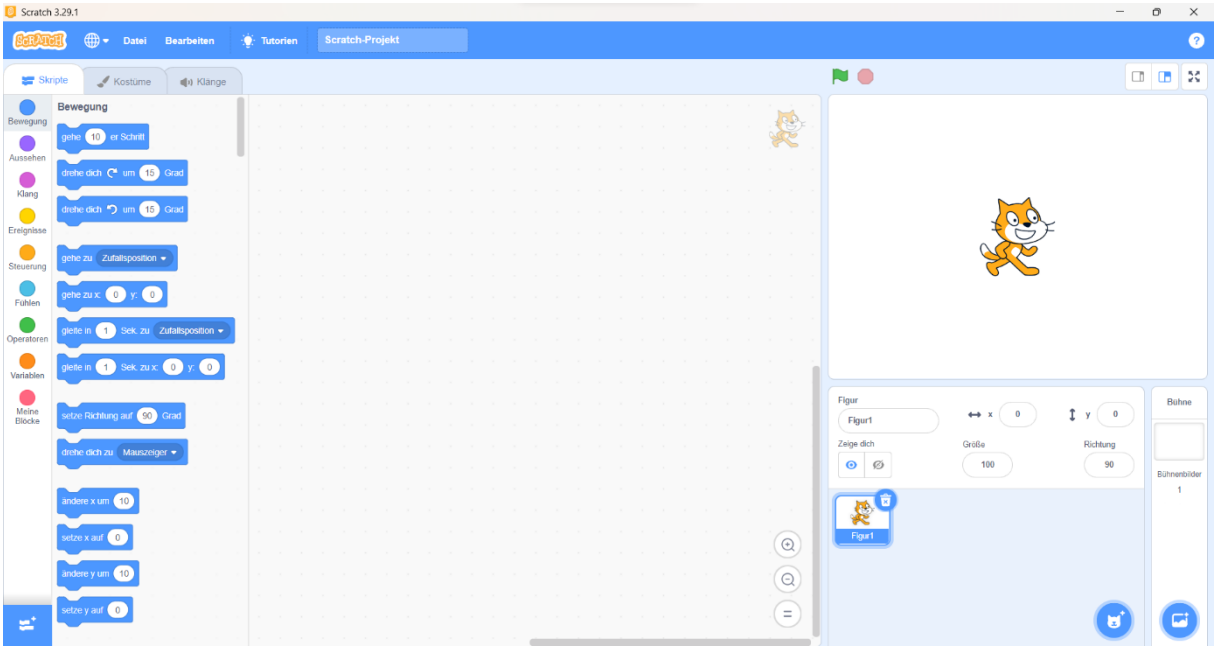

Bühne

Die Bühne ist die hinterste Ebene aller Elemente und hat einige besondere Eigenschaften. Eine Bühne ist stationär, was dazu führt, dass sie sich nicht bewegen ihre Größe nicht ändern und keine Aktionen in Bezug auf ihre Position ausführen kann. Figuren können mit dem Nutzer über Sprechblasen kommunizieren, Bühnen hingegen nicht. Bühnen können jedoch Fragen stellen. Die Quintessenz dieses Textes ist, dass die Bühne weniger Funktionen bietet als eine Figur. Zusammenfassend bedeutet dies, dass nicht alle Blöcke, die zum Schreiben eines Skripts für eine Figur verwendet werden können, auch zum Erstellen eines Bühnen-Skripts verwendet werden können. Es gibt noch einige weitere Einschränkungen für eine Bühne, die man sich nicht auswendig lernen muss und die bei Bedarf im Scratch-Wiki nachgeschlagen werden können (https://de.scratch-wiki.info/wiki/B%C3%BChne).

Die Bühne kann mehrere Hintergründe beinhalten. Hintergründe sind wie Figuren. Sie können im Bühnenfenster in der unteren rechten Ecke erstellt, bearbeitet und entfernt werden. Ein Hintergrund

kann Skripte enthalten, die ausgeführt werden, während der Hintergrund aktiv ist. Im Vergleich zu Figuren kann ein Hintergrund jedoch keine Kostüme enthalten. In Kapitel 7 werden wir uns Hintergründe und ihre Verwendbarkeit genauer ansehen.

2.7 Skriptbereich



Übersicht über den Scratch-Editor.

Der Skriptbereich ist das Herzstück des Scratch-Editor, wo die Magie des Programmierens stattfindet, und befindet sich in der Mitte des Editors. Wenn du den Skriptbereich nicht sehen kannst, vergewissere dich, dass du dich in der Registerkarte „Skript“ befinden und nicht den Mal- oder Klang-Editor verwendest.

Um ein Skript für eine Figur oder einen Hintergrund zu erstellen, ziehe den gewünschten Block per Drag & Drop in den Skriptbereich. Um zwei Blöcke miteinander zu verbinden, verschiebe den zweiten Block so nah an den ersten Block heran, bis eine graue Vorschau-Block erscheint. Der Skriptbereich bietet durch einen Rechtsklick eine Reihe sehr nützlicher Optionen.

Rechtsklick auf den Skriptbereich:

Rückgängig:	Macht die letzte Änderung rückgängig
Wiederherstellen:	Ersetzt die letzte Rückgängig-Bearbeitung
Blöcke aufräumen:	Organisiert alle deine Skripte vertikal
Kommentar hinzufügen:	Fügt einen Kommentar hinzu
Lösche Blöcke:	Entfernt alle Böcke in dem Skriptbereich

Rechtsklick auf einen Block im Skriptbereich:

Duplizieren:	Dupliziert das Skript, zu dem der Block gehört
Kommentar hinzufügen:	Fügt einen Kommentar hinzu
Lösche Block:	Löscht den Block einschliesslich aller darin enthaltener Blöcke

Ein Hinweis zu Kommentaren: Kommentare sind der Schlüssel zu gutem Code. Ich werde dir dies anhand einer kleinen Geschichte erklären. Stelle dir vor, du beginnst heute ein neues Projekt, das jedoch so umfangreich ist, dass du es nicht an einem Tag fertigstellen kannst. Nach einem halben Jahr findest du endlich wieder Zeit, um dein grossartiges Projekt fortzusetzen. Wenn du keine Kommentare oder nur nutzlose Kommentare hast, musst du jedes einzelne Skript, dass du geschrieben hast, verstehen. Mit nützlichen Kommentaren kannst du leicht nachvollziehen, welchen Zweck jedes Skript erfüllt. Was sind gute Kommentare? Ein guter Kommentar beschreibt nicht, was der Code macht, sondern wozu der Code verwendet wird. Ein Beispiel: Du möchtest ein Skript erstellen, das deine Figur um 20 Pixel nach rechts verschiebt. Ein guter Kommentar wäre „Verschieben zur Startposition”. Ein schlechter Kommentar wäre „Figur um 20 Pixel nach rechts verschieben”, da alles, was der Kommentar aussagt, aus dem Skript hervorgeht, das jeder Programmierer, also auch du, leicht versteht.

2.8 Quiz

Welche Editoren stellt Scratch zur Verfügung?

☐ Malprogramm

☐ Klangeditor

☐ Texteditor

☐ Videoeditor

Welche Blocktyp wird am Anfang jedes Skripts verwendet?

☐ Stapelblöcke

☐ Klammerblöke

☐ Kopfblöcke

☐ Abschlussblöcke

Was ist eine Figur?

☐ Ein Hintergrundbild

☐ Eine Konfigurationsoption

☐ Ein Charakter

☐ Eine Gruppe von Blöcken

Wie kannst du dein Spiel starten?

☐ Drücke de S-Taste auf deiner Tastatur

☐ Drücke die grüne Flagge

☐ Doppelklicke den Skriptbereich

☐ Drücke den roten Knopf

Was ist ein Skript?

☐ Ein Audio

☐ Ein Klammerblock

☐ Eine Gruppe an Kostümen

☐ Eine Gruppe von Blöcken

Lösungen dazu findest du in der Paketdatei 2_01.txt

2.9 Notizen

3 Bewegung

Lasse uns beginnen und mit Scratch etwas zaubern. In diesem Abschnitt geht es darum, die berühmte Katzen-Figur zu bewegen und zu drehen. Die Befehle zum Bewegen der Figur findest du in der Kategorie Bewegung im Skriptbereich. Alle Blockkategorien sind mit einer bestimmten Farbe hinterlegt, wodurch sie leicht zu unterscheiden sind. Die Bewegungsblöcke sind blau hinterlegt. Bewegungsblöcke sind meist Stapelblöcke, aber es gibt drei Werteblöcke, welche die Positionswerte der Figur darstellen.

Scratch Katze

3.1 Bewege und drehe!

Um deine Figur zu bewegen und zu drehen, bietet Scratch eine Reihe von Befehlen, die du im oberen Bereich der Blockkategorie „Bewegung“ findest. Ich empfehle dir, den Scratch Editor zu öffnen und die Erklärungen direkt auszuprobieren. Nimm dir dazu den Block und verschiebe ihn in den Skriptbereich. Bearbeite die Werte und doppelklicken auf den Block, um ihn auszuführen.

Gehe () er Schritte

Der einfachste Scratch-Block ist der Block **gehe () er Schritte**. DU kannst einen numerischen Wert in die Klammer setzen (in Scratch setzt du den numerischen Wert in die weisse Lücke). Ein numerischer Wert kann verschiedene Ausdrücke haben. Schauen wir uns diese einmal an.

Positive Zahl: Dies ist der offensichtlichste Wert. Wenn du eine beliebige positive Zahl verwendest, bewegt sich die Figur in festgelegten Schritten in die Richtung, in die es zeigt. Standardmäßig ist dies eine Bewegung von links nach rechts.

Negative Zahl: Wenn du einen negativen Wert mit einem vorangestellten Minuszeichen angibst, bewegt sich die Figur entgegen ihrer Ausrichtung. Standardmäßig ist dies eine Bewegung von rechts nach links.

Gleitkommazahl: Du kannst Gleitkommazahlen als positive oder negative Werte verwenden. Der Gleitkommawert wird durch einen Punkt getrennt. Wenn du den Wert in den Figureigenschaften überprüfst, siehst du bei der Arbeit mit Gleitkommazahlen einen falschen Wert. Dies ist jedoch nur ein Darstellungsproblem. Der tatsächliche Wert wird im Hintergrund korrekt gespeichert, die Figureigenschaften rundet diesen Wert, wodurch 0,5 wie 1 aussieht.

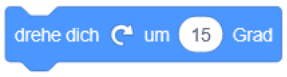
Wissenschaftliche Zahl: Wenn du versuchst, einen Text als Schritte hinzuzufügen, werden die Buchstaben nicht akzeptiert, außer dem Buchstaben e. Aber warum ist das so? Es gibt eine einfache Antwort: e wird in der wissenschaftlichen Notation verwendet und steht für 10 hoch x. Der Ausdruck $0.5 * 10^2$ kann als $0.5e2$ a geschrieben werden und wäre als Wert zulässig.

Wie wir gesehen haben, kannst du numerische Werte verwenden, aber du kannst keine Berechnungen innerhalb der Lücke einfügen. Auch wenn der Wert 5-4 akzeptiert wird, wird er nicht ausgeführt. Wir werden im Abschnitt über Operatoren einen Weg finden, Berechnungen innerhalb der Lücke zu realisieren.

Ein sehr interessantes Verhalten des Blocks **gehe () er Schritte** lässt sich beobachten, wenn er auf den Rand der Bühne trifft, wodurch verhindert wird, dass sich deine Figur nicht ins Nichts bewegt. Selbst wenn du eine sehr große Zahl verwendest, bleibt die Figur am Rand der Bühne stehen.

Jetzt kann sich unsere Figur nach links und rechts bewegen, aber was ist, wenn wir unsere Figur an den oberen oder unteren Rand der Bühne oder in eine andere Richtung bewegen möchten? In Scratch gibt es zwei Blöcke, die dieselbe Funktion auf leicht unterschiedliche Weise erfüllen. Sie können

entweder den Block **drehe dich rechts um () Grad** oder den Block **drehe dich links um () Grad** verwenden. Du kannst einen beliebigen numerischen Wert innerhalb der Klammern verwenden (in Scratch gibt's du den numerischen Wert in die weiße Lücke ein).



