FRANZIS

Programmieren lernen mit Scratch

So wirst du ein **echter Scratcher** der weltweiten **Scratch-Community**

CHRISTIAN IMMLER

Bibliografische Information der Deutschen Bibliothek

Die Deutsche Bibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte Daten sind im Internet über *http://dnb.ddb.de* abrufbar.

Der Autor
Christian Immler, Jahrgang 1964, war bis 1998 als Dozent für Computer Aided Design an der Fachhochschule Nienburg und an der University of Brighton tätig. Einen Namen hat er sich mit diversen Veröffentlichungen zu Spezialthemen wie 3D-Visualisierung, Smartphone-Betriebssysteme, Linux und Windows gemacht. Seit mehr als 20 Jahren arbeitet er als erfolgreicher Autor mit mehr als 200 veröffentlichten Computerbüchern.

Lektorat: Ulrich Dorn
Satz: PC-DTP-Satz und Informations GmbH, Alexandra Kugge
Covergestaltung: Julia Harrer
ISBN: 978-3-645-20672-3

Inhalt

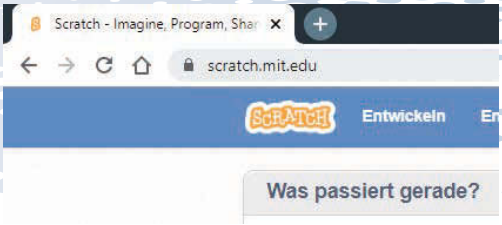
Inhalt

So geht Scratch

Bevor wir mit den ausführlichen Erklärungen beginnen, wie Scratch funktioniert und was man alles damit machen kann, probiere doch dieses kleine einfache Beispiel aus.

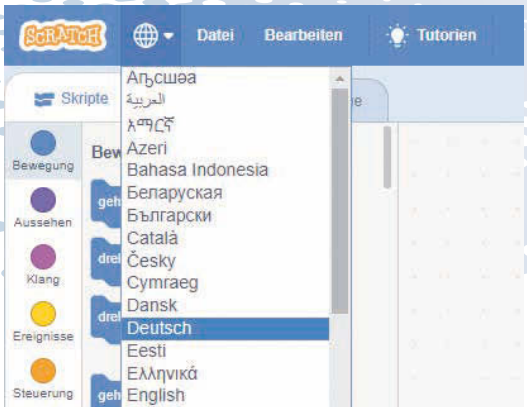
1 Starte einen Webbrowser (z. B. Firefox oder Chrome) auf dem PC und besuche die Webseite ***scratch.mit.edu***.

2 Klicke oben links auf ***Entwickeln***. Damit kannst du ein neues Scratch-Programm erstellen.

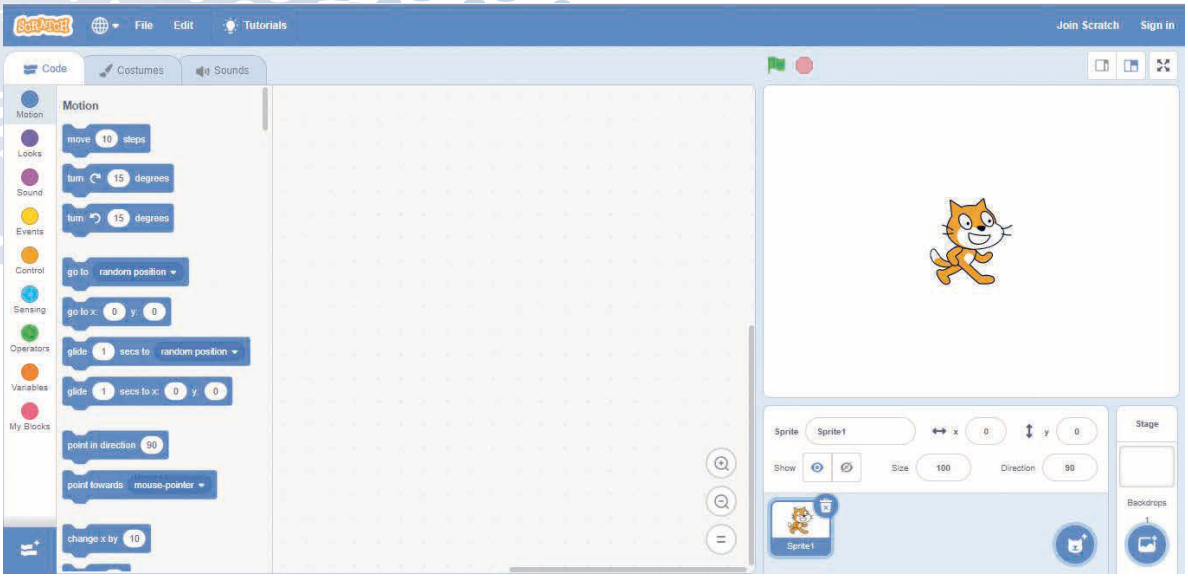


3 Jetzt erscheint die Scratch-Programmieroberfläche mit der typischen Katze, der Symbolfigur von Scratch.

4 Klicke links oben auf die Weltkugel und wähle ***Deutsch*** als Sprache für die Benutzeroberfläche.



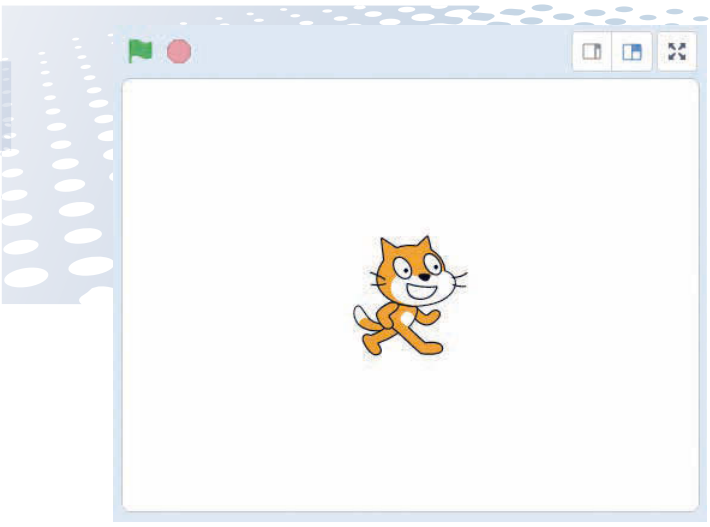
5 In einem ganz einfachen Skript soll die Katze einmal im Kreis herum laufen und dabei ihre Farbe verändern.



6 Klicke in der Blockpalette ganz links auf das gelbe Symbol ***Ereignisse***. Dann werden die Blöcke angezeigt, die auf bestimmte Ereignisse reagieren.

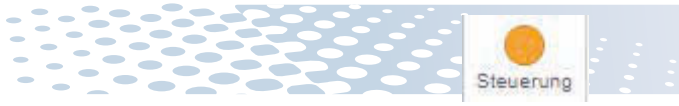
7 Ziehe den abgebildeten Block ***Wenn Fahne angeklickt wird*** aus der Blockpalette in das Skriptfenster in der Mitte.

8 Auf der Scratch-Bühne ist oben links ein grünes Fähnchen. Es dient üblicherweise dazu, ein Programm zu starten. Die Bühne ist das Fenster rechts oben, in dem die Katze in der Mitte steht.



9 Der Block ***Wenn Fahne angeklickt wird*** bewirkt, dass die folgenden Blöcke ausgeführt werden, wenn der Benutzer auf das grüne Fähnchen klickt. Der Block ist oben rund, passt also unter keinen anderen Block. Er muss immer als Erstes gesetzt werden.

10 Klicke jetzt in der Blockpalette auf das orangefarbene Symbol ***Steuerung***.



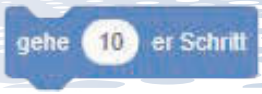
11 Die kreisförmige Bewegung der Katze wird aus einzelnen Gehen- und Drehen-Schritten zusammengesetzt. Diese werden so oft wiederholt, bis die Katze einen ganzen Kreis gegangen ist. Ziehe für die Wiederholungsschleife den Block ***wiederhole ... mal*** in das Skriptfenster und docke ihn unten an den dort bereits vorhandenen Block an. Wenn du zwei Blöcke nahe genug aneinanderziehst, verbinden sie sich automatisch.

12 In jedem Bewegungsschritt soll sich die Katze um 15 Grad drehen. Dabei dreht sie sich in 24 Schritten genau einmal. Tippe in das weiße Zahlenfeld des ***wiederhole ... mal***-Blocks und ändere den vorgegebenen Wert auf ***24***.

13 Damit die Katze eine Kreisbewegung ausführt, muss sie zunächst einen Schritt gehen, sich danach um 15 Grad drehen, wieder einen Schritt gehen usw. Um eine Bewegung zu

programmieren, klicke in der Blockpalette auf das blaue Symbol ***Bewegung***.

14 Ziehe den Block ***gehe ...er Schritt*** in das Skriptfenster und docke ihn innerhalb der Schleife an.



15 Ändere dann noch die Schrittweite auf ***20***.

16 Ziehe danach den Block für die Drehbewegung gegen den Uhrzeigersinn in das Skriptfenster.

17 Platziere ihn so über den vorhandenen Blöcken, dass er sich innerhalb der Schleife nach der Gehbewegung einklinkt.

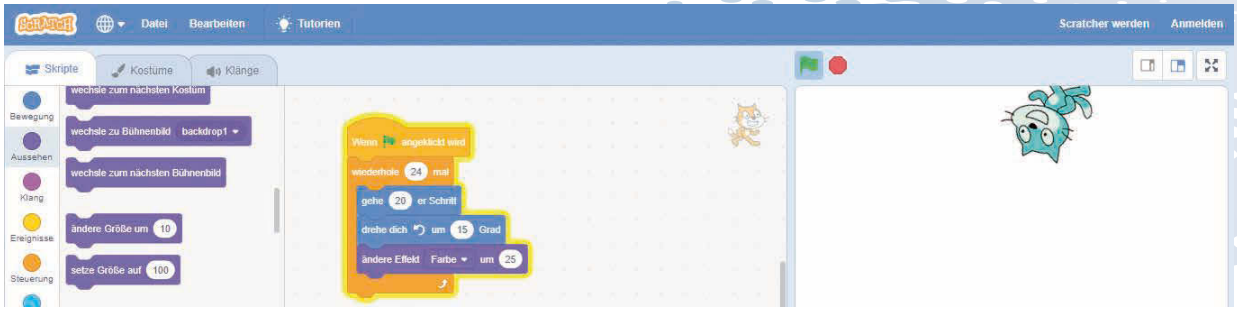
18 Das Skript sollte jetzt wie abgebildet aussehen. Probiere es aus, ob es auch wie erwartet funktioniert. Klicke dazu links oben in der Bühne auf das grüne Fähnchen. Die Katze wird eine Kreisbewegung gehen.

19 Jetzt fehlt nur noch die gewünschte Veränderung der Farbe. Klicke dazu in der Blockpalette auf das dunkelblaue Symbol ***Aussehen***. Dann werden Blöcke angeboten, die das Aussehen des aktiven Objekts (der Katze in unserem Fall) beeinflussen.

20 Ziehe den abgebildeten Block ***ändere Effekt Farbe um ...*** in das Skriptfenster in die Wiederholung hinter den ***drehe dich um ... Grad***-Block.

21 Damit wird nach jeder Bewegung die Farbe um 25 Einheiten geändert.

22 Lässt du das Skript jetzt wieder ablaufen, ändert die Katze im Laufe ihrer Bewegung zyklisch ihre Farbe. Am Ende der 24 Wiederholungen steht die Katze erneut an ihrer ursprünglichen Position und hat auch wieder ihre ursprüngliche Farbe.



0 Scratch installieren oder einfach nutzen

Was ist Scratch? Fragen wir einfach die Entwickler. Das Scratch-Team beschreibt es so: ***„Scratch ist eine Programmiersprache und Onlinegemeinschaft, in der du deine eigenen interaktiven Geschichten, Spiele und Animationen erstellen und deine Werke mit anderen überall auf der Welt teilen kannst. Beim Entwickeln und Programmieren von Scratch-Projekten lernen junge Menschen kreativ zu denken, systematisch vorzugehen und kooperativ mit anderen zusammenzuarbeiten – grundlegende Fähigkeiten für das Leben im 21. Jahrhundert.“***

In dem YouTube-Video ***What most Schools don't teach*** (Was die meisten Schulen nicht unterrichten) erklären berühmte „Tech-Helden“ wie Mark Zuckerberg (Facebook), Bill Gates (Microsoft) und andere, warum es wichtig ist, programmieren zu können: ***youtu.be/vwDSNbR-6UE***.

Scratch war 15 Jahre lang ein Projekt der Lifelong-Kindergartengruppe am MIT-Media-Lab und wurde aufgrund seines großen Erfolgs in eine eigene gemeinnützige Stiftung, die Scratch Foundation, überführt. Es ist und bleibt kostenlos.

Scratch möchte für jeden nutzbar sein, egal welche Sprache du sprichst, welche Hautfarbe oder Religion du hast, ob du ein Junge oder ein Mädchen bist. Die meisten Scratcher sind übrigens zwischen 11 und 14 Jahre alt. Im vergangenen Jahr haben über 20 Millionen Kinder und Jugendliche auf der ganzen Welt Projekte mit Scratch entwickelt.

Um möglichst viele Menschen anzusprechen, ist Scratch in 50 Sprachen verfügbar und läuft auf vielerlei verschiedenen Computern. Scratch wird in unterschiedlichen Varianten angeboten:

SCRATCH 3.0 IM BROWSER

Am einfachsten funktioniert Scratch im Browser auf einem PC. Du brauchst nichts zu installieren, sondern einfach nur die Seite ***scratch.mit.edu*** zu besuchen, und los geht's mit der aktuellsten Version 3.0 von Scratch.

SCRATCH 3.0 IM BROWSER LÄUFT:

- mit den aktuellen Versionen von Firefox (57 und höher), Chrome (63 und höher), Edge (den Chromium-basierten Versionen) auf Windows-PCs mit mindestens 1.024 × 768 Pixeln Bildschirmauflösung,
- mit den aktuellen von Firefox (57 und höher), Chrome/Chromium (63 und höher) auf den meisten Linux-Distributionen auf PCs und auf macOS,
- auf Tablets mit Android (6 und höher) oder Windows 10 mit mindestens 1.024 × 768 Pixeln Bildschirmauflösung,

- auf Chromebooks mit Chrome (63 und höher).

SCRATCH 3.0 IM BROWSER LÄUFT NICHT:

- auf Smartphones,
- auf dem Raspberry Pi,
- im Browser auf Fernsehern und Spielkonsolen,
- mit dem Microsoft Internet Explorer,
- auf Windows-RT-Tablets (nur bestimmte Effekte werden nicht unterstützt).

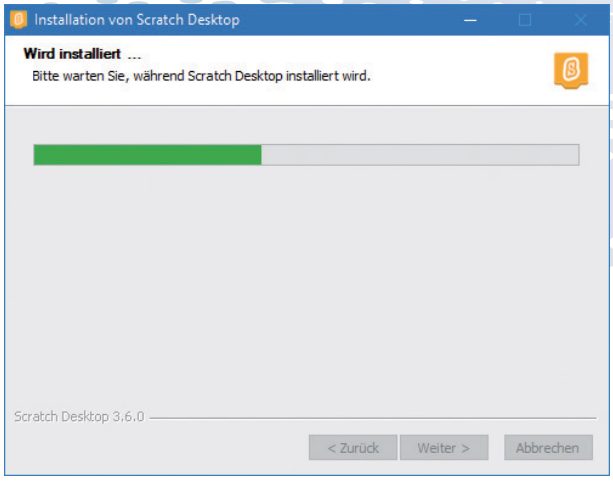
SCRATCH DESKTOP OFFLINE NUTZEN

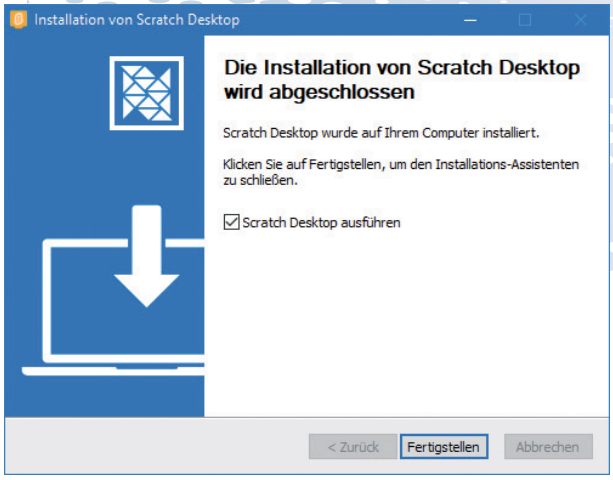
Nicht überall hat man ständig Internetzugang. Möchtest du zum Beispiel auf einer längeren Bahnfahrt etwas mit Scratch machen, kannst du dir bei ***scratch.mit.edu/download*** den Scratch Desktop herunterladen. Viele Schulen haben zwar Computer in Klassenzimmern oder Aufenthaltsräumen, die aber häufig nicht mit dem Internet verbunden sind. Auch dort kann man mit dem Scratch Desktop programmieren.

SCRATCH-3.0-DESKTOP (OFFLINE-EDITOR) INSTALLIEREN

Der Scratch-3.0-Desktop läuft offline unter Windows 10 und macOS (10.13 und höher), auf Android-Tablets (6 und höher) sowie auf aktuellen Chromebooks mit Google-Play-Store-Unterstützung, nicht aber unter Linux, nicht auf Smartphones und nicht unter iOS.

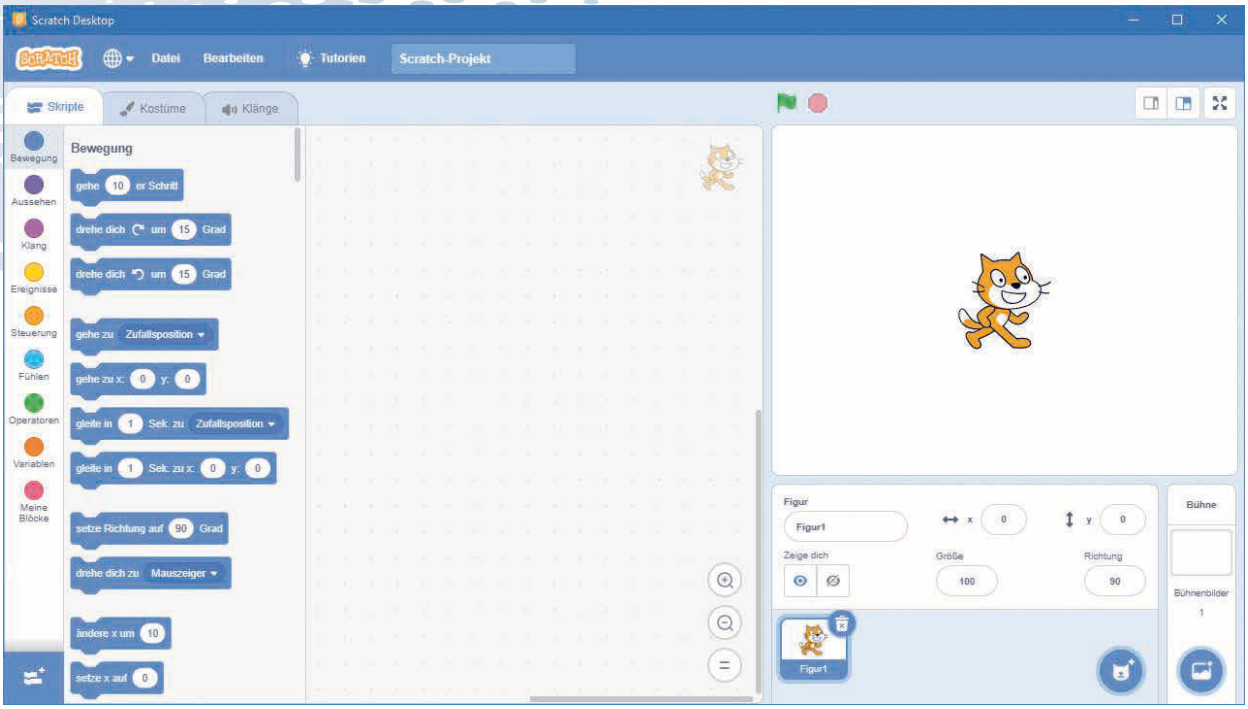
Installiere die heruntergeladene Datei auf deinem PC.





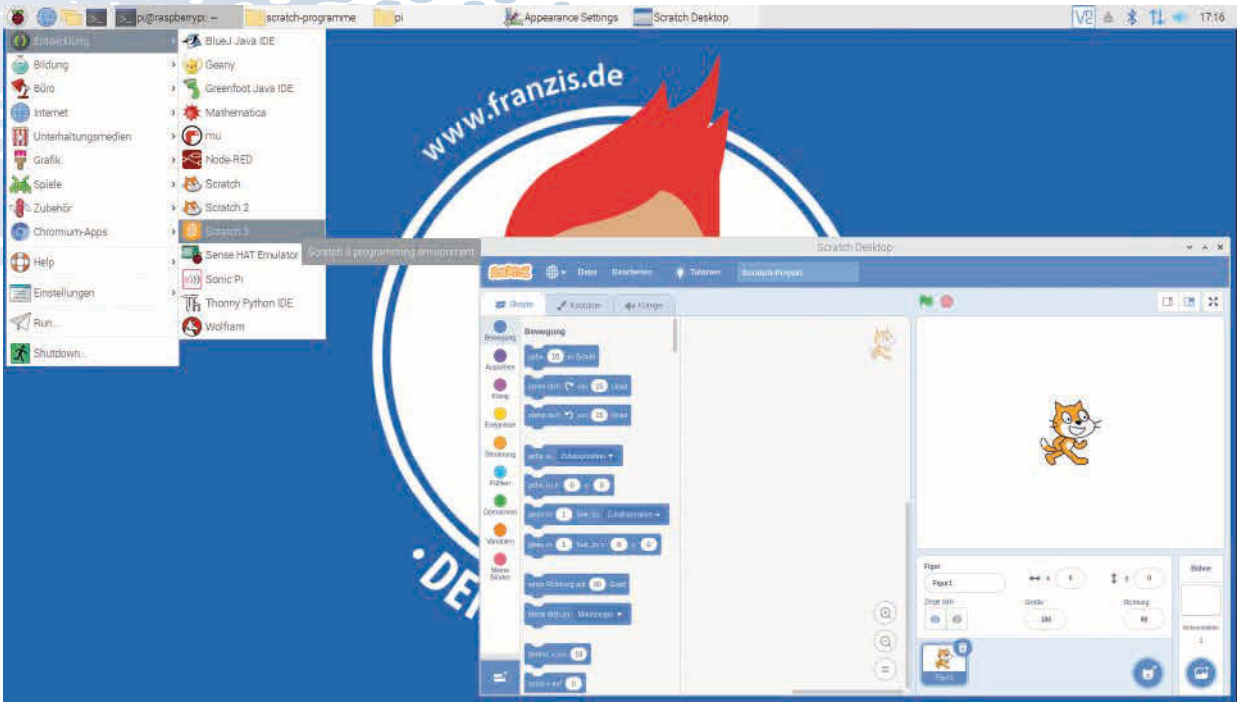
1 Scratch installieren oder einfach nutzen

Nach der Installation kannst du den Scratch Desktop ohne Internetverbindung nutzen.



SCRATCH 3.0 AUF DEM RASPBERRY PI

Auf dem Minicomputer Raspberry Pi ist die aktuelle Version Scratch 3.0 bereits im Raspbian-Betriebssystem vorinstalliert. In der neuesten Raspbian-Version gibt es standardmäßig keine Desktopsymbole mehr. Wähle hier ***Scratch 3.0*** aus dem Menü ***Entwicklung***. Der Raspberry Pi verwendet den Scratch Desktop ohne Browser. Diese Software kann offline ohne Internetverbindung genutzt werden.



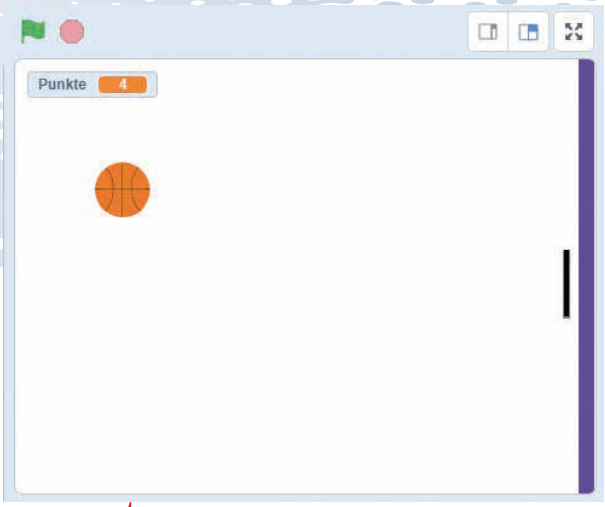
Scratch 3.0 läuft auf dem aktuellen Raspberry Pi 4 sehr flüssig. Für ältere Modelle sind auch noch die früheren Versionen Scratch 2.0 und Scratch 1.4 verfügbar. Scratch im Browser läuft auf dem Raspberry Pi nicht.

Wie Scratch auf dem Raspberry Pi genau funktioniert und wie du damit Elektronik steuern kannst, wird dir am Ende dieses Buchs beschrieben.

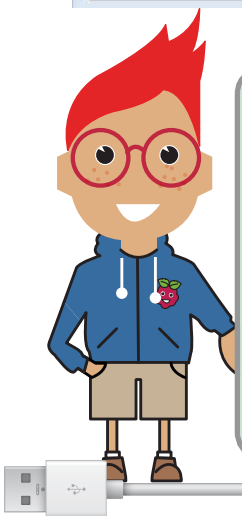
2 Das erste eigene Spiel

Computerspiele zu programmieren gehört zu den großen Herausforderungen für Profiprogrammierer. Spielthema, Grafik, Geschwindigkeit und die Spiellogik müssen optimal aufeinander abgestimmt werden, damit das Spiel ein Erfolg wird.

Scratch ist bestens dafür geeignet, Computerspiele zu programmieren, wenn auch ganz einfache. Unser nächstes Projekt ist ein klassisches Pong-Spiel, wie man es von diversen Retrokonsolen kennt. In diesem Spiel versuchst du, einen Ball, der im Raum herumfliegt, mit dem Schläger zurückzuschlagen. Wenn der Ball die farbige Linie am rechten Bildschirmrand berührt, bekommst du einen Minuspunkt, und der Ball startet in der Mitte wieder neu.



In diesem Programm verwenden wir nicht die Katze, sondern eigene Figuren, denen bestimmte Funktionsblöcke zugeordnet werden. Dabei wirst du einige wichtige Programmiertechniken kennenlernen. Wir werden schrittweise den Hintergrund, die Figuren und das eigentliche Programm zusammenbauen.