Drehe dich rechts um () Grad

Diese beiden Blöcke sind fast gleich. Lassen Sie uns dies anhand eines Beispiels untersuchen. Um deine Figur um 90 Grad nach rechts zu drehen, kannst du den Block **drehe dich rechts um (90) Grad** verwenden, aber du könntest auch den Block **drehe dich links um (-90) Grad** oder sogar **drehe dich links um (270) Grad** verwenden. Der Grund dafür ist die Definition des Kreises. Ein Kreis hat 360 Grad, was 360 gleich großen Teilen entspricht. Wenn du nun die ersten 90 Teile des Kreises nach rechts gehst, befindest du dich an derselben Position, wie wenn du die ersten 270 Teile des Kreises nach links oder die ersten -90 Teile des Kreises nach links nehmen würdest. Das Bild auf der rechten Seite veranschaulicht die beschriebene Redundanz. Mathematisch kann man berechnen: Grad nach rechts = 360 - Grad nach links oder Grad nach links = 360 - Grad nach rechts.

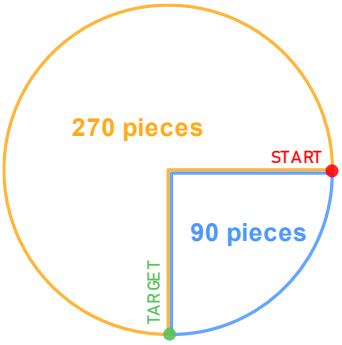


Abbildung einer Links- und Rechtsdrehung

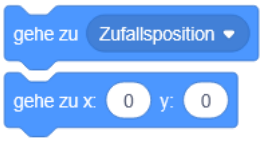
Wenn du den Block bewegen und drehen kombinierst, kannst du deine Figure in jede Richtung und über jede Entfernung bewegen. Beachte dabei jedoch, dass die neuen Koordinaten nicht (1,1) lauten, wenn du deine Figur um 1 Schritt und 45 Grad nach links bewegst, da sich die Figur nicht nur nach rechts, sondern auch nach oben bewegt. Wenn du dich für dieses spezielle Verhalten interessierst, dann lese den Artikel zum Sinusgesetz auf Wikipedia, der in folgender mathematischer Gleichung endet $y = \frac{length * \sin(\alpha)}{\sin(90°)} = length * \sin(\alpha)$ and $x = \frac{length * \sin(180-\alpha)}{\sin(90°)} = length * \sin(180 - \alpha)$.

3.2 Koordinaten

Wir wissen bereits, wie man eine Figur bewegt, aber wie können wir sicherstellen, dass die Figur an der richtigen Position startet? Es gibt eine Reihe von Funktionen, mit denen du deine Figur korrekt auf der Bühne positionieren kannst.

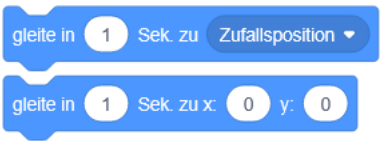
Die Bühne, auf der das Ergebnis angezeigt wird, hat eine feste Größe von 480 Pixeln in der Breite und 360 Pixeln in der Höhe. Der Ursprung (0,0) des Koordinatensystems befindet sich in der Mitte der Bühne. Zusammenfassend lässt sich sagen, dass deine Figur jede Position zwischen (-240, -180) oben links und (240,180) unten rechts einnehmen kann, um eine eindeutige Position zu bieten.

Verwenden Sie den Block **gehe zu x: () y: ()**, um deine Figur an einer beliebigen Stelle auf der Bühne zu positionieren. Als Werte kannst du numerische Werte verwenden. Wenn dein Wert zu hoch oder zu niedrig ist, bleibt die Figur am Rand der Bühne stehen. Wenn deine Figur zu einer Position gleiten soll, gibt es einen integrierten Block mit dem Namen **gleite in () Sek. zu x: () y: ()**. Gib in der ersten weißen Lücke die Bewegungszeit in Sekunden an. Du kannst dieselben numerischen Werte wie im Block **gehe () er Schritte** verwenden, ohne negative Werte. Wenn du einen negativen Wert verwendest, verhält sich der Block **gleite in () Sek. zu x: () y: ()** gleich wie der Block **gehe zu x: () y: ()**.



Gehe zu Stapelblock

Wie du vielleicht bemerkt hast, gibt es zweimal dieselben Blöcke, jedoch mit unterschiedlichen weißen Lücken, welche auch als Argumente bezeichnet werden. Du hast die Möglichkeit, deine Figur entweder gleitend oder direkt an einer zufälligen Position oder bei deinem Mauszeiger zu positionieren. Verwende dazu den Block **gehe zu (zufällige Position/Mauszeiger)** oder **gleite in () Sek. zu (zufällige Position/Mauszeiger)**.



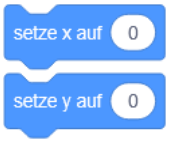
Gleite zu Stapelblock

Der Block **gehe zu (zufällige Position)** oder **gleite in () Sekunden lang zu (zufällige Position)** bewegt deine Figur jedes Mal, wenn du den Block ausführst, an eine zufällige Position. Dies kann verwendet werden, um Gegner an zufälligen Positionen zu generieren.

Mauszeiger Skript

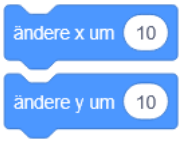
Der Block **gehe zu (Mauszeiger)** oder **gleite in () Sek. zu (Mauszeiger)** positioniert deine Figur bei deinem Mauszeiger. Um den Block **gehe zu (Mauszeiger)** oder **gleite in () Sek. zu (Mauszeiger)** zu demonstrieren, verwende das Skript auf der rechten Seite (auch zu finden im Paket unter /examples/3_mouse_pointer_follow.sb3). Klicke auf die grüne Flagge oben links bei der Bühne und bewege deine Maus auf der Bühne. Die Figur sollte nun deiner Maus folgen, was in vielen Spielen ein gängiger Anwendungsfall ist. Wir werden uns die anderen Blöcke in späteren Kapiteln genauer ansehen.

Was aber, wenn du nur die y- oder x-Koordinate ändern möchtest? Dazu gibt es die Blöcke **setze x auf ()** und **setze y auf ()**. Du kannst beliebige numerische Werte verwenden, diese sind jedoch durch die Größe der Bühne begrenzt. Die Blöcke sind besonders nützlich, wenn du deine Figur auf eine bestimmte Höhe oder Breite nach oben und unten oder vorwärts und rückwärts verschieben möchtest.



Setze Stapelblock

Für die Blöcke **ändere x um ()** und **ändere y um ()** wird im Hintergrund die aktuelle Position verwendet. Diese Blöcke ähneln stark den Schritten **gehe () er Schritte** in Kombination mit **drehe dich links um () Grad** mit einem Wert von 0 oder 90 Grad. Du kannst jeden beliebigen numerischen Wert wie im Block **gehe () er Schritte** verwenden, aber wenn der Rand der Bühne erreicht ist, kann die Bewegung nur in umgekehrter Richtung erfolgen.



Ändere () Stapelblock

Alle diese Blöcke ändern die Position der Figur unter direkter Verwendung der aktuellen Position, aber wie können wir die aktuelle Position der Figur ermitteln? Vielleicht sind dir die Werteblöcke unten in der Kategorie „Bewegungsblöcke" aufgefallen. Der Werteblöcke **x-Position** und **y-Position** geben die aktuelle Position an. Du kannst diese Werteblöcke in alle weißen Lücken einfügen, die die gleiche Form wie ein Werteblock haben. Du könntest beispielsweise den Block **ändere x um ()** verwenden und die weiße Lücke mit dem aktuellen x-Wert füllen, indem du den Werteblock **x-Position** verwendest. Wenn du das Kontrollkästchen links neben dem Reporterblock aktivierst, wird der aktuelle Wert der Variable in der oberen linken Ecke der Bühne angezeigt. Dies kann nützlich sein, um zu überprüfen, ob Ihr Skript wie vorgesehen funktioniert und um die Fließkommazahlen anzuzeigen.

Position Werteblock

3.3 Wohin zeigen

Wir wissen, dass es möglich ist, eine Figur zu drehen. Diese Funktion bringt jedoch das Problem mit sich, dass die Figur bei einer Drehung um 180 Grad auf dem Kopf steht. Scratch bietet eine Lösung für dieses Problem. Mit dem Block **setze Drehtyp auf ()** kann ein Drehungsstil wie folgt festgelegt werden.

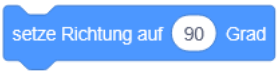
Setze Drehtyp auf Stapelblock

Rundherum: Dies ist der Standardwert, der die Figur um 360 Grad dreht.

Links-Rechts: Dieser Wert dreht die Figur (spiegelt es) am Bruchpunkt von 0 und 180 Grad, wodurch das Problem der umgekehrten Ausrichtung gelöst wird. Wenn du die Figur jedoch bewegst, bewegt es sich in die angegebene Richtung. Wenn du in Richtung 200 Grad zeigst, wird die Figur gespiegelt und bewegt sich während der Bewegung in die linke untere Ecke der Bühne.

Nicht drehen: Dieser Wert dreht deine Figur nicht, aber wenn du eine Punktrichtung festlegst, bewegt sich die Figur in diese Richtung. Diese Option ähnelt der Option „Links-Rechts", jedoch ohne sichtbare Drehung.

Um die Drehung festzulegen, musst du den Block **setze Richtung auf () Grad** verwenden. Du kannst einen beliebigen numerischen Wert in das weiße Feld eingeben, aber denk daran, dass ein Kreis in 360 Teile unterteilt ist, was bedeutet, dass 0 und 360 denselben Wert haben.



Setze Richtung auf () Stapelblock

Einige Spiele folgen nicht nur dem Cursor, sondern zeigen auch in Richtung des Mauszeigers. Daher kann die Methode **dreh dich zu (Mauszeiger)** verwendet werden. Lass uns unser vorheriges Skript, das zum Verfolgen des Mauszeigers verwendet wurde, so abändern, dass es immer in Richtung des Mauszeigers zeigt. Das Skript findest du auf der rechten Seite oder im Paket unter /examples/3_mouse_pointer_direction.sb3. Wenn du auf die grüne Flagge klickst, sollte die Figur immer in Richtung deines Mauszeigers zeigen, unabhängig davon, ob du mit der Maus über die Bühne fährst oder nicht.

Mouse pointer Skript

Um die aktuelle Richtung zu ermitteln, in die die Figur zeigt, kann die Richtung des Werteblocks mit detaillierten Werten auf der Bühne angezeigt oder sogar in jeder weißen Lücke mit derselben Form wie ein Reporterblock verwendet werden.

Um die aktuelle Richtung zu ermitteln, in welche die Figur zeigt, kann der Werteblock **Richtung** verwendet werden. Dieser zeigt auf der Bühne den detaillierten Wert an oder kann auch in weissen Lücken mit der gleichen Form wie ein Werteblock verwendet werden.

Richtung Werteblock

3.4 Übungen

Übung 3_01: Bewege deine Figur 100 Schritte nach rechts.

Übung 3_02: Bewege deine Figur 100 Schritte nach links.
a) Mit einem Block
b) Mit zwei Blöcken

Übung 3_03: Drehe die Figur um 50 Grad nach rechts.
a) Mit dem Block **drehe dich rechts um () Grad**.
b) Mit dem Block **drehe dich links um () Grad**.

Übung 3_04 Drehe deine Figur um 50 Grad nach links und bewege es um 50 Schritte.

Übung 3_05: Positioniere deine Figur bei x = 200, y = 100.

Übung 3_06: Positioniere deine Figur bei x = 100, y = 50 und setze x zu 20

Übung 3_07: Positioniere deine Figur bei x = 100, y = 50 und verringere x um 20

Übung 3_08: Lass deine Figur innerhalb von 5 Sekunden von x = -100, y = 50 nach x = 100, y = -50 gleiten.