ALLE PROJEKTE ONLINE

Alle in diesem Buch beschriebenen Projekte sind online auf der Scratch-Webseite verfügbar: *scratch.mit.edu/users/franzis*.

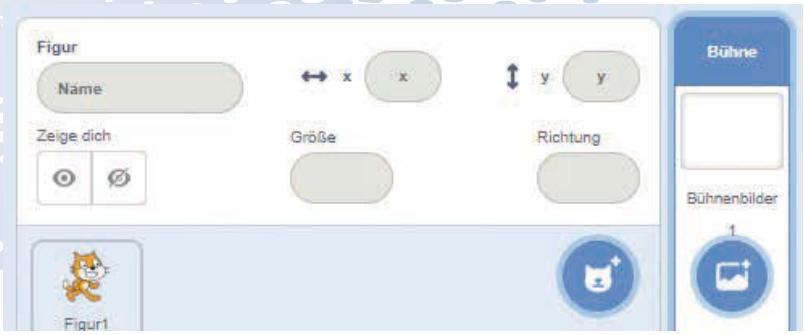
DER HINTERGRUND

1 Klicke in Scratch oben auf das Menü ***Datei*** und wähle dort ***Neu***, um mit einem ganz neuen Programm anzufangen.

2 Unten rechts findest du die Liste aller Figuren im Programm. Hier ist am Anfang nur die Katze. Ganz rechts außen siehst du ein weißes Feld ***Bühne***, das das weiße Fenster oben rechts bezeichnet, auf dem sich die Katze oder eine andere Figur bewegt.

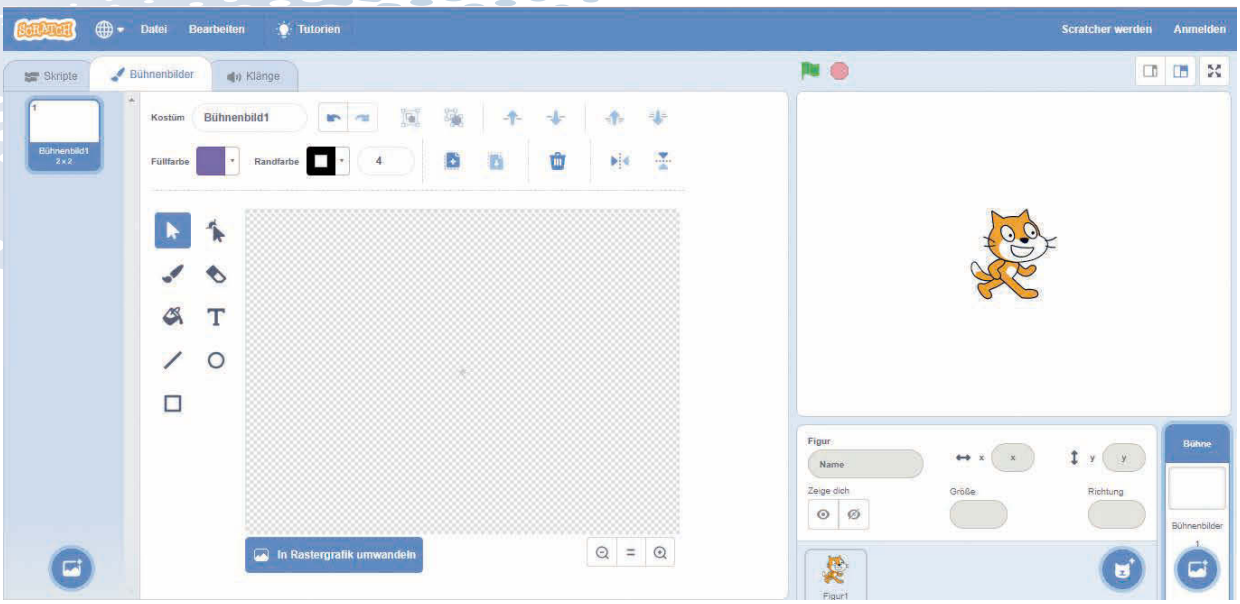
3 Klicke rechts auf das Feld ***Bühne***. Die Blöcke auf der Blockpalette ***Bewegung*** verschwinden, da sich die Bühne nicht bewegen kann. Alle

Programmblöcke in diesem Bereich gelten immer für das angezeigte Objekt.

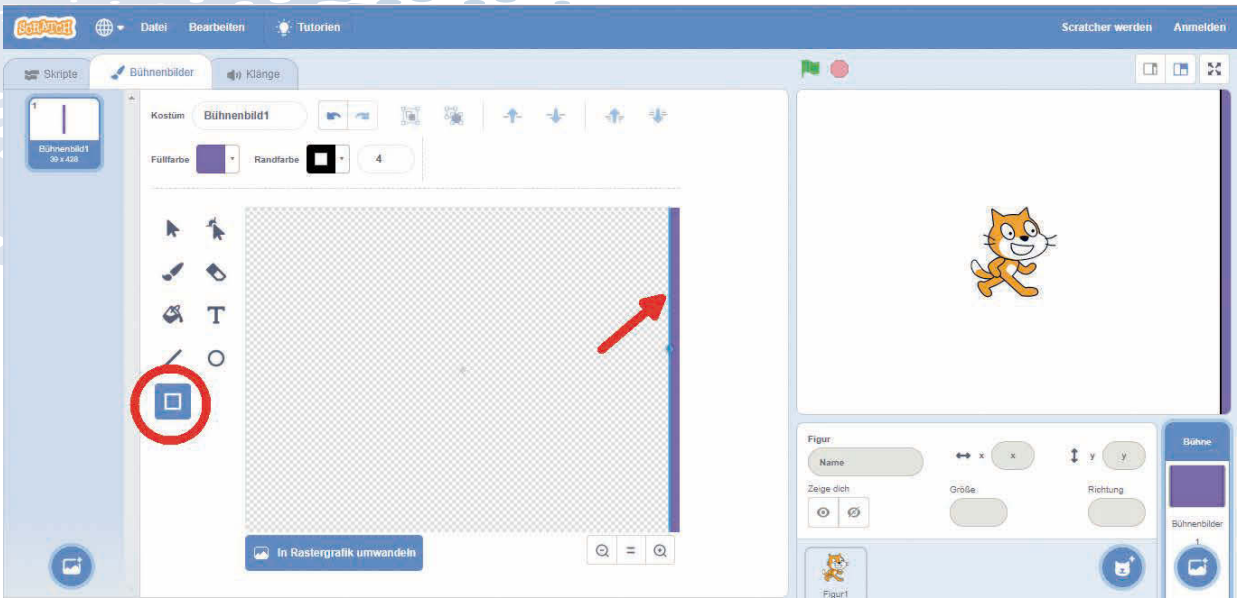


4 Klicke links oben auf die Registerkarte ***Bühnenbilder***, um den Hintergrund der Bühne zu bearbeiten. Jetzt öffnet sich innerhalb von Scratch ein einfaches Grafikprogramm, mit dem du den Hintergrund und auch andere Objekte malen kannst.

5 Wähle das Rechteck-Werkzeug aus. Wähle dann eine auffällige Farbe aus, z. B. Lila, und male damit am rechten Rand ein schmales Rechteck über die gesamte Höhe. Es erscheint auch gleich rechts oben auf der Bühne.



2 Das erste eigene Spiel

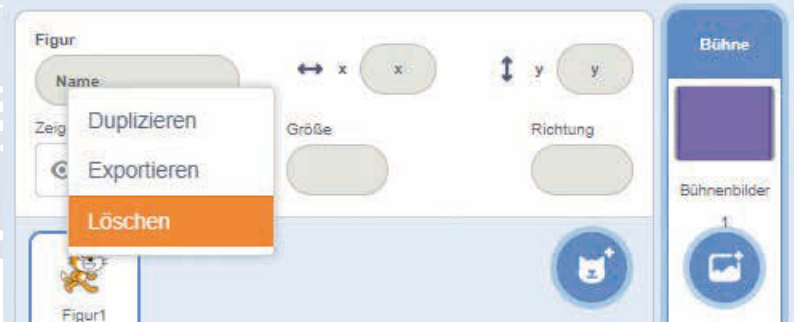


DER BALL

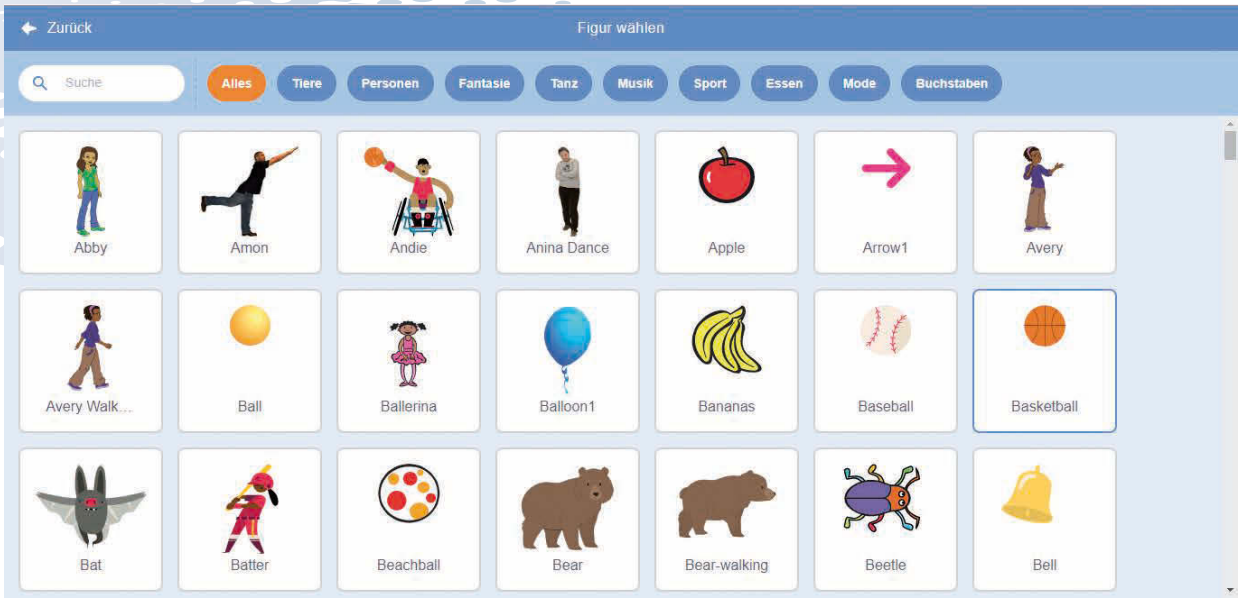
Als Nächstes brauchen wir den Ball. Natürlich könntest du auch die Scratch-Katze durch den Raum werfen, aber ein Ball sieht echter aus.

1 Lösche als Erstes die Scratch-Katze, indem du mit der rechten Maustaste auf die Figur in der Figurenpalette klickst und im Menü ***Löschen*** auswählst.

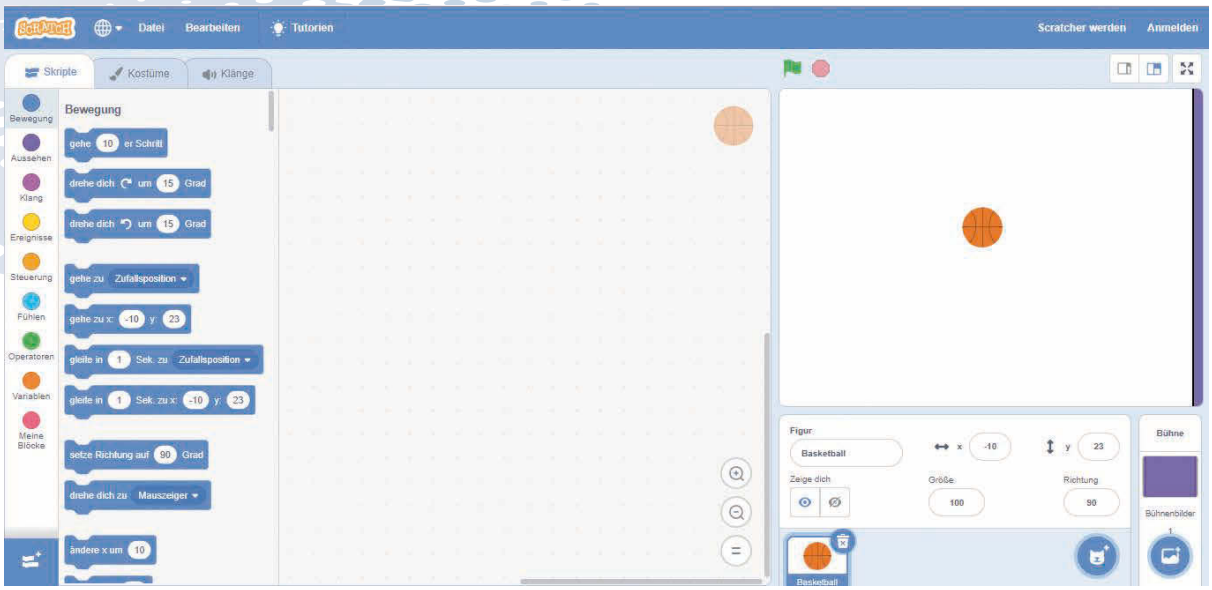
2 Klicke auf das Symbol ***Figur wählen*** in der Figurenliste.