Übung 3_09: Setze den Drehtyp auf rechts-links und zeige in Richtung 200 Grad.

Übung 3_10: Setze den Drehtyp auf nicht drehen und zeige in Richtung 45 Grad und laufe 50 Schritte.

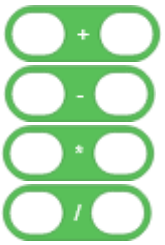
3.5 Notizen

4 Operatoren

Du hast bereits gelernt, wie du deine Figur bewegen kannst. Aber was passiert hinter den Kulissen? Beim Ausführen eines Programms geht es um Berechnungen mit Binärzahlen. In diesem Abschnitt schauen wir uns Berechnungen und Vergleiche an und sehen uns an, welche Blöcke in der Kategorie Operatoren zu finden sind. Da Dezimalzahlen viel einfacher zu verstehen sind, werden wir weiterhin diese verwenden.

4.1 Berechnungen

In der Schule hast du die Operationen Addition **() + ()**, Subtraktion **() - ()**, Multiplikation **() * ()** und Division **() / ()** gelernt. Diese sind genauso ebenfalls in Scratch in der Operatoren Kategorie verfügbar. Die Blöcke nehmen jeweils zwei numerische Werte entgegen und geben das Resultat zurück. Du kannst irgendeinen numerischen Wert oder einen anderen Werteblock in den Lücken verwenden.



Basisoperationen

Werteblöcke können auch verschachtelt werden. Du kannst mehrere Berechnungen durch Verschachtelung verbinden. Die Rechenregeln sind allerdings etwas anders als du dir das gewohnt bist. Wenn du 2*3+1 rechnest würdest du normalerweise erst 2*3 und dann +2 rechnen, was 8 ergibt. In Scratch würde diese Reihen folge allerdings folgendes rechnen: **(2) * ((3) + (2))**, was 10 ergibt. Verschachtelte Werteblöcke werden dementsprechend von innen nach aussen gerechnet. Um die vorherige Rechnung korrekt durchzuführen, müsstest du den Multiplikations-Block nach innen verschieben, sodass **((2) * (3)) + (2)** entsteht.

Wie du bereits weisst kann das Resultat einer Division eine Fliesskommazahl sein. Vielleicht möchtest du aber nur Ganzzahlen verwenden. Scratch bietet dafür den Block **() gerundet**, welcher Werte auf- und abrundet. Wenn die erste Dezimalstelle kleiner als 5 ist, wird der Wert abgerundet, ansonsten aufgerundet.

Runden Werteblock

Scratch kennt auch den Modulo-Operator. Eventuell kennst du diesen noch nicht, aber er ist in der Programmierung, und vor allem in der Verschlüsselung, von grosser Bedeutung. Der Operator ist unter dem Block **() mod ()** verfügbar. Die Modulo-Operation gibt den Restwert einer Division zurück. Beispielsweise würde **12 mod 5** in 2 resultieren, da 12 / 5 = 2, Rest 2. Solch eine Operation ist beispielsweise nützlich für folgende Fälle:

Modulo Wertelock

Gerade oder ungerade: Falls du herausfinden möchtest, ob eine Zahl gerade ist, kannst du sie Modulo 2 rechnen. Wenn das Resultat 0 ist, ist die Zahl gerade, ansonsten ist das Resultat 1 und die Zahl ungerade.

n-tes Element: Falls du etwas nur jedes n-te Mal ausführen möchtest, kannst du **(x) mod (n)** berechnen. Falls das Resultat 0 ist, bist du bei der n-ten Ausführung.

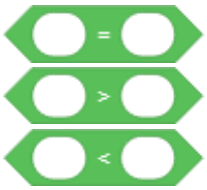
Zeit: Wenn du eine Zeit lesbar gestalten möchtest, ist der Modulo-Operator ebenfalls nützlich. So kannst du beispielsweise für 140 Minuten die Zeit wie folgt formatieren: Die Anzahl Minuten sind 140 mod 60 = 20 und die Anzahl Stunden sind **Abrunden (140 / 60)** = **2**.

Es gibt noch viele weitere mathematische Operatoren in Scratch. Diese sind über den **() von ()**-Block verfügbar. Unten findest du eine Liste der verfügbaren Operationen - aber keine Sorge, wenn du nicht alle verstehst. Du wirst dir mit der Zeit automatisch lernen, sobald du sie benötigst.

Betrag	Gibt den absoluten Wert zurück	Beispiel: Betrag von -10 = 10
Abrunden	Gibt den abgerundeten Wert zurück	Beispiel: Abrunden von 10.99 = 10
Aufrunden	Gibt den aufgerundeten Wert zurück	Beispiel: Aufrunden von 10.01 = 11
Wurzel	Gibt die Wurzel zurück	Beispiel: Wurzel von 9 = 3
Sin	Gibt den Sinus zurück	Beispiel: sin von 90 = 1

cos	Gibt den Cosinus zurück	Beispiel: cos von 0 = 1
tan	Gibt den Tangens zurück	Beispiel: tan von 0 = 0
asin	Gibt den Arc-Sinus zurück	Beispiel: asin von 1 = 90
acos	gibt den Arc-Cosinus zurück	Beispiel: acos von 1 = 0
atan	Gibt den Arc-Tangens zurück	Beispiel: atan von 0 = 0
ln	Gibt den natürlichen Logarithmus zurück	Beispiel: ln von e = 1
log	Gibt den 10-er Logarithmus zurück	Beispiel: log von 10 = 1 log (10)
e ^	Gibt den e^<Wert> zurück	Beispiel: e ^ von 1 = e
10 ^	Gibt den 10^<Wert> zurück	Beispiel: 10 ^ von 5 = 100'000

4.2 Vergleiche



Gleich, Grosser als und kleiner als Wahrheitsblöcke

Nach dem Berechnen eines Wertes vergleicht man ihn oft mit einem anderen Wert, um das Verhalten des Programmes basierend auf dem Resultat zu ändern. Dazu gibt es in Scratch die aus der Schule bekannten Vergleichsoperatoren: gleich **() = ()**, grösser als **() > ()** und kleiner als **() < ()**. Die Vergleichsoperatoren findest du in der Operatoren Kategorie. Die Vergleiche geben jeweils einen Wahrheitswert zurück (wahr oder falsch). Als Argumente kannst du alphanumerische Werte verwenden.

Wenn du numerische Werte verwendest, musst du mit den Blöcken für "grösser als" und "kleiner als" aufpassen, da es keine grosser als oder gleich und kleiner als oder gleich Blöcke gibt. Dies bedeutet, dass () < (100) nur wahr zurückgibt, sofern der Wert von negativem unendlich bis 99 geht. Umgekehrt gilt auch, dass () > (100) nur wahr zurückgibt, sofern der Wert von 101 bis unendlich geht.

Wenn du einen alphabetischen Wert verwendest, musst du ebenfalls aufpassen. Die Gross- und Kleinschreibung ist irrelevant. Das bedeutet **a = A** ist wahr. Für "Grösser als" und "Kleiner als" gilt die alphabetische Sortierung, heisst **a < b** ist wahr und **b > c** ist falsch, da A vor B und B vor C kommt.

Scratch bietet auch logische Operationen, um mehrere Wahrheitswerte zu vergleichen. Wir schauen uns im Folgenden ein paar Beispiele an.

Ist die x-Position der Figur zwischen 50 und 100?
Das Problem sieht schwierig aus, aber sobald du das Konzept verstehst, ist es se sehr einfach. Wir müssen die Frage in zwei Vergleiche aufteilen. Erst überprüfen wir, ob der Wert größer als 50 ist und anschliessend ob er kleiner als 100 ist. Dazu gibt es in Scratch einen Wahrheitsblock: **() und ()**:

Bei diesem Block müssen alle Vergleiche wahr sein damit die Gesamt-Aussage wahr ist.

Ist die x-Position der Figur kleiner als 50 oder grösser als 100?
Auch hier müssen wir die Frage in zwei Vergleiche aufteilen, allerdings mit einem anderen Schlüsselwort. Erst überprüfen wir, ob die x-Position kleiner als 50 ist und dann, ob die x-Position grösser als 100 ist - unabhängig vom ersten Resultat. Dazu bietet Scratch einen Wahrheitsblock: **() oder ()**:

Bei diesem Block muss mindestens ein Vergleich wahr sein, damit die Gesamt-Aussage wahr ist.

Ist die x-Position der Figur nicht kleiner als 200?
Dieses Problem kann auf mehrere Arten gelöst werden. Es gibt in Scratch den **nicht ()**-Block, welcher das Resultat einer Bedingung umkehrt. Damit wird aus wahr falsch und umgekehrt:

Hier schauen wir, ob die x-Position kleiner als 200 ist. Durch die Umkehrung des Resultats erhalten wir die Antwort auf die originale Frage. Alternativ könnte man auch **x-Position > 199** überprüfen. Diese Lösung ist einfacher verständlich und führt zu demselben Resultat.

Die Vergleichs-Blöcke sind sehr mächtig aber werden noch mächtiger, wenn man sie verschachtelt. Man kann die **() und ()**, **() oder ()** und **nicht ()**-Blöcke frei kombinieren, um komplexe Fragestellungen zu lösen.

Geschachtelte Bedingungen

4.3 String-Operationen

Scratch bietet auch Basis-Blöcke für die Verarbeitung von Texten (“Strings”). Strings werden verwendet, um Nachrichten an den Nutzer zu schicken oder um die Antwort eines Nutzers zu verstehen. Wir schauen uns diese Interaktion in späteren Kapiteln genauer an. Strings können fast alle Zeichen erhalten (Zahlen, Buchstaben, Emoji, etc.).

Der **verbinde () und ()**-Block kombiniert zwei Strings zu einem. Dies ist beispielsweise nützlich, wenn du den Nutzer nach seinem Vor- und Nachnamen fragst und anschliessend die beiden Namen kombinieren möchtest. Du kannst diesen Block verschachteln, um Strings zu kombinieren.

Verbindungsblock

Scratch bietet weitere Blöcke, die verwendet werden, um Strings genauer zu analysieren. Dazu gehört beispielsweise der **Zeichen () von ()**-Block. Dieser gibt den n-ten Buchstaben eines Strings zurück. Beispielsweise würde der Block **Zeichen (1) von (Hallo)** den Buchstaben ”a” zurückgeben.

Zeichen () von () Stapelblock

Der Block **Länge von ()** gibt die Länge, respektive die Anzahl Zeichen eines Strings zurück.

Länge eines Strings

Der Wahrheitsblock **() enthält ()?** überprüft, ob der erste String den Zweiten enthält. Beispielsweise würde **banana enthält a?** wahr zurückgeben, während **banana enthält x** falsch zurückgibt. Die Überprüfung ignoriert die Gross- und Kleinschreibung.

String enthält Substring

4.4 Übungen

Übung 4_01: Berechne (4 + 5) * (3 – 6) mittels Scratch.

Übung 4_02: Überprüfe, ob 3 * 5 grösser als 14, und 5 * 6 kleiner als 31 ist.

Übung 4_03: Überprüfe, ob der absolute Wert von -20 grösser ist als 0.

Übung 4_04: Überprüfe, ob 15 mod 2 in 1 resultiert oder ob 15 mod 2 nicht grösser als 0 ist.

Übung 4_05: Überprüfe, ob der String “bracelet” den String “race” enthält und finde die Länge des Strings.

Übung 4_06: Überprüfe, ob der dritte Buchstabe von “Adventure” der Buchstabe “v” ist und ob er grösser ist als “w”.

Übung 4_07: Überprüfe, ob (“a” grösser als “c” und “c” grösser als “b") oder (“a” grösser als “b” und “b” nicht kleiner als “c”) ist.

Übung 4_08: Vereine die Wörter "Anti” und “dote” und überprüfe, ob das Resultat das Wort “ido” beinhaltet und der 7te Buchstabe “h” ist.