3 Wähle in der Figurenbibliothek den ***Basketball*** aus und klicke auf ***OK***.



4 Der Ball erscheint in der Figurenliste, auf der Bühne und, da er gerade ausgewählt ist, auch oben rechts im Programmfenster. Alle neu zusammengebauten Programmblöcke gelten also jetzt für den Ball.

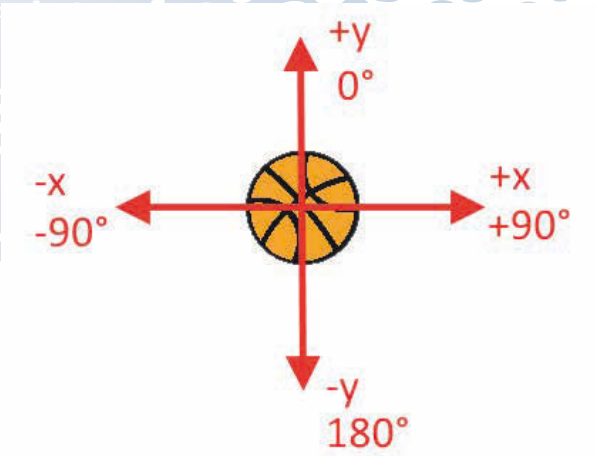


2 Das erste eigene Spiel

5 Als Nächstes bekommt der Ball seine Programmblöcke, nach denen er sich bewegen soll. Schalte dazu auf die gelbe Blockpalette ***Ereignisse*** und ziehe den Block ***Wenn Fahne angeklickt wird*** aus der Blockpalette in das Skriptfenster.

6 Beim Start des Programms soll der Ball immer in der Mitte beginnen. Die Blöcke auf der blauen Blockpalette ***Bewegung*** steuern die Bewegung von Scratch-Figuren. Wähle den Block ***gehe zu x: ... y: ...*** und trage in beide Zahlenfelder eine ***0*** ein.

7 Der Punkt ***x:0, y:0*** ist der Mittelpunkt des Koordinatensystems auf der Scratch-Bühne. Die positive x-Richtung läuft nach rechts, die negative nach links. Die positive y-Richtung läuft nach oben, die negative nach unten.

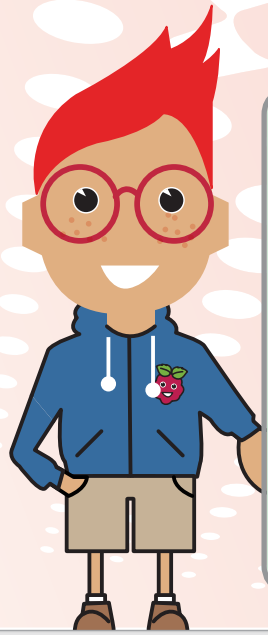


8 Der Ball soll in einem zufälligen Winkel schräg nach unten losfliegen. Dazu wird die Richtung mit einem blauen Block ***setze Richtung auf ...*** auf einen zufälligen Wert zwischen ***-20*** und ***-160*** gesetzt.

9 Dabei verwenden wir eine wichtige Technik in Scratch, nämlich Blöcke ineinander zu verschachteln. Der blaue Block ***setze Richtung auf ...*** hat ein weißes Zahlenfeld mit abgerundeten Rändern links und rechts. Das bedeutet, hier kann ein weiterer Block mit abgerundeten Rändern eingesetzt werden.

10 Ziehe den Block ***Zufallszahl von ... bis ...*** in dieses Feld und trage in die beiden Felder die Werte ***-20*** und ***-160*** ein. Übrigens haben auch diese Eingabefelder abgerundete Ränder. Hier könnten ebenfalls weitere Blöcke eingebaut werden.

11 Anschließend wird die Bewegung des Balls fortlaufend wiederholt. Er prallt vom Rand ab, sollte er ihn berühren. Andernfalls fliegt er einen 4er-Schritt in die eingestellte Richtung. Diese Bewegung wiederholt sich theoretisch endlos. Ziehe dazu aus der gelben Blockpalette ***Steuerung*** den Block ***wiederhole fortlaufend*** in das Skriptfenster.

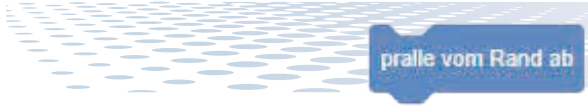


WIE ENTSTEHEN EIGENTLICH ZUFALLSZAHLEN?

Man denkt, in einem Programm könne nichts zufällig passieren, alles sei geplant. Wie kann ein Programm dann in der Lage sein, zufällige Zahlen zu generieren? Teilt man eine große Primzahl durch irgendeinen Wert, ergeben sich ab der x-ten Nachkommastelle Zahlen, die kaum noch vorhersehbar sind. Sie ändern sich auch ohne jede Regelmäßigkeit, wenn man den Divisor regelmäßig erhöht. Dieses Ergebnis ist zwar scheinbar zufällig, lässt sich aber durch ein identisches Programm oder den mehrfachen Aufruf des gleichen Programms jederzeit reproduzieren. Nimmt man aber eine aus einigen dieser Ziffern zusammengebaute Zahl und teilt sie wiederum durch eine Zahl, die sich aus der aktuellen Uhrzeitsekunde oder dem Inhalt einer beliebigen Speicherstelle des Rechners ergibt, kommt ein Ergebnis heraus, das sich nicht reproduzieren lässt und daher als Zufallszahl bezeichnet wird.

12 Alle Blöcke, die innerhalb dieser Klammer eingefügt werden, werden fortlaufend so lange wiederholt, bis du auf das rote Stoppsymbol klickst oder im Programm irgendetwas passiert, das dafür sorgt, dass die Wiederholung abgebrochen wird.

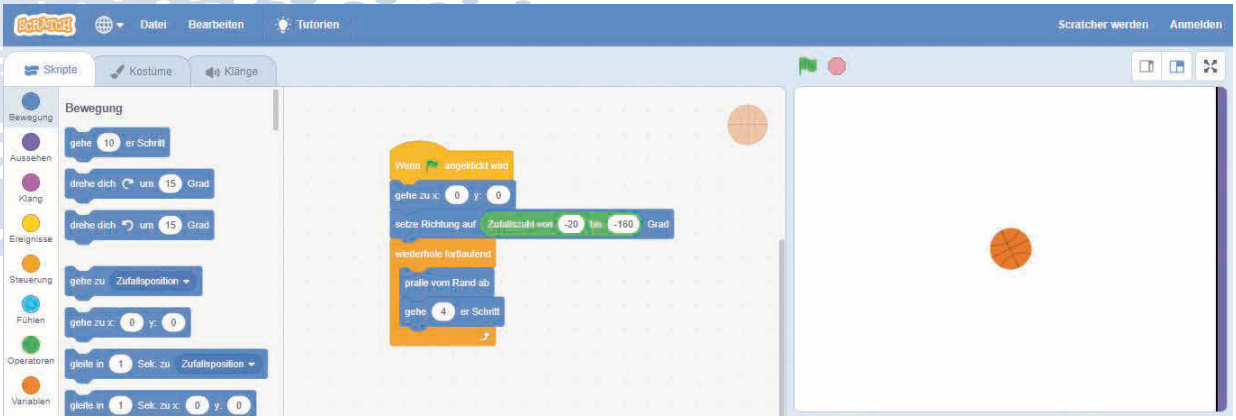
13 Ziehe als Erstes aus der blauen Palette ***Bewegung*** den Block ***pralle vom Rand ab*** in die Programmschleife ***wiederhole fortlaufend*** hinein. Dieser Block enthält mehrere Abfragen und Bewegungsvorgaben, die sonst mühsam programmiert werden müssten. Immer wenn die Figur (hier der Ball) an den Rand der Bühne stößt, ändert er automatisch seine Bewegungsrichtung, sodass es wie das Abprallen von einer Wand aussieht. Berührt die Figur in diesem Moment den Rand nicht, passiert gar nichts.



14 Füge dahinter einen Block ***gehe ...er Schritt*** ein und ändere den Wert im Zahlenfeld auf **4**, da sich der Ball einen vier Koordinateneinheiten großen Schritt bewegen soll.

2 Das erste eigene Spiel

15 So sieht das Programm bis jetzt aus:



16 Probiere es aus und starte das Programm mit dem grünen Fähnchen.

17 Der Ball fliegt gleichmäßig durch den Raum und prallt an allen vier Rändern ab. Die lilafarbene Kante rechts unterscheidet sich nicht von den anderen Rändern. Die Zahlen bei ***x:*** und ***y:*** in der Figurenpalette zeigen die aktuelle Position des Balls an.

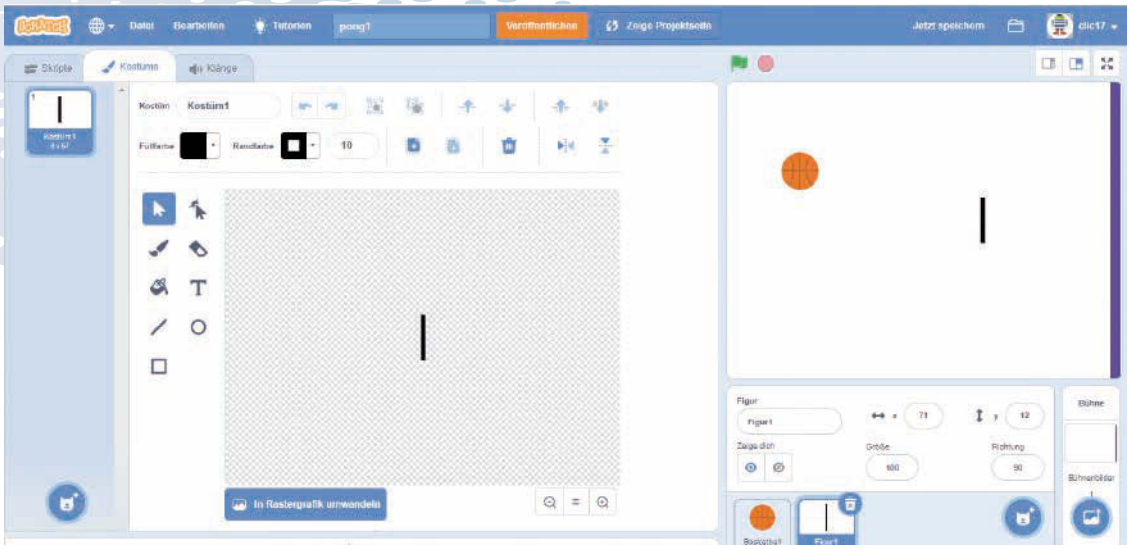
18 Stoppe den Ball mit dem roten Stoppsymbol, um weiter an dem Programm zu arbeiten.

1 Fahre dazu in der Figurenpalette rechts unten auf das Symbol für eine neue Figur und klicke dann in der neu eingeblendeten Symbolleiste auf das Symbol ***Malen***. Den Schläger werden wir nicht als fertiges Objekt importieren, sondern schnell selbst zeichnen.

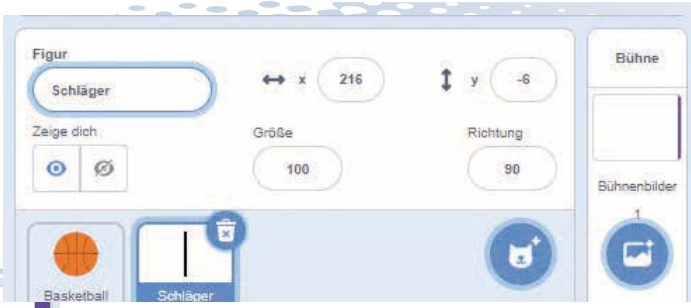
DER SCHLÄGER

Als Nächstes zeichnen wir den Schläger, mit dem du den Ball zurückwirfst, damit er nicht gegen die lilafarbene Kante rechts stößt.

2 Dazu wird das gleiche Malprogramm wie für den Hintergrund verwendet. Wähle das Werkzeug ***Linie***, die Farbe ***Schwarz*** und die Linienbreite ***10*** aus. Zeichne dann wie in der Abbildung eine kurze senkrechte Linie. Halte beim Zeichnen die Umschalt-Taste gedrückt. Dann wird die Linie automatisch exakt senkrecht gezeichnet und nicht leicht schief.



3 Klicke in der Mitte oben auf die Registerkarte ***Skripte***, um das Malprogramm zu verlassen. Der Schläger liegt bis jetzt noch irgendwo auf der Bühne. Klicke darauf und ziehe ihn kurz vor die lilafarbene Kante in die Mitte zwischen oberem und unterem Rand der Bühne.



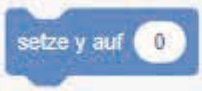
4 In der Figurenpalette taucht der Schläger als ***Figur1*** auf. Um ihm einen sinnvollen Namen zu geben, trage in das Feld ***Figur*** auf der Figurenpalette den neuen Namen ***Schläger*** ein.

5 Der Schläger wird über die Pfeiltasten auf der Tastatur gesteuert und braucht dazu ein Programm. Wähle den ***Schläger*** in der Figurenpalette aus, bis jetzt ist das Skriptfenster noch leer.

6 Baue hier ein kleines Programm, das automatisch startet, wenn du auf das grüne Fähnchen klickst.

2 Das erste eigene Spiel

7 Dieses Programm soll als Erstes die y-Koordinate des Schlägers auf ***0*** setzen, sodass der Schläger am Anfang genau in der Mitte zwischen oberem und unterem Rand der Bühne steht. Füge dazu einen Block ***setze y auf 0*** hinter dem Block ***Wenn Fahne angeklickt wird*** ein.

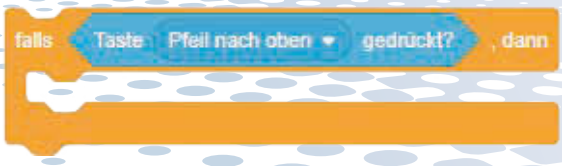


8 Anschließend beginnt eine Endlosschleife, die die beiden verwendeten Tasten der Tastatur abfragt. Ziehe dazu erneut einen Block ***wiederhole fortlaufend*** in das Programm.

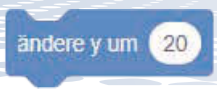
9 Baue jetzt eine ***falls***-Abfrage, die prüft, ob die Pfeiltaste nach oben gedrückt wurde. Solche ***falls***-Abfragen führen die Programmblöcke innerhalb der Klammer immer dann aus, wenn die angegebene Bedingung wahr ist.

10 Für die Abfrage selbst ist im ***falls ..., dann***-Block ein längliches Feld mit spitzen Enden vorgesehen. Hier muss ein Block eingefügt werden, der ebenfalls spitze Enden hat. Ziehe aus der blauen Blockpalette ***Fühlen*** den Block ***Taste ... gedrückt*** in das Feld im Block ***falls ..., dann***.

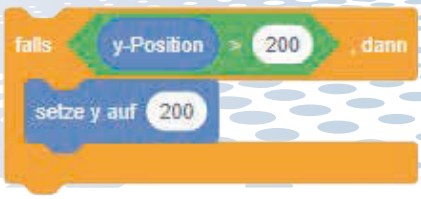
11 Klicke auf das kleine Dreieck neben dem Wort ***Leertaste*** in diesem Block. Wähle dann aus der Liste die Taste ***Pfeil nach oben*** aus. Hier werden alle Tasten zur Auswahl angeboten, die Scratch unterstützt. Das sind die Pfeile, die Leertaste sowie alle Buchstaben und Ziffern.



12 Jedes Mal, wenn du die Pfeiltaste nach oben drückst, soll sich der Schläger um 20 Koordinateneinheiten nach oben bewegen. Dazu gibt es auf der blauen Blockpalette ***Bewegung*** einen Block ***ändere y um ...*** Trage hier den Wert ***20*** ein und ziehe ihn in die Klammer des ***falls ..., dann***-Blocks.



13 Falls der Schläger am oberen Rand angekommen ist, soll er sich nicht weiterbewegen. Hier verwenden wir eine ***falls***-Abfrage, die prüft, ob die y-Position größer als 200 ist. Ist das der Fall, wird die y-Position einfach auf 200 gesetzt.



14 Diesmal bildet ein grüner Operator die Abfrage im ***falls ..., dann***-Block. Ziehe aus der Blockpalette der Operatoren den Block ***... > ...*** in das Feld für die ***falls***-Bedingung.

15 Ziehe in das linke Feld des ***größer***-Operators den Block ***y-Position*** aus der blauen Blockpalette ***Bewegung*** und trage in das rechte Feld die Zahl ***200*** ein.

16 Wenn die Bedingung, dass die y-Position größer als 200 ist, zutrifft, soll der Schläger auf die y-Position 200 gesetzt werden. Ziehe dazu einen Block ***setze y auf ...*** in die Klammer des ***falls ..., dann***-Blocks. Trage hier den Wert ***200*** ein.

17 Die gesamte Abfrage der Pfeiltaste nach oben sieht jetzt so aus:

18 Nun brauchen wir noch eine ähnliche Abfrage, die die Pfeiltaste nach unten abfragt und, wenn diese gedrückt wurde, den Schläger um 20 Einheiten nach unten bewegt. Auch in diesem Fall muss überprüft werden, ob der Schläger unter –200 steht. Natürlich schreibt ein echter Programmierer keinen Programmteil doppelt. Auch in Scratch brauchst du das nicht zu tun, sondern kannst vorhandene Programmteile duplizieren. Klicke mit der rechten Maustaste auf den ***falls ..., dann***-Block und wähle im Kontextmenü ***Duplizieren***.

19 Jetzt hast du eine Kopie des ganzen Blocks mit allen Blöcken, die darin sind oder darunterhängen. Ziehe diese auf eine freie Fläche im Skriptfenster. Dann kannst du alle Werte in den Blöcken bearbeiten.

20 Wähle in der Liste ***Taste ... gedrückt*** die Taste ***Pfeil nach unten***.

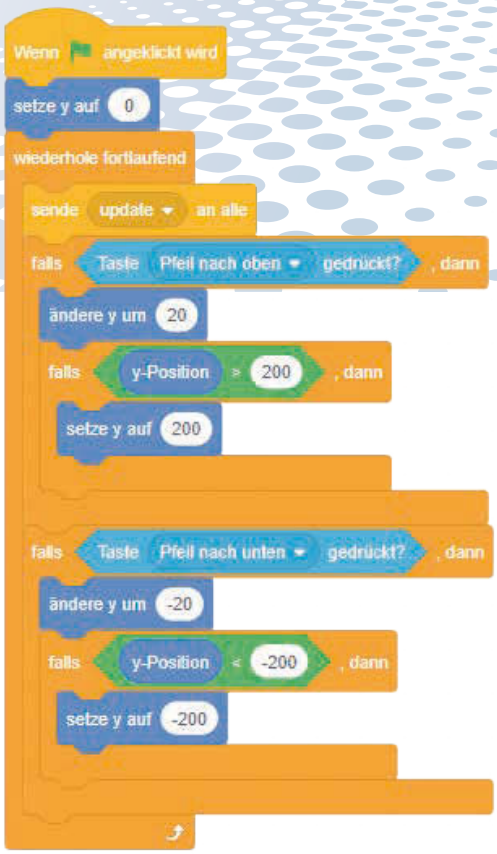
2 Das erste eigene Spiel

21 Trage in den Block ***ändere y um ...*** den Wert ***-20*** ein.

22 Ersetze den ***größer***-Operator durch einen ***kleiner***-Operator, indem du ihn aus dem ***falls ..., dann***-Block ziehst. Ziehe auch den Block ***y-Position*** heraus, füge einen neuen Block ***kleiner*** ein und ziehe den Block ***y-Position*** wieder in den Block.

23 Trage in den Block ***falls y-Position <, ... dann*** den Wert ***-200*** ein.

24 Trage in den Block ***setze y auf ...*** den Wert ***-200*** ein.



25 Baue diese beiden ***falls ..., dann***-Abfragen in die Hauptschleife des Programms ein.

26 Starte das Programm mit einem Klick auf das grüne Fähnchen. Du kannst mit den Pfeiltasten den Schläger steuern. Der Ball fliegt aber weiterhin völlig unbeirrt durch den Raum. Die Programmblöcke für den Ball und für den Schläger starten beide, nachdem du auf das grüne Fähnchen geklickt hast.

27 Stoppe den Ball mit dem roten Stoppsymbol, um weiter an dem Programm zu arbeiten.

DIE SPIELREGELN

Nachdem die Spielmechanik nun funktioniert, braucht das Spiel noch seine Regeln. In diesem einfachen Spiel gibt es nur zwei:

- Prallt der Ball gegen den Schläger, wird er zurückgeschlagen, wobei es eine leichte zufällige Veränderung der Flugbahn geben soll, damit der Ball nicht immer völlig gleichförmig durch den Raum fliegt. Sonst könnte man als Spieler den Schläger einfach an einer bestimmten Stelle stehen lassen.

- Prallt der Ball gegen den lilafarbenen Balken am rechten Rand (weil der Spieler ihn mit dem Schläger verfehlt hat), gibt es einen Minuspunkt, und der Ball startet in der Mitte des Spielfelds neu.

Diese beiden Regeln werden in zwei zusätzlichen Programmblöcken definiert, die beide automatisch mit dem Spiel starten.

0 Wähle auf der Figurenpalette unten rechts den Ball aus. Die neuen Programmblöcke betreffen diese Figur. Im Programmfenster ist

dann wieder das Programm zu sehen, das die Ballbewegung steuert.

2 Baue hier eine weitere Gruppe von Programmblöcken, die ebenfalls starten soll, wenn das grüne Fähnchen angeklickt wird.

3 Danach folgt eine Endlosschleife, die immer wieder prüfen soll, ob die aktuelle Figur, der Ball, den Schläger berührt. Baue dazu erneut einen ***falls ..., dann***-Block in einen ***wiederhole fortlaufend***-Block ein.

4 Um die Berührung zu überprüfen, gibt es auf der Blockpalette ***Fühlen*** den Block ***wird ... berührt***. Wähle hier im Listenfeld das Objekt ***Schläger*** aus.

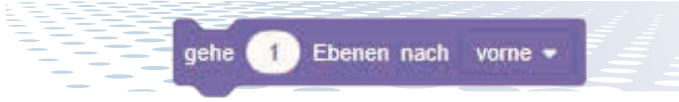
5 Ziehe diesen Block in das Abfragefeld mit den spitzen Enden im ***falls ..., dann***-Block.

6 Falls der Ball den Schläger berührt, wird die Bewegungsrichtung ins Negative umgekehrt. Der Ball fliegt im gleichen Winkel nach links unten weiter, in dem er von links oben kam, oder umgekehrt – wenn er von links unten kam, fliegt er nach links oben weiter.

7 Baue dazu in einen blauen ***setze Richtung auf ...***-Block einen grünen Operator **–** ein.

8 Trage in das erste Feld dieses Operators eine ***0*** ein und ziehe in das zweite Feld den Block ***Richtung*** aus der Blockpalette ***Bewegung***. Dieser Block enthält die aktuelle Richtung als Zahlenwert, mit dem ganz normal gerechnet werden kann.

9 Um die Bewegung etwas unvorhersehbarer zu gestalten, wird der Ball zunächst einen kleinen 5er-Schritt bewegt, damit danach der Schläger auf keinen Fall mehr berührt wird.



10 Anschließend wird die Flugrichtung gegenüber der bisherigen Richtung um einen zufälligen Wert zwischen ***-20*** und ***20*** Grad verändert. Ziehe dazu in das Zahlenfeld eines ***drehe dich***

2 Das erste eigene Spiel

um ... Grad-Blocks einen grünen Block ***Zufallszahl von ... bis ...***

11 Zusammen ergibt das diese Gruppe von Programmblöcken, die von Anfang an mit gestartet wird und darauf wartet, dass der Ball den Schläger berührt. Nur dann werden die blauen Blöcke ausgeführt.

12 Starte jetzt wieder das Programm mit einem Klick auf das grüne Fähnchen.

13 Du kannst mit den Pfeiltasten den Schläger steuern, und der Ball prallt davon ab, wenn du ihn triffst. Berührt der Ball allerdings die lilafarbene Kante am rechten Bildschirmrand, passiert noch nichts Besonderes.

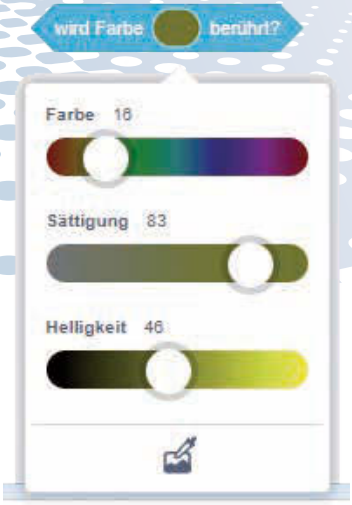
14 Stoppe den Ball mit dem roten Stoppsymbol, um weiter an dem Programm zu arbeiten.

PUNKTE ZÄHLEN

Um die Punkte zu zählen, verwenden wir im Programm eine sogenannte Variable. Variablen sind kleine Speicherplätze, in denen man sich während eines Programms eine Zahl oder irgendetwas anderes merken kann.

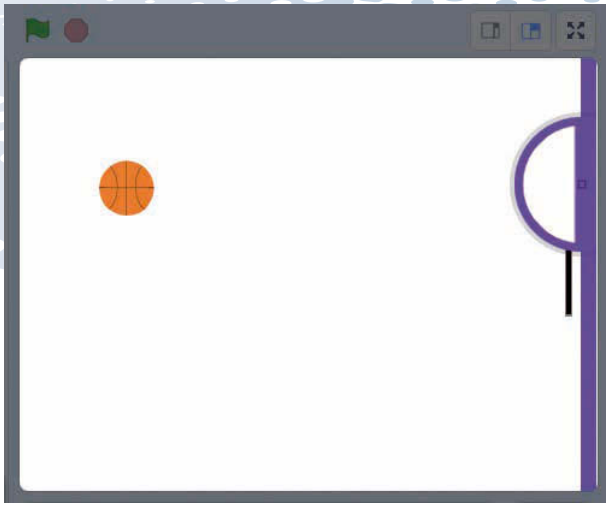
1 Um die zweite Spielregel umzusetzen, nämlich was passiert, wenn der Ball den lilafarbenen Balken berührt, baue eine weitere Gruppe von Programmblöcken, die ebenfalls starten soll, wenn das grüne Fähnchen angeklickt wird.

2 Auch hier findet sich wieder ein ***wiederhole fortlaufend***-Block, in dem ein ***falls ..., dann***-Block steckt, der diesmal nur durchlaufen soll, wenn der Ball den lilafarbenen Balken berührt. Dafür gibt es auf der Blockpalette ***Fühlen*** den Block ***wird Farbe ... berührt?***.



3 Tippe in diesem Block auf das Farbfeld, und es erscheint ein Farbauswahlfeld, in dem du eine beliebige Farbe einstellen kannst.