4.5 Notizen

5 Ereignisse und Steuerung

Dieses Kapitel zeigt dir wie du komplexe und innovative Werkzeuge erstellen kannst. Es bereitet dich darauf vor, Entscheidungen in deinem Skript zu treffen, wie wann etwas starten soll oder was an einem bestimmten Punkt passieren soll. Die Blöcke in diesem Kapitel können in den Kategorien Ereignisse und Steuerung gefunden werden.

5.1 Ereignisse bei Aktionen

Wir mussten bisher jeden Block doppel-klicken, um ihn auszuführen. Mittels Kopfblöcke können wir nun einen oder mehrere Stapelblöcke kombinieren und sie unter gewünschten Bedingungen ausführen. Einer der wichtigsten Blöcke nennt sich **Wenn grüne Flagge geklickt wird**. Dieser führt alle darunter angefügten Blöcke aus, wenn die grüne Flagge angeklickt wird. Um die Ausführung zu stoppen kannst du den Stopp Knopf neben der Flagge klicken. Diese Abfolge wird oft verwendet, um ein Programm zu starten und die Anfangskonfiguration, wie die Startposition zu setzen.

Wenn Flagge geklickt wird

Scratch bietet auch den **Wenn Taste () gedrückt wird** Block, welcher dein Skript ausführt, sobald die gewünschte Taste gedrückt wird. Du kannst eine der folgenden Optionen wählen:

Wenn () gedrückt wird Kopfblock

Buchstaben: Das Skript wird ausgeführt, wenn der entsprechende Buchstabe gedrückt wird (unabhängig davon ob gross oder klein geschrieben).

0-9: Das Skript wird ausgeführt, wenn die entsprechende Ziffer (0-9) gedrückt wird.

Pfeil hoch/runter/links/rechts: Das Skript wird ausgeführt, wenn die entsprechende Pfeiltaste gedrückt wird. Dies wird oft verwendet, um der Figur in die entsprechende Richtung zu bewegen.

Leertaste: das Skript wird ausgeführt, wenn die Leertaste gedrückt wird. Dies wird oft verwendet, um das Spiel zu pausieren.

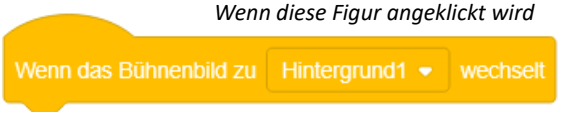
Beliebige Taste: Das Skript wird ausgeführt, wenn irgendeine Taste gedrückt wird. Dies wird oft verwendet, um das Programm zu starten.

Du kannst alle Tasten für dein Spiel verwenden (diese Liste ist nicht vollständig). Wir empfehlen dir in einem separaten Dokument die Zuweisung der Tasten zur entsprechenden Funktion zu dokumentieren und dem Nutzer zur Verfügung zu stellen, oder gar am Anfang des Programms zu zeigen damit er die Steuerung kennt. Achte ebenfalls darauf, dass du nicht zwei verschiedene Funktionalitäten auf die gleiche Taste konfigurierst.

Scratch bietet den Block **Wenn diese Figur angeklickt wird** welches das Skript ausführt sobald die entsprechende Figur angeklickt wird.

Wenn diese Figur angeklickt wird

Mittels des Blocks **Wenn das Bühnenbild zu () wechselt** wird dein Skript ausgeführt, sobald der Hintergrund auf den gewünschten Hintergrund wechselt. Dies kann verwendet werden, um deine Figur in die richtige Position zu bringen. Falls du den gleichen Hintergrund in mehreren Szenen verwendest, wird dein Skript jedes Mal ausgeführt, entsprechend musst du aufpassen, dass sich dein Spiel in beiden Fällen korrekt verhaltet.



Wenn das Bühnenbild zu () wechselt Kopfblock

Es gibt auch noch den Block, **wenn () > ()**, welche primär für die Fühle-Blöcke verwendet wird (diese sehen wir noch in einem späteren Kapitel). So kann er beispielsweise verwendet werden, um ein Skript auszuführen, falls die Spielzeit einen gewissen Wert erreicht hat (das ”>” steht für das mathematische ”grösser als”).

Wenn () > () Kopfblock

5.2 Nachrichten verarbeiten

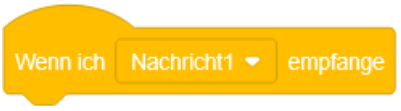
Manchmal müssen mehrere Figuren miteinander kommunizieren können. Dafür bietet Scratch einen Broadcast. Diese Blöcke erlauben dir Nachrichten zu versenden und zu empfangen, wobei du eine spezifische Figur oder alle Figuren als Empfänger definieren kannst.

Sende () an alle Stapelblock

Mittels der **sende () an alle** und **sende () an alle und warte** -Blöcke können Nachrichten verschickt werden. Beide verschicken eine Nachricht, der letztere wartet allerdings, bis alle Empfänger die Nachricht verarbeitet und ihre entsprechenden Scripts ausgeführt haben.

Sende () an alle und warte Stapelblock

Auf der Empfängerseite kann man basierend auf der Nachricht mittels des Blocks, **wenn ich () empfange** eine entsprechende Aktion, respektive ein Skript ausgeführt werden.



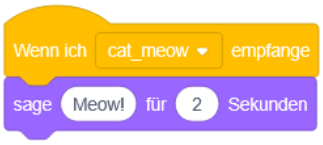
Wenn ich () empfange, Kopfblock

Um eine Nachricht zu erstellen, wähle die Option “Neue Nachricht” im Dropdown eines der Broadcast-Blöcke. Du musst dabei der Nachricht einen Namen geben, ähnlich wie bei einer Variable. Versuche einem Namenskonzept zu folgen, um mehrere Nachrichten für die gleiche Aktion zu vermeiden.

Dies war nun viel Theorie aber lass uns die Kommunikation in Scratch an einem Beispiel anschauen. Möglicherweise hast du eine Katze, einen Hund und zwei Knöpfe in deinem Feld. Der erste Knopf soll den Hund bellen lassen und der zweite Knopf soll die Katze miauen lassen. Wir verwenden die Namenskonvention “<Empfänger>_<Aktion>” für die Scripts:

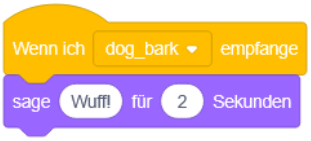
Knopf (Katze)

Katze



Knopf (Hund)

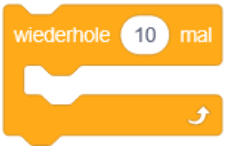
Hund



Das Skript kann in den Ressourcen unter “/examples/5_message_example.sb3” gefunden werden. Versuche, das Skript zu öffnen und überprüfe ob und wie das Anklicken der Knöpfe die entsprechenden Nachrichten verschickt und die Aktionen auslöst.

5.3 Schleifen

Oftmals muss ein Code-Abschnitt mehrmals durchgeführt werden. Wir könnten ihn entsprechend mehrfach schreiben. Allerdings ist das schwer zu unterhalten und zu verstehen. Dafür verwendet man sogenannte Schleifen. Jeder Durchgang einer Schleife nennt man eine Iteration. Scratch bietet die folgenden Schleifenarten:



Wiederhole x-mal

Wiederhole () mal: Dieser Block führt das darunterliegende Skript mehrfach aus, bis zu einem gewünschten Limit. Beispielsweise 47 Durchläufe/Iterationen. Danach läuft das nachfolgende Skript weiter.

Wiederhole bis

Wiederhole bis (): Dieser Block führt das darunterliegende Skript mehrfach aus, bis eine Bedingung erfüllt wird. Danach läuft das nachfolgende Skript weiter. Die Bedingung wird bei jeder Iteration der Schleife überprüft.

Wiederhole fortlaufend

Wiederhole fortlaufend: Dieser Block führt das darunterliegende Skript endlos, immer und immer wieder aus.

Welche Schleife du verwendest hängt von deinem Problem ab, welches du zu lösen versuchst. Die **Wiederhole fortlaufend** -Schleife ist nützlich für unendliche Aktionen, wie das Bewegen der Figur zur

Mausposition, oder der Repetition der Hintergrundmusik. Die **Wiederhole () mal** und **Wiederhole bis ()**-Schleifen sind nützlich, wenn eine Aktion ab einem gewissen Zeitpunkt enden soll.

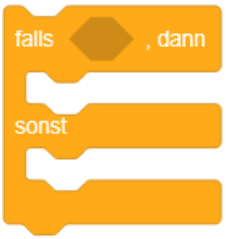
Als Beispiel: Stell dir vor deine Figur bewegt sich entlang eines Kreises. Du kannst nun einen Block hinzufügen, der deine Figur um 1 Grad dreht und 2 Schritte nach vorne bewegt. Nun kannst du den Block entweder 360-mal im Skript hinzufügen oder mittels einer **Wiederhole () ml**-Schleife 360-mal den einen Block ausführen.

5.4 Aussagen und Bedingungen

Du hast bereits alle Operatoren und wichtigen Wahrheitsblöcke kennengelernt. Allerdings haben wir noch nicht angeschaut, wo du diese Blöcke verwenden kannst. Die mächtigsten Blöcke sind die **falls (), dann** und **falls (), dann sonst**-Blöcke. Diese Blöcke entscheiden basierend auf deinen Aussagen, welche Scripts ausgeführt werden und welche nicht. Diese Operatoren-Kombination nennt man eine Bedingung.

Falls () Klammerblock

Stell dir vor du fragst jemanden nach ihrem alter. Falls die Person älter als 60 Jahre ist möchtest du vielleicht sagen ”Diese Person ist geniesst den Ruhestand”. dies kann mit einem, **falls (), dann**-Block gelöst werden, mit der Bedingung, dass das Alter mindestens 60 ist. In jedem anderen Fall möchtest du sagen ”Diese Person ist voller Energie”. Dafür kannst du einen **falls (), dann sonst**-Block verwenden, mit derselben Bedingung.



Fals () dann, sonst

Wenn du nun beispielsweise für Personen, welche jünger als 20 Jahre sind “diese Person lernt viel”, für Personen über 60 “diese Person geniesst den Ruhestand” und für alle anderen Personen “Diese Person arbeitet viel” sagen möchtest, dann musst du die Bedingungen verschachteln:

5.5 Warten & stoppen

Erinnerst du dich an den Kreis, auf dem die Figur sich bewegte? Die Figur bewegte sich mit einer vordefinierten Geschwindigkeit in der Schleife. Um diese Zeit zu verlängern, gibt es in Scratch den **warte () Sekunde**-Block. Dieser Block pausiert das Skript für die angegebenen Anzahl an Sekunden. Es ist auch möglich eine Gleitkommazahl z verwenden und weniger als 1 Sekunde zu warten. Dieser Block kann auch verwendet werden, um beispielsweise Instruktionen anzuzeigen oder das Spiel nach 5 Sekunden zu starten.

Warte () Sekunden Stapelblock

Stell dir vor du hast zwei Figuren, die den Kreis entlang gehen und die erste Figur startet 3 Sekunden früher (mittels des **warte () Sekunden**-block. Nach einem halben Kreis stoppt die erste Figur und geht weiter sobald die zweite Figur aufgeholt hat. Dafür gibt es in Scratch den Block **warte bis ()** welcher wartet, bis eine Bedingung zutrifft.

Warte bis () Stapelblock

Stell dir nun vor, deine Figur jongliert, während sie den Kreis entlang geht. Das Jonglieren und Gehen sind zwei verschiedene Skripts. In der oberen, rechten Ecke gibt es drei Notfall-Knöpfe. Der erste Knopf stoppt das Jonglieren und Gehen, der zweite nur das Jonglieren und der dritte nur das Gehen. Für den ersten Knopf können wir einen **stoppe ()**-Block mit dem Argument **alles** verwenden. Beim zweiten Knopf können wir den **stoppe ()**-Block mit dem Argument **andere Skripte dieser Figur** verwenden. Für den dritten Knopf verwenden wir das Argument **dieses Skript**. Eine ähnliche Implementation findest du im Beispiel 5_stop_scripts.sb3.

Stoppe () Abschlussblock

5.6 Übungen

Übung 5_01: Lass die Figur 100 Schritte gehen und dann drehe ihn um 90° nach rechts. Wiederhole das 7 mal. In welche Richtung schaut die Figur?

Übung 5_2: Lass die Figur 45° nach links drehen und gehe 20 Schritte, wenn die Pfeiltaste rechts gedrückt wird. Wenn die Pfeiltaste links gedrückt wird, mach das Gegenteil (drehe 45° nach rechts und gehe 20 Schritte).

Übung 5_3: Wähle eine zufällige Zahl zwischen 884 und 920, wenn die Leertaste gedrückt wird. Falls die Zahl grösser als 21*43 ist, gehe 20 Schritte nach rechts. Ansonsten gehe 20 Schritte nach links.

Übung 5_4: Erstelle einen Knopf. Wenn er gedrückt wird, drehe die Figur 15° nach links und gehe 10 Schritte.

Waren diese Übungen schwierig? Keine Sorge, diese Übungen erfordern zuvor erworbenes Wissen, das Zeit braucht, um sich zu festigen.

5.7 Fortgeschrittene Übungen

Die folgenden Übungen sind etwas schwieriger, bieten allerdings einen Lösungsvorschlag, falls du Schwierigkeiten hast sie zu lösen.

Übung 5_05: Steuere deine Figur

Ziel: Steuere die Bewegung deine Figur mit den Pfeiltasten. Jeder Tastendruck soll die Figur 10 Schritte in die entsprechende Richtung bewegen.

Schritte:
1. Wähle den **Wenn Taste () gedrückt wird**-Block
2. Setze den Rotations-Stil (Rauf und runter = rundherum; Links und Rechts = links-rechts)
3. Zeige in die Richtung (90° = rechts)
4. Gehe 10 Schritte
5. Wiederhole die Schritte für jede Pfeiltaste und modifiziere die Parameter

Übung 5_06: Steuere deine Figur mit Knöpfen

Ziel: Steuere die Bewegung deiner Figur mit Knöpfen, welche die Pfeiltasten repräsentieren. Beim Drücken eines Knopfes soll sich die Figur 10 Schritte in die entsprechende Richtung bewegen.

Schritte:
1. Erstelle einen Knopf (Tipp: Zeichne einen Knopf in Scratch)
2. Konfiguriere die Knöpfe
    - 2.1 Wähle den **Wenn diese Figur angeklickt wird**-Block
    - 2.2 Füge einen Broadcast hinzu
    - 2.3 Sende eine neue Nachricht für die entsprechende Richtung
3. Konfiguriere deine Figur
    - 3.1 Wähle den **Wenn ich () empfange**-Block

3.2 Setze den Rotations-Stil (Rauf und runter = rundherum; Links und Rechts = links-rechts)
3.3 Zeige in die Richtung (90° = rechts)
3.4 Gehe 10 Schritte

4. Wiederhole die Schritte für jeden Knopf und modifiziere die Parameter

Übung 5_07: Meide die Ränder

Ziel: Lass deine Figur durchgehend immer 10 Schritte gehen. Sobald deine Figur einen Rand berührt, wechsle die Richtung um 55° und bewege die Figur weiter. Die Figur sollte immer nach rechts oder links schauen.

Schritte:
1. Wähle den **Wenn grüne Flagge angeklickt wird**-Block
2. Setze den Rotations-Stil auf “links-rechts”
3. Füge die **wiederhole fortlaufend**-Schleife hinzu
4. Erstelle ein **Wenn (), dann**-Statement in der Schleife
5. Erstelle eine Randberührungsbedingung
   ([Betrag von (x-Position)] > 190) oder ([Betrag von (y-Position)] > 110)
6. Drehe im **Wenn (), dann**-Statement um 55 Grad
7. Gehe 10 Schritte

5.8 Notizen

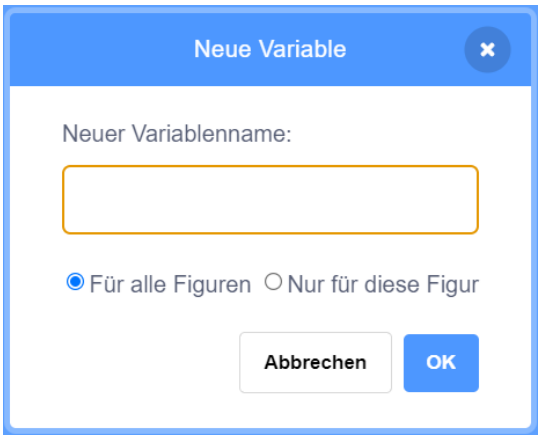
6 Variablen und Listen

Dieser Abschnitt behandelt Variablen und Listen, welche die Möglichkeit bieten, Daten in Scratch zu speichern. Dies ist in vielen Situationen von Nutzen, wie beispielsweise für das Abspeichern des Nutzernamens, um gesammelte Punkte im Spiel abzuspeichern und vieles mehr. Die Blöcke dieses Abschnitts können in der Kategorie Variablen gefunden werden.

6.1 Variablen verwalten

Variablen sind sehr nützlich, um einzelne Werte zu speichern. Der Wert kann eine Zahl oder ein Text sein und ist als Werteblock nutzbar. Sie werden oft für das Zwischenspeichern von Werten einer Figur genutzt.

Anders als die bisherigen Blöcke müssen diese Blöcke erst mittels des “Neue Variable”-Knopf erstellt werden. Dies öffnet ein neues Fenster, um die Variablen-Informationen zu erfassen:



Popup beim erstelleneiner neuen Variable

Variablenname: Jede Variable wird durch einen eindeutigen Namen identifiziert, heisst es gibt nie zwei Variablen mit dem gleichen Namen. Du kannst für den Namen eine willkürliche Zeichenfolge nutzen, aber vorzugsweise solltest du einen beschreibenden, generischen Namen nutzen. Es sollte anderen Entwicklern auf den ersten Blick klar sein, wozu die Variable genutzt wird. Du kannst auch mehrere Wörter im Namen nutzen.

Globale Variablen: Globale Variablen können durch die Option “Für alle Figuren” gesetzt werden. Dadurch wird die Variable für alle Figuren nutzbar. Dies kann für Spielstände oder Nutzernamen nützlich sein, welche für mehrere Figuren relevant sind. Du solltest jedoch immer dem “Need-To-Know"-Prinzip folgen, heisst vorzugsweise ist eine Variable eine Lokale Variable, ausser es gibt einen Grund sie global zu setzen.

Lokale Variablen: Lokale Variablen werden auch private Variablen genannt, da sie nur einer spezifischen Figur zur Verfügung stehen. Um eine lokale Variable zu erstellen, wähle die Option “Nur für diese Figur”. Lokale Variablen werden meist für die Speicherung des Zustands einer spezifischen Figur genutzt.

Du kannst auch Variablen für Hintergründe (“backdrops”) erstellen welche für alle Figuren zur Verfügung stehen. Es wird empfohlen, Variablen, die nicht für eine spezifische Figur sind in einem Hintergrund zu speichern. Bevor du eine Variable erstellst, überlege dir, wo du sie nutzt und welche Sichtbarkeit sie haben muss. Nach dem Ausfüllen des Formulars kannst du “Ok” klicken, um die Variable zu erstellen.

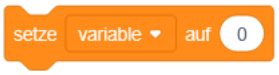
Eine Variable kann über vier Stapelblöcke gesteuert werden. Der Block **zeige Variable ()** zeigt den aktuellen Wert der Variable in der oberen, linken Ecke an. Alternativ kann auch die Checkbox links am Werteblock in der Blockübersicht gesetzt werden. Mit dem **verstecke Variable ()**-Block kannst du den Wert verstecken. So kannst du beispielsweise für einen Highscore

Zeige Variable () Stapelblock

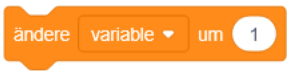
eine Variable mittels **verstecke Variable ()** verstecken und sobald das Spiel durch ist mittels der **zeige Variable ()** wieder anzeigen. Da bei den beiden Blöcken die Variable als Argument mitgegeben wird, kannst du so beispielsweise auch zwischen dem persönlichen und dem generellen Highscore wechseln.

Verstecke Variable () Stapelblock

Um den Wert einer Variable zu setzen kannst du den **setze () auf ()**-Block verwenden. Dieser Block überschreibt den aktuell gespeicherten Wert in der ausgewählten Variable zu der Zahl oder dem Text, den du angibst. Mittels **ändere () um ()**-Block kann der Wert einer Zahl in einer Variable zusätzlich auch um den angegebenen Wert erhöht oder mittels einer negativen Zahl gesenkt werden. Dies kann beispielsweise für Lebenspunkte oder generelle Punkte im Spiel verwendet werden.



Setze () auf () Stapelblock



Ändere () um () Stapelblock

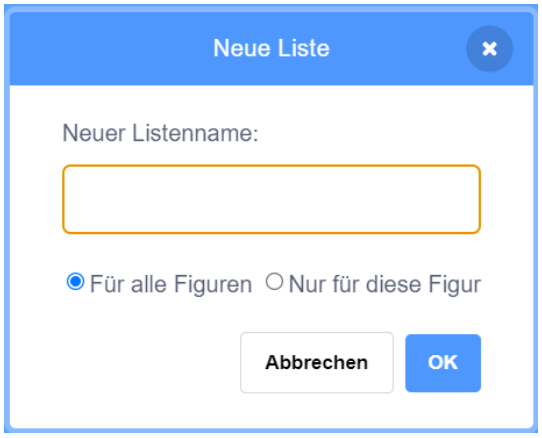
Um auf einen Variablen-Wert zuzugreifen kannst du den entsprechenden Werteblock verwenden, ähnlich des "x-Position" oder "y-Position"-Blöcke in der Bewegung Kategorie. Dieser Werteblock wird oft für Berechnungen oder Vergleiche verwendet.

Werteblock

Um eine Variable umzubenennen kann irgendein Block verwendet werden, der Variablen verwaltet. Dort gibt es jeweils eine Option zur Umbenennung der Variable, welche dasselbe Formular wie bei Erstellen der Variable anzeigt. Dort kannst du den Namen entsprechend ändern. Falls die Variable gar nicht mehr benötigt wird, kannst du sie auch löschen.

6.2 Listen verwalten

Variablen eröffnen viele Möglichkeiten für innovative und herausfordernde Programme. Allerdings sind sie umständlich, wenn viele Daten verarbeitet, werden müssen, wie beispielsweise bei einer Rangliste. In diesem Fall helfen Listen, bei welchen Werte kontinuierlich hinzugefügt werden können. Du kannst eine Liste über das "Neue Liste"-Menü erstellen, ähnlich wie bei der Erstellung einer Variable.



Popup beim Erstellen einer neuen Liste

Verwende bei der Benennung der Liste die gleiche Vorgehensweise wie bei den Variablen. So solltest du stets sprechende Namen verwenden. Ähnlich wie bei den Variablen kann eine Liste ebenfalls lokal oder global verfügbar sein, entsprechend deiner Auswahl bei der Erstellung. Für Listen, welche nicht an Figuren angehängt sind, kannst du ebenfalls eine Liste bei Hintergründen ("backdrops") erstellen.

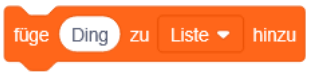
Listen sind etwas mächtiger als Variablen und haben mehr Blöcke zur Verwaltung der Liste und ihrer Werte. Mittels **zeige Liste ()** und **verstecke Liste ()** können Listen angezeigt und versteckt werden. Während der Entwicklung ist es nützlich, Listen anzuzeigen, um zu überprüfen, ob die Logik entsprechend den Erwartungen funktioniert. Dazu sollte die Checkbox neben dem Werteblock gesetzt werden. Denke einfach daran, vor der Vervollständigung deines Spiels die Checkbox wieder zurückzusetzen.