4 Klickst du hier unten auf das Palettensymbol, wird die Bühne erleuchtet. Darin erscheint eine kreisförmige Lupe, mit der du die Farbe Lila auf der Bühne auswählen musst. Diese wird automatisch in das Farbfeld übernommen.



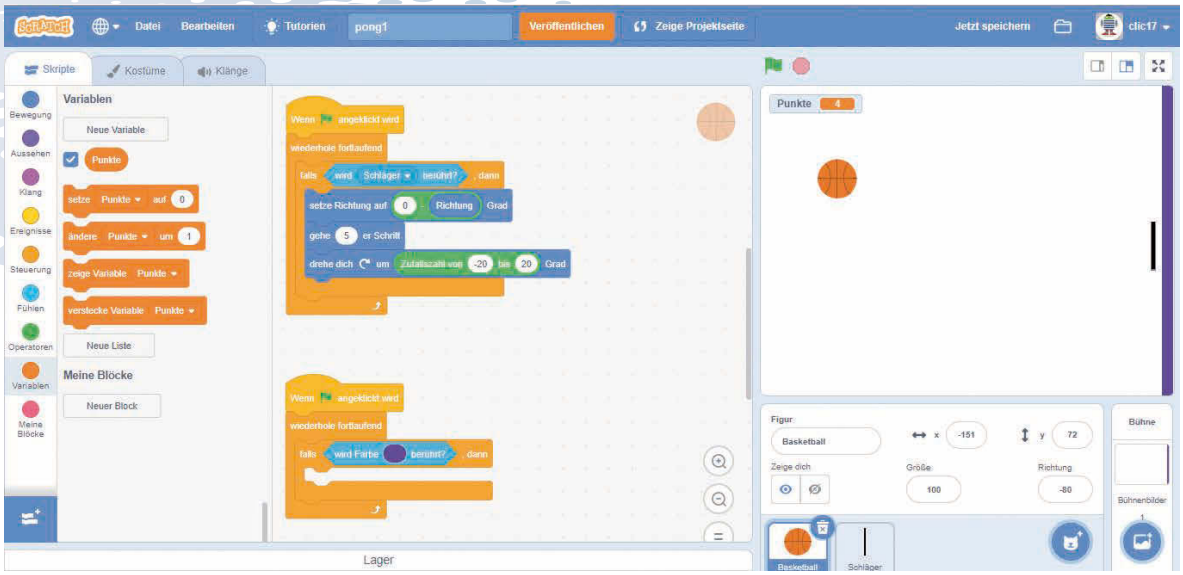
5 Ziehe den Block in das Abfragefeld des ***falls ..., dann***-Blocks.

6 Berührt der Ball den lilafarbenen Balken, bekommt der Spieler einen Minuspunkt. Um den Punktestand zu speichern, brauchen wir eine Variable. Variablen müssen in Scratch erst einmal angelegt werden, bevor man sie benutzen kann. Klicke in der Blockpalette oben auf das orangefarbene Symbol ***Variablen*** und dann auf ***Neue Variable***.

7 Gib der Variablen den Namen ***Punkte***.

8 In der Blockpalette werden jetzt verschiedene Blöcke zur Arbeit mit Variablen angezeigt. Aktiviere den Schalter links neben der Variablen ***Punkte***, und diese Variable wird automatisch auf der Bühne in einem kleinen orangefarbenen Feld angezeigt. So siehst du immer den aktuellen Punktestand.

2 Das erste eigene Spiel



9 Berührt der Ball den lilafarbenen Balken, wird die Variable ***Punkte*** um 1 erhöht. Ziehe dazu den Block ***ändere Punkte um ...*** in die Klammer des ***falls wird Farbe ... berührt?, dann***-Blocks. Trage im Zahlenfeld dieses Blocks ***1*** ein.

10 Anschließend wird der Ball wieder in die Spielfeldmitte auf die Position ***x: 0, y: 0*** gesetzt. Ziehe dazu einen Block ***gehe zu x: ... y: ...*** in das Programm und trage in beide Zahlenfelder ***0*** ein.

11 Auch in diesem Fall wird der Ball um einen zufälligen Wert zwischen ***-20*** und ***20*** Grad gedreht, damit er nicht wieder exakt die gleiche Flugbahn nimmt, aber trotzdem ungefähr in die Richtung fliegt, in die er zuletzt geflogen ist.

12 So sieht die Blockgruppe zum Zählen der Minuspunkte aus:

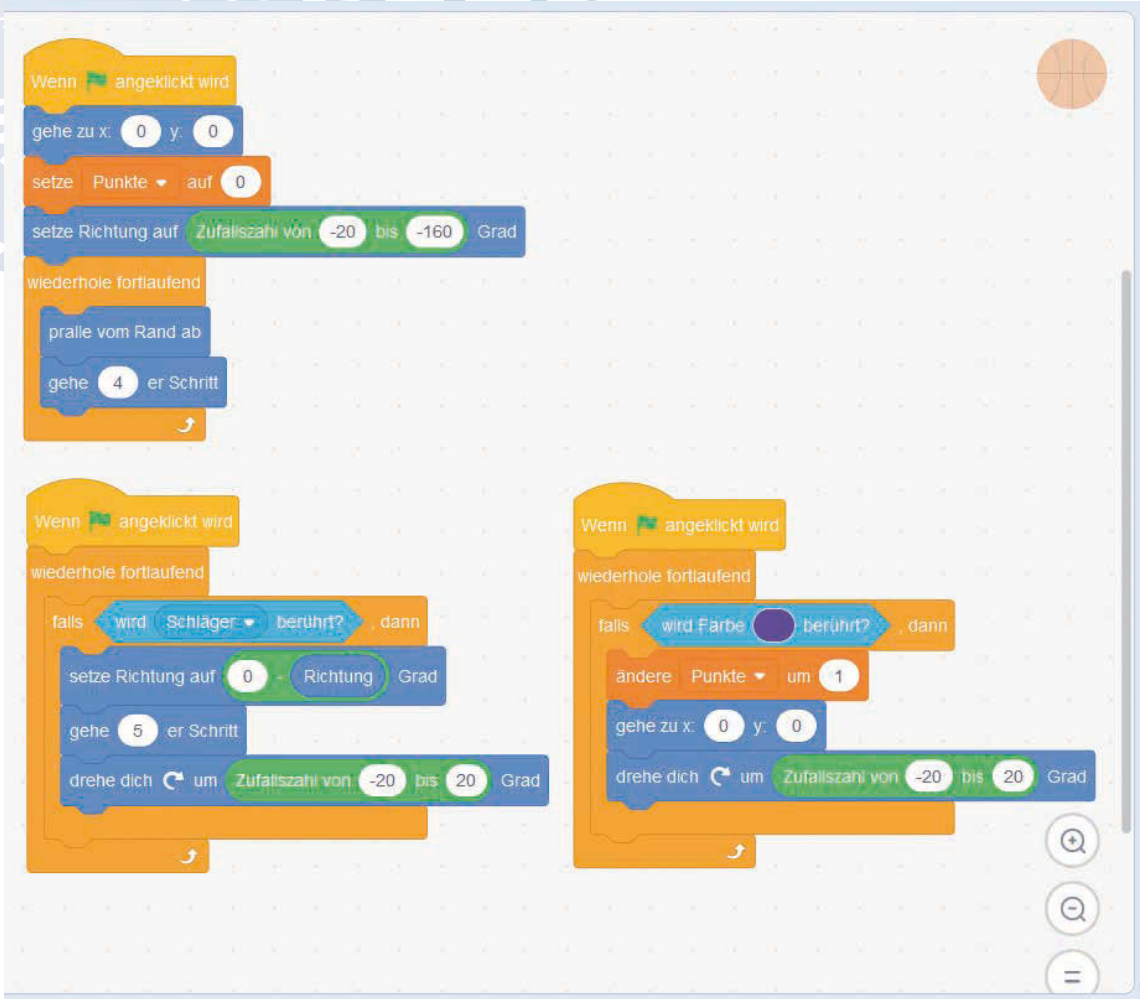
13 Wenn du das Programm jetzt mit dem grünen Fähnchen startest, scheint auf den ersten Blick alles zu funktionieren.

14 Hältst du das Programm mit dem roten Stoppsymbol an und startest es neu, werden die Punkte aber einfach weitergezählt, es wird nicht wieder bei 0 angefangen.

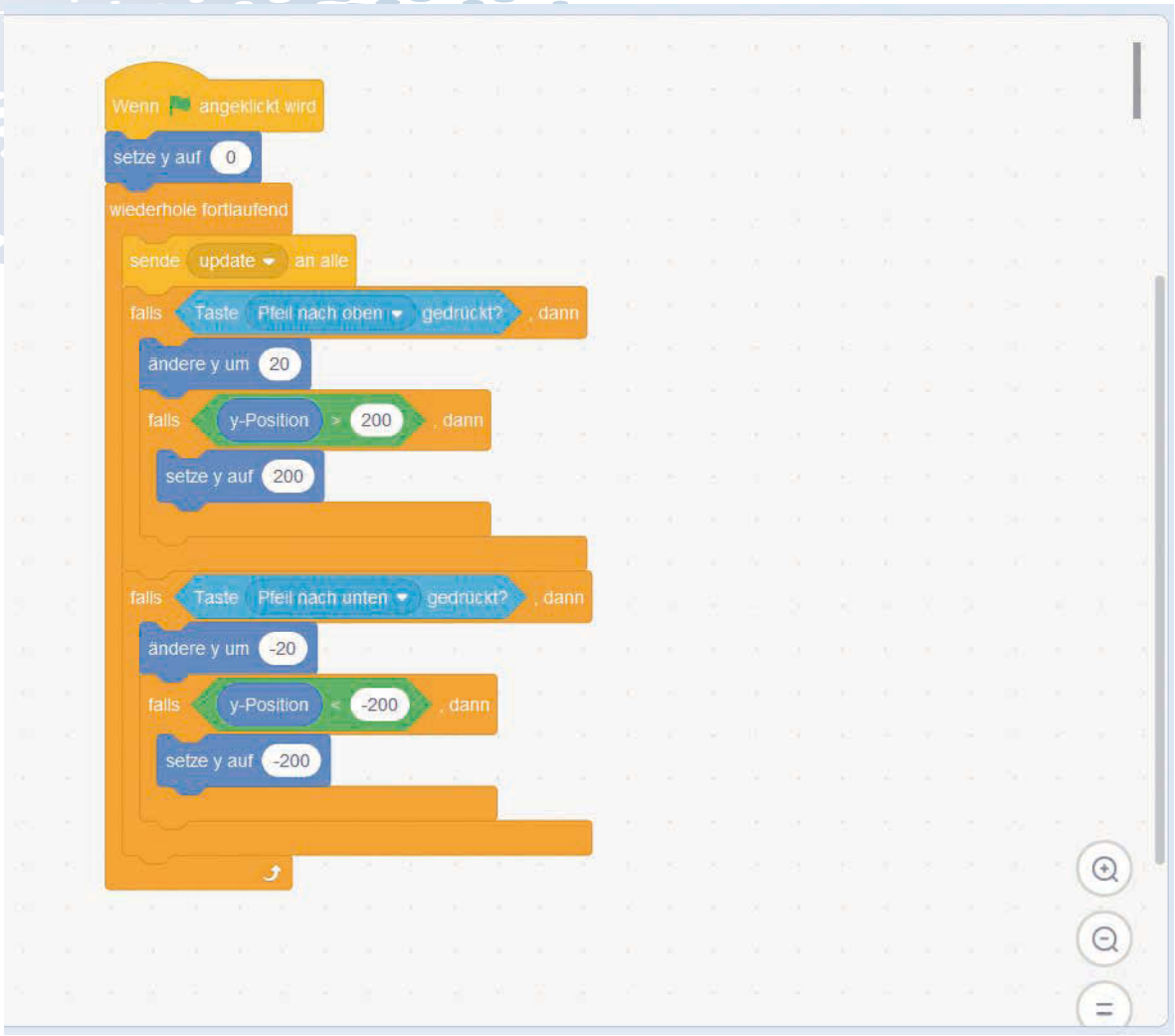
15 Ziehe deshalb einen Block ***setze Punkte auf ...*** aus der Blockpalette ***Daten*** ganz an den Anfang des Programms und trage in das Zahlenfeld ***0*** ein. Dieser Block setzt die Variable ***Punkte*** auf ***0***, egal auf welchem Wert sie vorher stand.

16 Damit ist das Programm fertig.

17 Wenn du auf das grüne Fähnchen klickst, starten nun drei Gruppen von Programmblöcken für den Ball und eine für den Schläger.



2 Das erste eigene Spiel



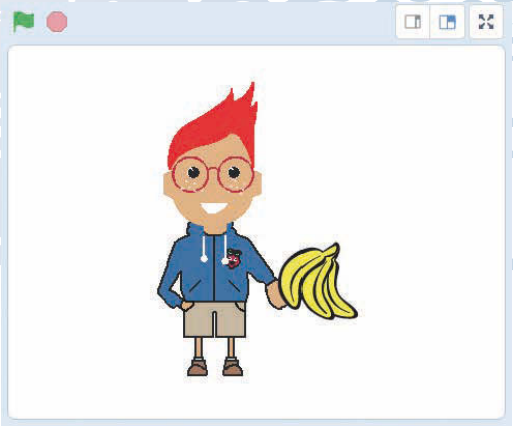
Klicke auf das grüne Fähnchen und versuche, mit dem Schläger den Ball immer wieder abzuwehren, um möglichst wenige Minuspunkte zu bekommen.