Zeige Liste () Stapelblock

Verstecke Liste () Stapelblock

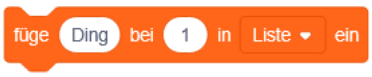
Eine Liste ist ein Behälter für Werte zu jedem Schlüssel. Die Schlüssel entsprechend einer Nummerierung welche bei 1 startet. Jeder neue Wert erhält einen neuen Schlüssel, der um 1 höher ist als der vorherige. Diese Schlüssel nennt man "index" und sie werden verwendet, um auf spezifische

Werte zuzugreifen. Als Beispiel kannst du dir eine Liste von Mitarbeitern vorstellen, bei welcher die Werte den Namen des entsprechenden Mitarbeiters enthalten. Wenn nun ein neuer Mitarbeiter hinzukommt, wird sein Name hinzugefügt und er erhält eine neue Nummer. Muss ein Name angepasst werden, kann man über seine Nummer den richtigen Eintrag finden.



Füge () zu () hinzu Stapelblock

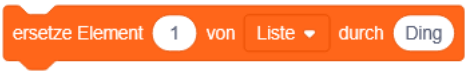
Um einen Wert zu einer Liste hinzuzufügen, kann der Block **füge () zu () hinzu** verwendet werden. Dabei ist das erste Argument den Wert und das zweite die gewünschte Liste. Falls ein Eintrag mittendrin hinzugefügt werden soll, kann der Block **füge () bei () in () ein** verwendet werden. Dabei ist das erste Argument der neue Wert, das zweite die Position und das dritte die gewünschte Liste. Die Position muss grösser als 0 sein und kleiner oder gleich der Anzahl bisheriger Einträge in der Liste.



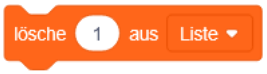
Füge () bei () in () ein Stapelblock

Um die aktuelle Anzahl Einträge zu erfahren, kannst du den Block **Länge von ()** verwenden. Wenn du einen Eintrag mitten in der Liste hinzufügst, wird der Index aller Elemente nach dieser Position um 1 erhöht. Wenn du also einen Eintrag beim Index 10 hast und nun bei Position 10 einen Eintrag einfügst, dann hat der ursprüngliche Eintrag nun den Index 11, was vorher 11 war ist nun 12 und so weiter. Oftmals möchtest du anstatt eines Eintrags mitten in der Liste hinzuzufügen, stattdessen einen Eintrag austauschen. Dazu dient der Block **ersetze Element () von () durch ()**. Das erste Argument ist dabei der Index des Eintrags, das zweite die Liste und das dritte der neue Wert. Falls am gewünschten Index noch nichts existiert, passiert nichts.

Läng von () Werteblock



Ersetze Element () von () durch () Stapelblock



Lösche () aus () Stapelblock

Manchmal möchte man auch Einträge löschen. Dazu gibt es den **lösche () aus ()**-Block. Dabei ist das erste Argument der Index des Eintrags und das zweite Argument die Liste. Falls es den Index nicht gibt, passiert nichts. Du kannst eine Liste mittels des Blocks **lösche alles aus ()** auch komplett leeren.

Lösche alles aus ()

Wenn du Einträge aus der Liste abfragen möchtest, kannst du dazu den Werteblock **Element () von ()** verwenden, wobei wiederum das erste Argument der Index und das zweite die Liste ist. Falls es an dem Index nichts gibt, wird ein leerer Wert zurückgegeben. Du kannst auch umgekehrt den Index mittels des Werts ermitteln, dazu verwendest du den Block **Nummer von () in ()**, wobei das erste Argument der Wert ist und das zweite die Liste. Falls der Wert nicht gefunden wurde, erhältst du den Index 0. Um Fehler zu verhindern ist es oftmals sinnvoll, vorher mittels des Wahrheitsblocks **() enthält ()?** zu überprüfen, ob die Liste den Wert überhaupt enthält.

Element () von () Werteblock

Nummer von () in () Werteblock

() enthält () Wahrheitsblock

Ähnlich wie bei Variablen, kann eine Liste über irgendeinen Block mit dem Listen verwaltet werden eine Liste umbenannt oder gelöscht werden.

6.3 Übungen

Übung 6_01: Erstelle eine Variable mit dem Namen “name” und speichere darin deinen Namen, wenn das Spiel startet.

Übung 6_02: Erstelle eine Liste mit dem Namen “familie” und füge alle Familienmitglieder hinzu.

Übung 6_03: Erstelle eine Liste mit dem Namen “Einkaufsliste” und füge Brot, Milch, Eier und Äpfel hinzu. Ersetze anschliessend den dritten Eintrag mit Butter, ersetze die Milch mit Reis, füge Käse am Index 2 hinzu und entferne die Eier.

Übung 6_04: Kontrolliere die Bewegung deiner Figur mittels der Pfeiltasten und speichere alle Bewegungen in einer Liste.

6.4 Fortgeschrittene Übungen

Die folgenden Übungen sind etwas schwieriger, bieten allerdings einen Lösungsvorschlag, falls du Schwierigkeiten hast sie zu lösen.

Übung 6_05: Automatisches füllen einer Liste

Ziel: Erstelle eine Liste mit allen Zahlen von 1-1000 ohne Code-Duplikate.

Schritte:

1. Erstelle eine Liste mit dem Namen “Zahlen”
2. Füge den **wiederhol () mal**-Block mit 1000 Repetitionen hinzu
3. Füge den **füge () zu (Zahlen)**-Block hinzu, mit der Länge der Liste + 1 als neuen Wert

Übung 6_06: Replay des Logs

Ziel: Spiele die Eingaben des Nutzers (siehe Übung 6_04) in der gleichen Reihenfolge wieder ab.

Schritte:

1. Erstelle eine Variable mit dem Namen “LogZeiger”
2. Wenn die Flagge gedrückt wird:
    - 2.1 Bewege deine Figur ins Zentrum (0,0)
    - 2.2 Richte die Figur in die Richtung 90° aus
    - 2.3 Setze den “LogZeiger” auf 0
3. Erstelle einen **wiederhole () mal**-Block mit der Anzahl Einträge in der Liste
    - 3.1 Überprüfe den Wert in der Liste an der Position von “LogZeiger”
    - 3.2 Bewege deine Figur um 10 Schritte
    - 3.3 Warte 1 Sekunde
    - 3.4 Füge 1 zum “LogZeiger” hinzu
4. Leere die Liste
5. Bewege deine Figur ins Zentrum (0,0)

6.5 Notizen

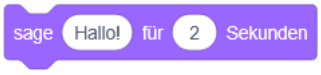
7 Aussehen

Du hast alles über die Kontrolle von Figuren innerhalb der Bühne gelernt. Dieses Kapitel zeigt dir nun, wie du mit dem Nutzer sprechen, Farben-Effekte verwenden und Figuren neu anordnen oder verändern kannst. Alle dafür notwendigen Blöcke findest du in der Kategorie Aussehen.

7.1 Mit dem Nutzer sprechen

Wir haben gelernt, wie man die Figur kontrollieren kann, aber nicht wie man den Nutzer informiert, wie er ihn kontrollieren kann. Scratch bietet dafür einige Optionen in der Kategorie Aussehen. Lass uns mit dem Block **sage ()** starten. Dieser Block öffnet eine Sprechblase in der oberen, rechen Ecke der Figur mit dem übergebenen Argument als Nachricht. Für die Nachricht kannst du alle Symbole verwenden (Buchstaben, Zahlen, Emoji, etc.) wie auch Werteblöcke, wie Variablen oder Ergebnisse von Berechnungen. Während die Nachricht dem Nutzer angezeigt wird, läuft das Skript automatisch weiter. Die Sprechblase verschwindet erst, wenn eine neue Sprech- oder Gedankenblase auftaucht oder eine leere Sprechblase erstellt wird. Falls du die Sprechblase nur temporär anzeigen möchtest, kannst du den Block **sage () für () Sekunden** verwenden. Dabei wird das Skript für die Anzeigedauer der Sprechblase pausiert.

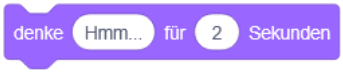
Sage () Stapelblock



Sage () für () Sekunden Stapelblock

Denke () Stapelblock

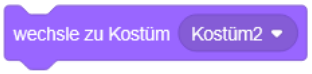
Anstelle einer Sprechblase kann auch eine Gedankenblase mittels **denke ()** und **denke () für () Sekunden** gleich wie die entsprechenden **sage ()**-Blöcke verwendet werden. Diese Gedankenblasen sind nützlich, wenn du dem Nutzer Hinweise oder Tipps geben möchtest.



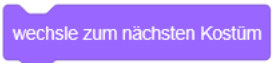
Denke () für () Sekunden Stapelblock

7.2 Das Aussehen verändern

Lass uns über Kostüme im Kostüm-Tab sprechen. Jede Figur besteht aus mindestens einem Kostüm (”costume"). Falls eine Figur mehr als ein Kostüm hat, kannst du das sichtbare Kostüm mittels Code-Blöcken ändern. Dies ist für Animationen sehr nützlich. Wir können beispielsweise bereits die Scratch-Katze nach rechts bewegen, das ist nicht schwierig. Allerdings sollte sie auch die Beine bewegen. Wenn du bei der Scratch-Katze auf die Kostüme klickst, siehst du zwei Kostüme. Mittels des **wechsle zu Kostüm ()**-Blocks kann das Kostüm gewechselt werden. Als Argument wird ein Werteblock mit der Nummer oder dem Namen des Kostüms angegeben. Mittels des folgenden Algorithmus können wir nun die Beine der Katze animieren:



Wechsle zu Kostüm () Stapelblock



Nächstes Kostüm Block

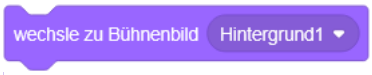
Kostüm () Werteblock

1. Wir setzen Kostüm mittels eines **wechsle zu Kostüm (Kostüm 1)**
2. Wir lassen die Katze nach rechts bewegen mittels eines **gehe (10) er Schritte**-Blocks
3. Dann warten wir kurz mittels eines **warte (0.05) Sekunden**-Blocks
4. Wir wechseln auf das zweite Kostüm mittels eines **wechsle zu Kostüm (Kostüm 2)**-Blocks.
5. Wir lassen die Katze nach rechts bewegen mittels eines **gehe (10) er Schritte**-Blocks
6. Dann warten wir kurz mittels eines **warte (0.05) Sekunden**-Blocks
7. Wir gehen zurück zu Schritt 1

Das Resultat ist eine natürliche Gehbewegung einer Katze.

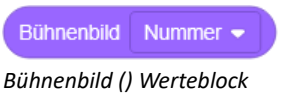
Vielleicht hast du gemerkt, dass man das vereinfachen kann, da der Algorithmus immer das nächste Kostüm auswählt. Wir können statt dem manuellen Wechsel zwischen den beiden Kostümen auch den Block **wechsle zu nächstem Kostüm** verwenden.

Falls du den Namen oder die Nummer des aktuellen Kostüms benötigst, kannst du den Werteblock **Kostüm ()** verwenden. Dies ist nützlich, wenn eine Figur mehrere Kostüme hat und du wissen möchtest, an welchem Punkt die Animation aktuell ist.



Wechsle zu Bühnenbild () Stapelblock
Nächstes Bühnenbild Block

Unabhängig davon, ob du ein Spiel, eine Geschichte oder eine Animation erstellst, möchtest du vermutlich verschiedene Hintergründe verwenden.

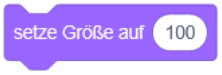


Bühnenbild () Werteblock

Scratch bietet dafür unten rechts im Panel Bühne die Hintergründe. Diese funktionieren ähnlich wie die Kostüme und du kannst die Hintergründe mithilfe von Blöcken **wechsle zu Bühnenbild ()** und **wechsle zu nächstem Bühnenbild** ändern, sowie mit dem Block **Bühnenbild ()** auf den aktuellen Hintergrund zugreifen.

Du kannst nun das Kostüm und den Hintergrund verändern. Falls du einmal die Grösse einer Figur verändern willst, kannst du mittels des Blocks **ändere Grösse um ()** die Grösse relativ zur aktuellen Grösse verändern. Bei positiven Zahlen wächst deine Figur, bei negativen schrumpft sie. Du kannst auch die Grösse relativ zur Originalgrösse mittels **setze Grösse auf ()** in Prozent setzen. Achtung: Wenn du eine Figur mit schlechter Auflösung hast, kann die Figur verpixelt aussehen wenn du diese zu stark wachsen lässt.

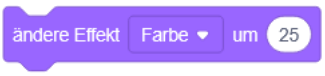
Ändere Grösse um () Stapelblock



Setze Grösse auf () Stapelblock

Grösse Werteblock

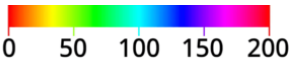
Es gibt noch mehr, was du am Aussehen ändern kannst. Mit den Blöcken **ändere Effekt () um ()** und **setze Effekt () auf ()** kannst du visuelle Effekte auf eine Figur anwenden. Das erste Argument ist dabei der gewünschte Effekt und das zweite den Wert des Effekts. Die folgenden Effekte stehen zur Verfügung:



Ändere Effekt() um () Stapelblock

Setze Effekt () auf () Stapelblock

Farbe: Dieser Effekt ändert die Farbe der Figur. Jedes Kostüm kann bis zu 200 Farben haben. Die initiale Farbe der Figur hat den Wert 0. Wenn deine Figur rot ist, wird er beim Wert 100 eine andere Farbe haben. Das Spektrum siehst du im Bild rechts. Schwarz und Weiss verändern sich nicht.



Hue Farbspektrum

Fischauge: Dieser Effekt bewirkt eine Verzerrung deiner Figur, als ob du ihn durch ein Weitwinkel-Objektiv anschaust. Der Wert kann von –100 bis 100 gehen. Probiere ihn aus!

Wirbel: Dieser Effekt verzerrt und verdreht deine Figur um seinen Mittelpunkt. Dieser Effekt verhält sich unterschiedlich, je nach System und Browser, die Nutzung wird entsprechend nicht empfohlen.

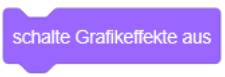
Pixel: Dieser Effekt verpixelt eine Figur. Je nach Wert werden die resultierenden Pixel grösser oder kleiner.

Mosaik: Dieser Effekt erstellt ein Mosaik aus deiner Figur, bei welchem die Steine wieder aus der Figur selbst bestehen. Je höher der Wert, desto mehr / kleinere “Steine” werden genutzt.

Helligkeit: Dieser Effekt verändert die Helligkeit deiner Figur. Der Wert geht von –100 bis 100, wobei 0 der Anfangswert ist. Bei –100 ist das Kostüm komplett schwarz und bei 100 komplett weiss.

Durchsichtigkeit: Dieser Effekt verändert die Transparenz der Figur. Der Wert setzt die Transparenz von 0 (komplett sichtbar) bis 100 (unsichtbar). Die Figur ist bei 100 zwar unsichtbar, aber immer noch da. Dies ist nützlich bei Spielen, beispielsweise für unsichtbare Blöcke oder Truhen.

Du kannst mehrere visuelle Effekte auf der gleichen Figur oder Kostüm anwenden. Um alle Effekte zurückzusetzen, kannst du den **schalte Grafikeffekte aus**-Block verwenden. Unsicher was die Effekte machen? Probiere sie aus!



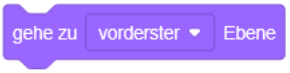
Schalte Grafikeffekte aus Stapelblock

7.3 Ab ins Rampenlicht!

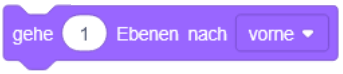
Der Durchsichtigkeit-Filter verändert die Transparenz einer Figur, aber diese ist immer noch da, auch wenn unsichtbar. Falls du eine Figur komplett verstecken möchtest, kannst du den Block **verstecke dich** verwenden. Um ihn wieder anzuzeigen kannst du den Umkehrblock **zeige dich** verwenden. Wenn du herausfinden möchtest, ob deine Figur unsichtbar oder versteckt ist, kannst du die Icons unter dem

Zeige/Verstecke dich

Figur Namen in den Figureigenschaften anschauen. Falls das linke (blaue) Icon ausgewählt ist, ist die Figur aktiv, ansonsten ist sie versteckt.



Gehe zu () Ebene Stapelblock



Gehe () Ebenen nach () Stapelblock

Scratch bietet auch Blöcke für die Neu-Anordnung von Figuren. Standardmässig ist die als letztes hinzugefügte Figur am vordersten. Um eine Figur ganz nach vorne zu bewegen kann der Block **gehe zu () Ebene** mit dem Argument ”vorderster” verwendet werden. Um ihn ganz nach hinten zu verschieben, kann der gleiche Block mit dem Argument ”hinterster” verwendet werden. Bei vielen Schichten kann die Figur mittels des **gehe () Ebenen nach ()** die gewünschte Anzahl Schichten nach vorne oder hinten verschoben werden. Mittels der zwei Blöcke kann eine Figur zu einer völlig anderen Schicht als der ursprünglichen verschoben werden. Entsprechend ist es wichtig im Blick zu behalten, was sich wann auf welcher Schicht befindet. Prinzipiell fängt man von hinten an zu zählen. Die hinterste Schicht ist die erste, die zweit-hinterste die Zweite und so weiter. Um beispielsweise auf die dritte Schicht zu gelangen, schickt man die Figur erst mit **geh zu (hinterster) Ebene** ganz nach hinten und denn mit **gehe (3) Ebenen nach (vorne)** 3 Schichten nach vorne.

7.4 Übungen

Übung 7_01: Begrüsse den Nutzer, wenn das Spiel startet

Übung 7_02: Erstelle zwei Knöpfe und verstecke sie, nachdem sie geklickt wurden

Übung 7_03: Füge die “walking bear”-Figur von der Scratch-Bibliothek hinzu. Erstelle eine Geh-Animation für 240 Schritte, mit einem Kostümwechsel alle 10 Schritte und einer Wartezeit von jeweils 0.05 Sekunden.

Übung 7_04: Füge die “bat”-Figur von der Scratch-Bibliothek hinzu. Erstelle eine unendlich andauernde Flug-Animation mit einer Wartezeit von jeweils 0.1 Sekunden.

Übung 7_05: füge die “dinosaur2”-Figur von der Scratch-Bibliothek hinzu und ändere die Farbe mit Effekten zu blau, und erstelle ein Mosaik aus 4 Dinosauriern.

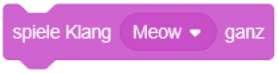
7.5 Notizen

8 Klänge

Du hast gelernt, wie man grafische Effekte anwendet, um dein Projekt farbenfroher zu gestalten. Von den 5 Sinnen des Menschen haben wir entsprechend das “Sehen” abgeschlossen. Lass uns weitergehen zum “Hören” und deinem Projekt Klänge hinzufügen. Alle dafür notwendigen Blöcke findest du in der Kategorie Klänge.

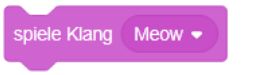
8.1 Klänge kontrollieren

Was macht den Unterschied in einem Spiel, einer Animation oder einer Geschichte? Genau, der Ton. Nicht sicher? Versuch einmal einen Film ohne Ton zu schauen, um den Unterschied zu sehen.



Spiele Klang () Stapelblock

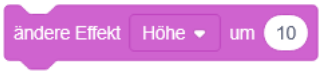
Zuerst müssen wir den Klang, den wir später spielen möchten mithilfe des lang-Editors hinzufügen. Wähle einen Klang aus der Scratch-Bibliothek, von deinem Computer oder nimm direkt einen in Scratch auf. Anschliessend kannst du den Klang sinnvoll benennen und ihn falls nötig bearbeiten. Mit dem Block **spiele Klang () ganz** kannst du den Klang abspielen. Der Block wird bei Ausführung den Klang abspielen und das Skript bis zum Ende des Klanges pausieren. Falls du nur den Klang starten möchtest, ohne das Skript zu pausieren, kannst du den Block **spiele Klang ()** verwenden. Falls du den Block zweimal ausführst, startet der Ton wieder von vorne. Du kannst ihn auch mittels des **stoppe alle Klänge**-Block stoppen.



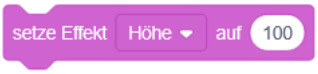
Spiele Klang () Stapelblock

Stoppe alle Klänge Stapelblock

Scratch bietet auch einige Soundeffekte. Mit den Blöcken **ändere Effekt () um ()** und **setze Effekt () auf ()** in der Kategorie Klang können diese Effekte genutzt werden. Diese Blöcke sollten nicht mit den gleichnamigen Blöcken in der Kategorie Aussehen verwechselt werden, welche für Figuren oder deren Kostüme verwendet werden.



Ändere Effekt () um () Stapelblock



Setze Effekt () au () Stapelblock

Die folgenden Effekte können ausgewählt werden:

Höhe: Der Höhe-Effekt ändert die Tonhöhe. Eine Änderung von 10 entspricht einem Halbton. Um eine ganze Oktave hoch oder runterzugehen, geht man entsprechend 120 Halbtöne hoch oder runter. Um nach unten zu gehen, wird die Zahl negativ notiert, beispielsweise um einen Halbton runterzugehen würde man –10 angeben. Die Tonhöhe kann um –360 bis +360 geändert werden.

Aussteuern: Der Aussteuern-Effekt verschiebt den Ton von links (left) nach rechts (right) oder umgekehrt. Bei einem positiven Wert wird der Ton nach rechts verschoben, bei einem negativen Wert nach links. Der Wert kann zwischen -100 und 100 liegen, wobei bei beispielsweise 100 der Ton nur noch rechts hörbar ist.

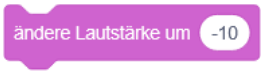
Beide Effekte können gleichzeitig genutzt werden. Um die Effekte wieder zu entfernen kann der Block **schalte Klangeffekte aus** genutzt werden. Es gibt noch weitere Effekte im Klangeditor, beachte allerdings, dass sich die Effekte des Klangeditors auf einen einzelnen Klang auswirken und es nicht möglich ist, sie während einer Animation oder eines Spiels zu ändern.

Schalte Klangeffekte aus Stapelblock

Ein wichtiger Aspekt von Klängen ist die Lautstärke. Die aktuelle Lautstärke ist im **Lautstärke**-Block gespeichert. Der Wert reicht von 0 (Stumm) bis 100 (maximale Lautstärke). Die Lautstärke gilt für alle Klänge und ist nur global veränderbar. Es ist nicht möglich, eine Lautstärke pro Ton festzulegen.

Lautstärke Werteblock

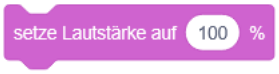
Die Lautstärke kann mittels der Blöcke **ändere Lautstärke um ()** und **setze Lautstärke auf ()%** festgelegt werden, wobei ersterer die aktuelle Lautstärke um die Eingabe erhöht (positive Zahlen) / senkt (negative Zahlen).



Ändere Lautstärke um () Stapelblock

8.2 Übungen

Übung 8_01: Spiel den Klang “cave” von der Scratch-Bibliothek unendlich lange sobald “Start” gedrückt wird.



Setze Lautstärke auf () %

Übung 8_02: Spiele den Klang “Jungle” von der Scratch-Bibliothek sobald “Start” gedrückt wird und ändere die Tonhöhe um zwei Oktaven nach 1.5 Sekunden.

Übung 8_03: Spiele die Klänge “Dance Energetic” und “Drum Machine” von der Scratch-Bibliothek gleichzeitig sobald “Start” gedrückt wird.

Übung 8_04: Spiel die Klänge “Movie 2” und “Mystery” von der Scratch-Bibliothek gleichzeitig, wenn “Start” gedrückt wird. “Movie 2” sollte lauter sein als “Mystery”.

8.3 Notizen

9 Fühlen

Wir haben im Voraus über die 5 Sinne des Menschen gesprochen, aber auch unsere Figuren haben Sinne. Dieses Kapitel beschreibt, wie du in Scratch Fühlen kannst. Alle dafür notwendigen Blöcke findest du in der Kategorie Fühlen.