GB
micro
SD

3 Der kleine Hacker und die Bananen

Scratch eignet sich ideal zum Erzählen kleiner Geschichten, wie man es früher in kurzen Zeichentrickfilmchen machte. Dieses ganz einfache Projekt zeigt, wie man eigene Figuren einbaut und miteinander spielen lässt.

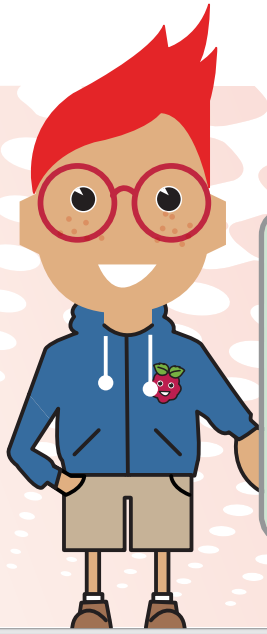
Der kleine Hacker mag Bananen – wenn er welche bekommt, wird sein Gesicht sofort fröhlich.



DER KLEINE HACKER IN SCRATCH

Als Erstes wird der kleine Hacker in Scratch importiert. Du brauchst ihn nicht selbst zu zeichnen.

1 Starte ein neues Projekt in Scratch. In den Downloads zum Buch findest du den kleinen Hacker als Grafikdatei. Fahre mit der Maus in der Figurenpalette auf das Symbol ***Figur wählen*** und klicke dann auf ***Figur hochladen***. Wähle aus dem Ordner, in den du die Downloads entpackt hast, das Bild ***Kleiner_Hacker1.png***.

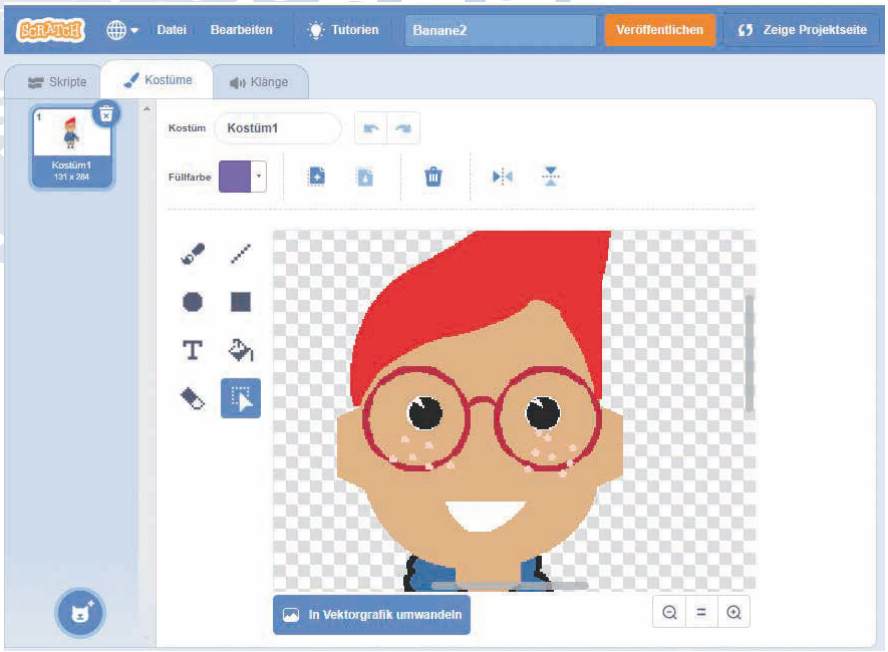


DOWNLOADS ZUM BUCH

Zu diesem Buch kannst du dir einige nützliche Dateien herunterladen. Besuche dazu die Seite *www.buch.cd* und gib dort diesen Code ein. Folge den weiteren Anweisungen auf der Seite, und du kannst dir eine Zip-Datei herunterladen, die du anschließend in einen persönlichen Ordner auf deiner Festplatte entpacken musst.

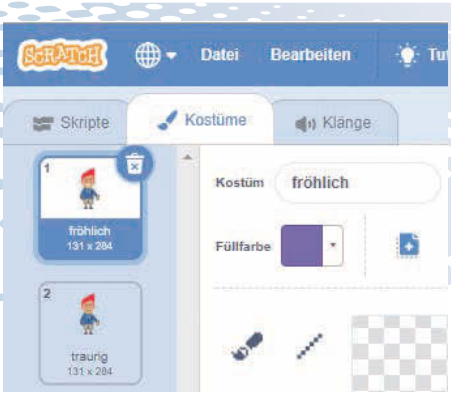
2 Lösche die Scratch-Katze, indem du sie in der Figurenliste auswählst und auf das blaue Papierkorbsymbol klickst.

3 Schalte auf die Seite ***Kostüme*** um. Nun ist der kleine Hacker im Grafikmodul von Scratch zu sehen. Bis jetzt zeigt er sein freundliches Gesicht.



4 Wenn er keine Bananen bekommt, soll der kleine Hacker ein trauriges Gesicht machen. Eine Scratch-Figur kann verschiedene sogenannte Kostüme haben. Wenn sie im Programm ihr Aussehen verändern soll, wechselt sie einfach das Kostüm. Die einfachste Methode, ein neues Kostüm zu erstellen, ist, das vorhandene zu duplizieren. Klicke dazu mit der rechten Maustaste auf das Kostüm links oben und wähle ***Duplizieren***. Unter dem kleinen Hacker erscheint ein zweiter, der genau gleich aussieht.

5 Jedes Kostüm bekommt automatisch einen Namen. Gib den beiden Kostümen leicht zu merkende Namen. Klicke dazu auf das obere Kostüm und ändere den Namen im Namensfeld auf ***fröhlich***, da der kleine Hacker fröhlich sein soll, wenn er die Bananen bekommt.

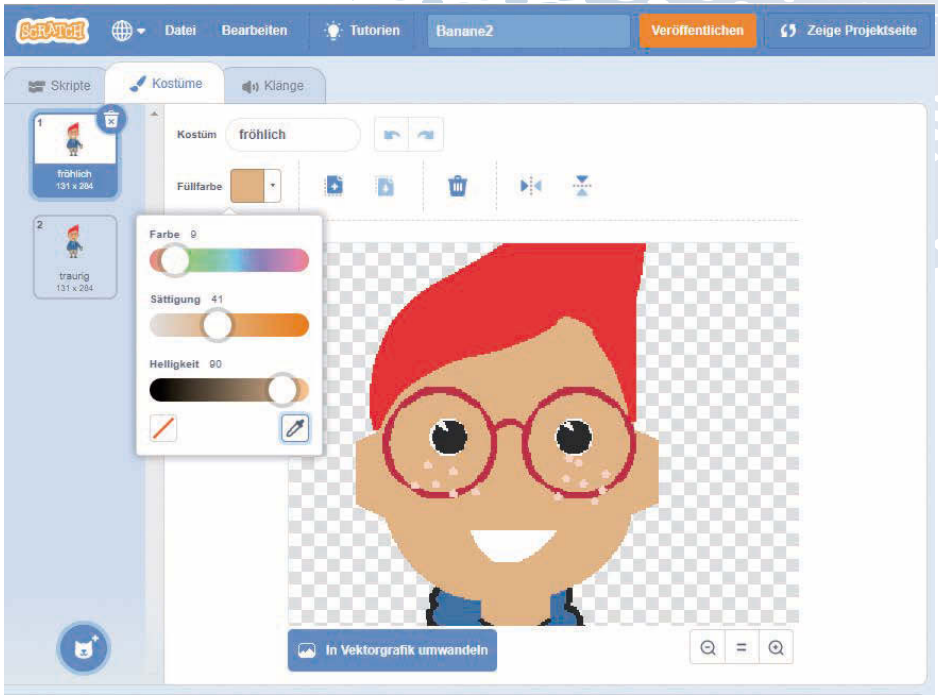


6 Ändere auf die gleiche Weise den Namen des zweiten Kostüms auf ***traurig***. Diese kurzen

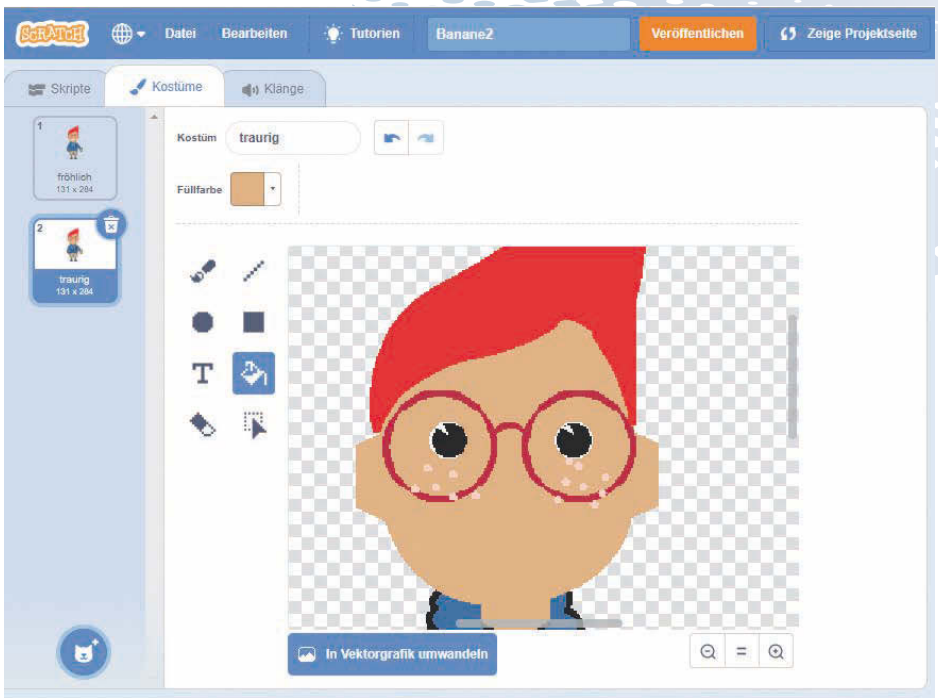
3 Der kleine Hacker und die Bananen

Namen sind in der Liste der Kostüme klar zu erkennen.

7 Das Kostüm ***fröhlich*** kann bleiben, wie es ist. Das traurige Kostüm soll einen traurigen Gesichtsausdruck bekommen. Am einfachsten erreicht man das durch einen anderen Mund. Wähle das Kostüm ***traurig*** und klicke auf das Symbol ***Füllfarbe***. Klicke dann unten im Farbeinstellfeld auf die Farbpipette. Wähle mit dieser Farbpipette die Gesichtsfarbe des kleinen Hackers. Diese erscheint dann als Farbe im Farbeinstellfeld. Um besser malen zu können, kannst du mit den Lupensymbolen unten rechts im Bild zoomen.

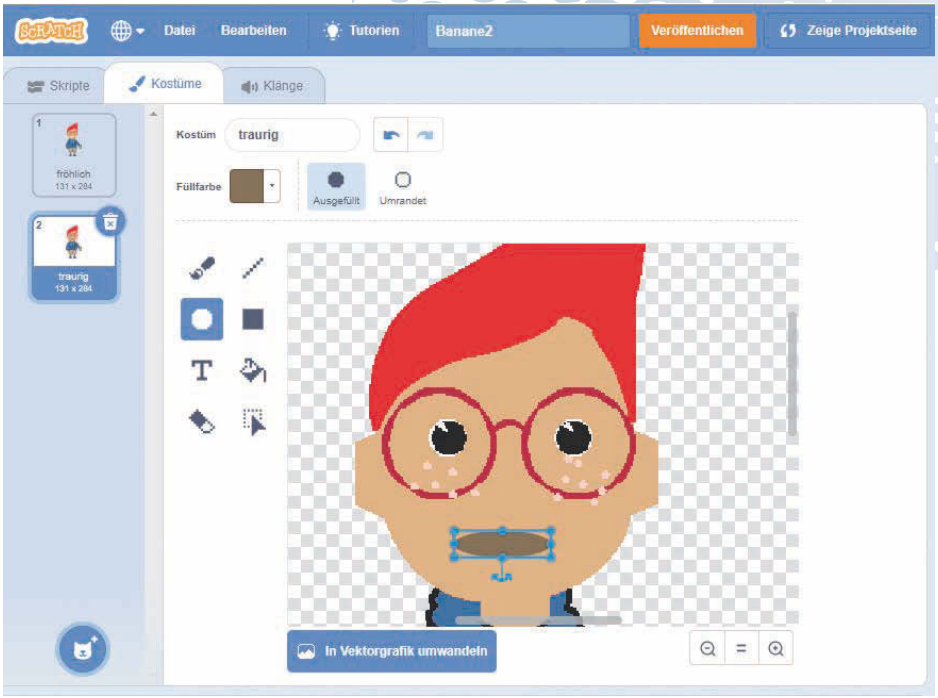


8 Fülle den weißen Mund mit der Gesichtsfarbe, um ihn verschwinden zu lassen. Klicke dazu auf das Symbol mit dem Farbeimer ***Fülleimer*** und dann auf den Mund.