9.1 Berührungen

Entfernung von () Werteblock

Du hast bereits einiges über die visuellen Optionen in Scratch gelernt. Wir gehen nun wieder in die Programmierung und sprechen über Berührungen. Um zu überprüfen, ob sich zwei Sachen berühren, kann das Pythagoras-Theorem verwendet werden. Dabei wird die Distanz zwischen zwei Punkten mittels Operatoren berechnet dessen Resultat anschliessend entsprechend interpretiert werden kann. In Scratch kann hierzu der Block **Entfernung von ()** verwendet werden. Dieser Block berechnet den Abstand zwischen den Mitten zweier Figuren. Mittels des Dropdown-Menüs können die entsprechenden Figuren sowie der Mauszeiger ausgewählt werden. Falls du ein ungültiges Argument wählst, wird als Distanz der Wert 10'000 angegeben. Der Block funktioniert unabhängig davon, ob die Figur sichtbar ist oder nicht.

Wird () berührt? Wahrheitsblock

Mittels etwas fortgeschrittener Berechnungen kannst du herausfinden, ob sich zwei Objekte berühren. Da diese Funktionalität in Scratch oft benötigt wird gibt es dafür einen eigenen Block **wird () berührt?.** Gültige Werte für die Argumente können im entsprechenden Dropdown gefunden werden. Dazu zählt der Mauszeiger, Kanten oder eine andere Figur. Der Block gibt wahr zurück, falls die visuelle Textur der Figur das jeweilige Objekt berührt. Falls eine Figur beispielsweise ein Kreis ist, gibt der Block wahr zurück, falls die Kante oder der Kreis selbst das andere Objekte berührt und nicht wenn der rechteckige Rahmen um die Figur das andere Objekt berührt. Der Block gibt falsch zurück, wenn die Figur das andere Objekt nicht berührt oder das andere Objekt versteckt ist. Achtung: Wenn du den Durchsichtigkeit-Filter verwendest, ist das andere Objekt zwar nicht sichtbar, aber weiterhin erkennbar, entsprechend wird der Block wahr zurückgeben.

Wird Farbe berührt? Wahrheitsblock

Manchmal soll sich eine Figur über eine bestimmte Farbe bewegen. Hierzu bietet Scratch den Block **wird Farbe () berührt?**. Der Block gibt wahr zurück, wenn die Figur die gewählte Farbe berührt, und falsch in allen anderen Fällen. Der Block ist langsamer als der **wird () berührt?**-Block und entsprechend nicht für das Erkennen von Berührungen geeignet. Hinzu kommt, dass er etwas unpräziser ist, da die Bühne über 16 Millionen Farben darstellen kann, aber der Block nur eine relativ kleine Anzahl an Farben überprüft und somit einen grosse Farbspanne besitzt.

Farbe () berührt ()? Wahrheitsblock

Für noch detaillierte Resultate kann der Block **Farbe () berührt ()?** verwendet werden, welcher wahr zurück gibt sobald die gewählte Farbe innerhalb der Figur die zweite Farbe berührt. Der Block besitzt allerdings die gleichen Einschränkungen wie der **wird Farbe () berührt?** Block.

9.2 Den Nutzer fragen

Frage () und Warte Stapelblock

Antwort Werteblock

In Scratch kann die Figur über die Blöcke **sage ()** oder **denk ()** eine Information weitereben. Manchmal möchte man allerdings auch eine Frage stellen, um beispielsweise den Namen des Nutzers zu erfahren Scratch bietet dafür den Block **frage () und warte**. Dieser Block erzeugt ein Eingabefeld am unteren Rand der Bühne welches vom Nutzer befüllt werden kann. Das erste Argument des Blocks ist die Frage, welche dem Nutzer gestellt wird. Die Antwort des Nutzers wird im **Antwort**-Block gespeichert. Letzterer wird bei einer neuen Antwort überschrieben. Falls du die Antwort länger benötigst, kannst du sie in einer Variable oder Liste abspeichern.

9.3 Tastatureingaben und Mausbewegungen

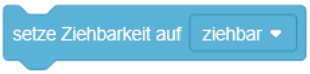
Taste () gedrückt? Wahrheitsblock

In früheren Kapiteln haben wir gelernt, wie man ein Skript starten kann, wenn eine Taste gedrückt wird. Die Kategorie Fühlen bietet den Block **Taste () gedrückt?** um zu überprüfen, ob eine bestimmte Taste gedrückt ist. Standardmässig kannst du hier alle Buchstaben von A-Z auswählen. Falls du mehr Buchstaben benötigst, verwende einen Werteblock, der deinen Buchstaben enthält. Der Block kann auch nützlich sein, um deinen Code zu strukturieren und die Anzahl Scripts zu minimieren, oder falls mehrere Tasten gleichzeitig gedrückt werden müssen.

Maustaste gedrückt?

Maus X und Y Position Werteblöcke

Falls du überprüfen möchtest, ob eine Maustaste gedrückt ist (während der Mauszeiger auf der Bühne ist) kannst du den Block **Maustaste gedrückt?** verwenden. Mit den Werteblöcken **Maus-x-Position** und **Maus-y-Position** kannst du die Position der Maus verwenden. Diese Kombination von Blöcken erlaubt es dir zu erkennen, ob eine Figur angeklickt wird, indem du die x- und y-Koordinaten der Maus mit den Koordinaten der Figur abgleichst oder wenn du eine Figur mit der Maus bewegen möchtest.



Setze Ziehbarkeit auf () Stapelblock

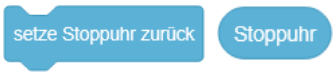
Falls du eine Figur bewegen möchtest, musst du den “drag and drop“-Modus mittels des **setze Ziehbarkeit auf ()**-Blocks auf ”ziehbar” setzen (oder auf ”nicht ziehbar” falls er nicht bewegt werden soll). Dieser Effekt funktioniert nur im Vollbild-Modus. Du kannst diesen Block auch unter bestimmten Bedingungen einsetzen, beispielsweise um eine Figur bewegbar zu machen, wenn eine Taste gedrückt ist oder basierend auf dem Wert einer Variable.

9.4 Umgebungslautstärke

Lautstärke Werteblock

Falls du mit einer etwas innovativeren Methode deine Figur bewegen möchtest, kannst du auch den Lautstärke-Block verwenden. Dieser Block benötigt ein Mikrofon, mit welchem er die Umgebungslautstärke misst. Falls kein Mikrofon angeschlossen ist, gibt der Block den Wert –1 zurück. Ansonsten geht der Wert von 0 bis 100, abhängig von der Lautstärke der Umgebung (je höher der Wert, desto lauter die Umgebung). Du kannst dies beispielsweise verwenden, um Bewegungen schneller zu machen je höher die Lautstärke ist, respektive langsamer, je leiser die Lautstärke ist.

9.5 Datum und Zeit



Blöcke der Stoppuhr

Manche Projekte messen die Zeit, welche ein Nutzer benötigt, um etwas abzuschliessen. Dazu kann man in Scratch eine Stoppuhr verwenden. Die Stoppuhr zählt die Anzahl Sekunden ab dem Klicken der grünen Flagge oder seit dem letzten Zurücksetzen mittels des **setze Stoppuhr zurück**-Blocks. Der Wert der Stoppuhr wird einmal pro Frame aktualisiert, also jede 30-stel Sekunde. Der Wert der Stoppuhr kann mittels dem **Stoppuhr**-Block abgefragt werden.

Zeitbezogene Blöcke

Manchmal ist es wichtig zu wissen, welcher Tag es ist. Dazu gibt es den Block **Tage seit 2000**, welcher die Anzahl Tage seit dem ersten Januar 2000 in der Zeitzone UTC zurückgibt. Für die aktuelle Zeit ist es auch möglich, den **() im Moment**-Block zu verwenden.

9.6 Werte anderer Objekte

() von () Werteblock

Scratch bietet einen mächtigen Block **() von ()**. Der Block nimmt als erstes Argument eine Eigenschaft, wie beispielsweise die x- oder y-Position und das zweite Argument eine Figur oder ein Bühnenbild. Der Block gibt den Wert der Eigenschaft für die gewählten Figur respektive das gewählte Bühnenbild zurück. Dies kann beispielsweise für vergleiche von Koordinaten oder die Überprüfung des Zustands eines anderen Objekts verwendet werden.

9.7 Übungen

Übung 9_01: Frage nach dem Namen des Nutzers und warte auf die Antwort. Begrüsse den Nutzer, nachdem du seinen Namen erhalten hast mit seinem Namen.

Übung 9_02: Bewege deine Figur nach rechts und lasse sie anhalten, sobald sie den Rand berührt.

Übung 9_03: Kontrolliere die Bewegungen der Figur mit den Pfeiltasten. Verwende ein einzelnes Skript für diese Aufgabe, respektive verwende nicht den **Wenn Taste () gedrückt wird**-Block.

Übung 9_04: Erstelle ein pulsierendes Herz dessen Frequenz abhängig von der Umgebungslautstärke ist.

Übung 9_05: Zeige die Aktuelle Zeit im Format Jahr-Monat-Tag Stunden:Minuten:Sekunden.

9.8 Notizen

Anhang I – Spickzettel

Rotation

Eine Figur kann um volle 360° rotiert werden. Allerdings wird in Scratch der Wert je nach Rotation positiv oder negativ angezeigt, abhängig davon in welche Richtung rotiert wird. Der Winkel wird im Uhrzeigersinn gerechnet, wobei 0° nach oben zeigen. Bei 90° wird die Figur um 90° nach rechts gedreht. Das ist gleichwertig mit -270° (270° gegen den Uhrzeigersinn).

Richtung -90 oder 270 Grad

Richtung 90 oder -270 Grad

Rotationsmodus

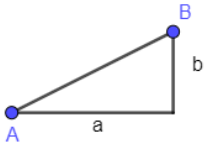
Scratch bietet drei Rotations-Modi:

Rundherum: Die Figur dreht in die gewünschte Richtung

Left-right: Die Figur zeigt nach rechts wenn der Wert zwischen 0 und 180 ist, ansonsten nach links.

Nicht drehen: Die Figur zeigt immer nach links oder rechts, unabhängig der Punkt-Richtung

Abstand zwischen zwei Punkten



Um den Abstand zwischen zwei Punkten zu berechnen, wird der eingebaute Werteblock **Entfernung von ()** in der Kategorie Fühlen verwendet. Falls du den Abstand zwischen zwei Punkten berechnen musst, kannst du auch das Pythagoras Theorem wie folgt anwenden:

1. Berechnen der Distanz zwischen A und B mittels des folgenden Scratch-Code:

2. Berechnen der Distanz mit dem Pythagoras-Theorem mittels des folgenden Scratch-Code:

Figur Kollision

Die Bühne in Scratch ist 480 Pixel breit und 360 Pixel hoch. Um zu überprüfen, ob sich zwei Figuren berühren, kannst du den **wird () berührt?** Block in der Kategorie Fühlen verwenden.

Um zu überprüfen, ob eine Figur am Rand befindet, kannst du den minimalen und maximalen Wert an der horizontalen und vertikalen Achse überprüfen. Füge die halbe Figur-Breite, respektive Höhe hinzu oder entferne sie, um zu schauen, ob der Rand geschnitten wurde.

Nachrichten

Mittels Nachrichten (“messages”) können alle anderen Figuren benachrichtigt werden. Eine Nachricht kann mittels des **sende () an alle** Block in der Kategorie Ereignisse verwendet werden. Wenn du **sende () an alle und warte** verwendest, wartet der Versender der Nachricht, bis alle Empfänger die Nachricht verarbeitet haben. Die Empfänger verwenden dazu den **Wenn ich () empfange**-Block.

Quellen

Informationen und Medien stammen aus dem offiziellen Scratch Wiki oder wurden eigens für die Dokumentation erstellt. Das Scratch Wiki kann unter dem folgenden Link aufgerufen werden: https://en.scratch-wiki.info/wiki/Scratch_3.0