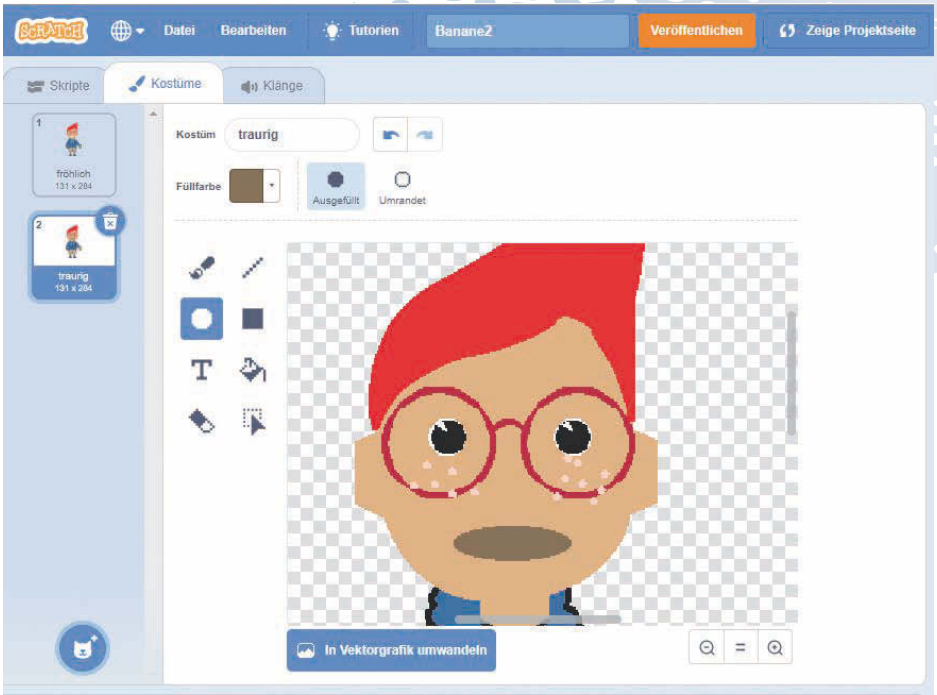


9 Male nun einen schlecht gelaunten Mund auf das Gesicht. Wähle dazu

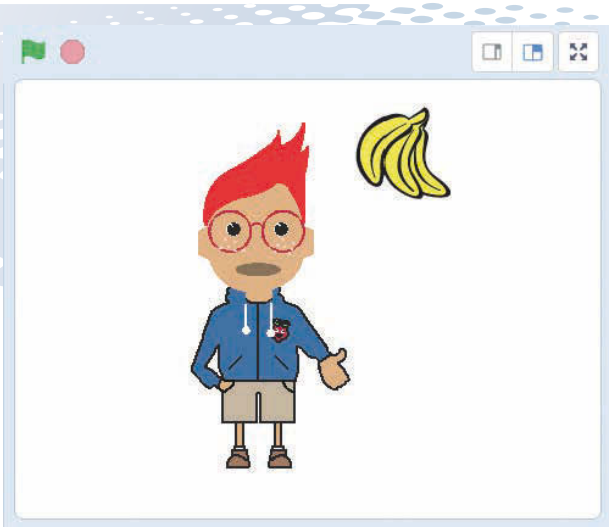
eine dunkelbraune Farbe und das Kreissymbol. Zeichne als Mund eine Ellipse. Wenn dir die Form oder die Lage der Ellipse beim ersten Versuch nicht gefällt, kannst du sie mit den kleinen quadratischen Griffen noch verändern und verschieben. Klicke danach irgendwo außerhalb der Ellipse ins Bild, um sie fest in die Zeichnung einzufügen.



Das neue Kostüm wird automatisch gespeichert. Wenn du in der Liste der Kostüme eines auswählst, nimmt der kleine Hacker auf der Scratch-Bühne es sofort an. Ziehe die Figur auch gleich so auf die Bühne, dass sie ganz zu sehen ist, falls sie etwas über den Rand gerutscht ist. Die genaue Lage beim Einfügen hängt von der Bildschirmauflösung ab, kann also auf jedem Computer etwas anders sein.

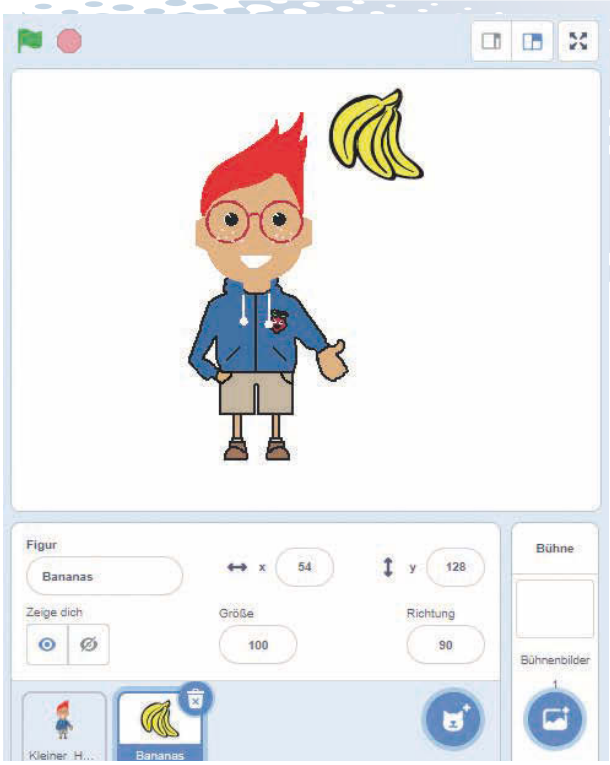


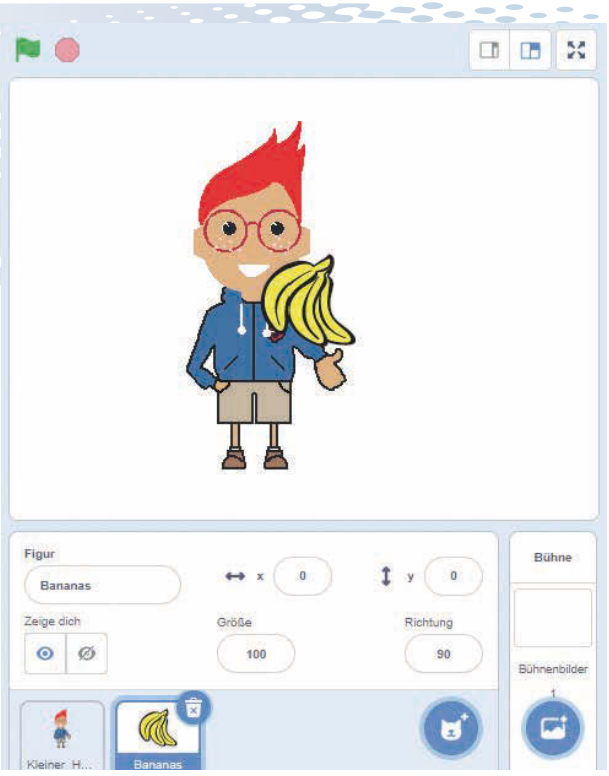
3 Der kleine Hacker und die Bananen



DIE BANANEN

1 Jetzt kommen die Bananen ins Spiel. Dafür verwenden wir ein fertiges Objekt aus der umfangreichen Scratch-Bibliothek. Klicke auf das Symbol ***Figur wählen*** in der Figurenpalette, wähle in der Figurenbibliothek die Figur ***Bananas*** aus und klicke auf ***OK***.

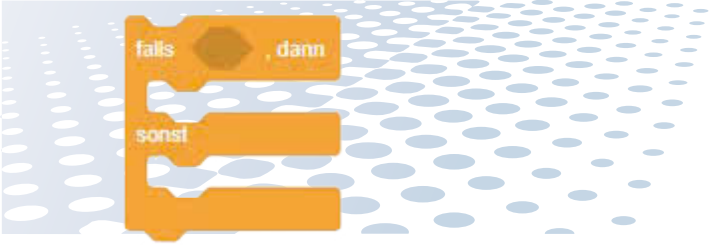




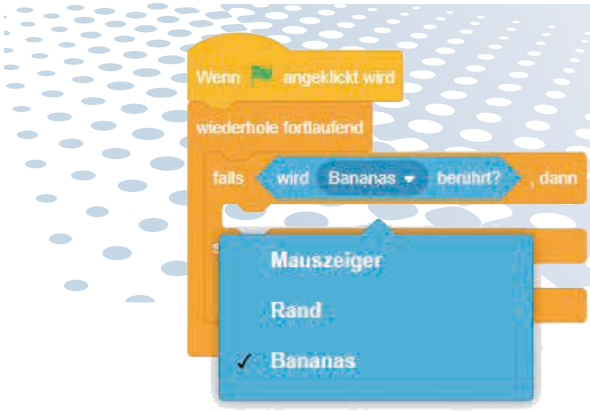
2 Schiebe die Bananen auf der Bühne rechts neben den kleinen Hacker, damit die beiden Figuren einander nicht mehr berühren. Immer wenn die Bananen den kleinen Hacker berühren, soll er sein fröhliches Gesicht zeigen. Das wird ein kurzes Scratch-Programm erledigen, das fortlaufend prüft, ob der kleine Hacker die Bananen berührt.

3 Wähle in der Figurenliste unten links den kleinen Hacker aus, für ihn soll das Programm gelten, da er sein Aussehen verändern soll. Das Programm soll automatisch starten, wenn du auf das grüne Fähnchen klickst, und dann endlos laufen.

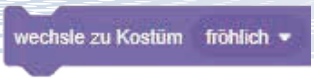
4 Innerhalb der Endlosschleife findet eine ***falls ... sonst***-Abfrage statt. Dies ist eine erweiterte Form der bereits bekannten ***falls***-Abfrage mit dem Unterschied, dass auch im Fall, dass die Abfrage kein wahres Ergebnis liefert, Programmblöcke ausgeführt werden und nicht einfach nichts passiert.



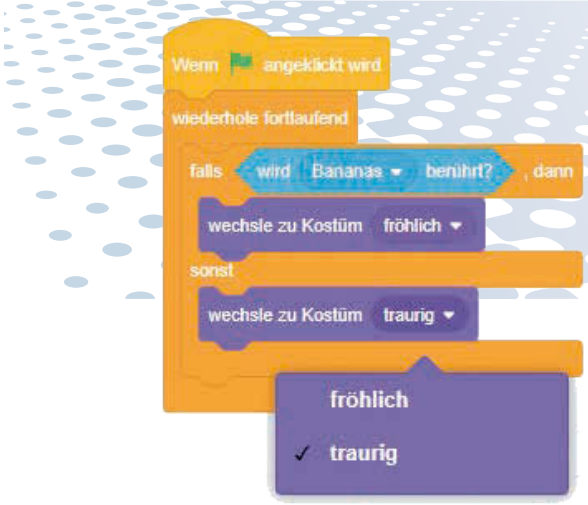
5 Die Frage lautet hier, ob der kleine Hacker die Bananen berührt. Wenn ja, soll er das fröhliche Kostüm annehmen. Ziehe dazu einen Block ***wird ... berührt*** aus der Blockpalette ***Fühlen*** in den ***falls ..., dann***-Block und wähle im Listenfeld die Figur ***Bananas*** aus. Die Figur ***Kleiner_Hacker*** wird gar nicht zur Auswahl angeboten, da sich eine Figur nicht selbst berühren kann.



6 Berührt der kleine Hacker die Bananen, wird er fröhlich. Ziehe dazu den Block ***Wechsle zu Kostüm*** von der Palette ***Aussehen*** in den ersten Blockbereich der ***falls ... sonst***-Abfrage. Wähle im Listenfeld dieses Blocks das Kostüm ***fröhlich***.



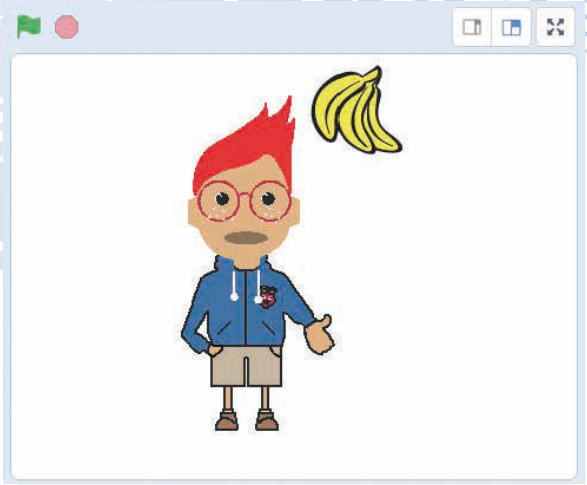
7 Ziehe einen weiteren Block ***Wechsle zu Kostüm*** in den zweiten Blockbereich der ***falls ... sonst***-Abfrage. Wähle diesmal im Listenfeld dieses Blocks das Kostüm ***traurig***, da der kleine Hacker sofort wieder traurig werden soll, wenn die Bananen weg sind.

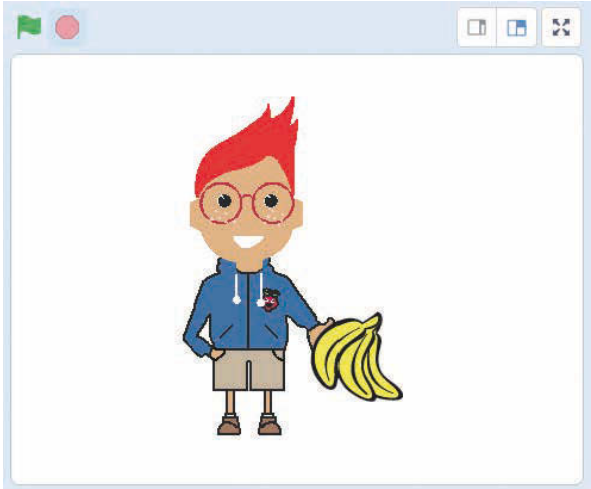


3 Der kleine Hacker und die Bananen

Jetzt kannst du erst einmal ausprobieren, ob bis hierhin alles funktioniert. Jeder Programmierer testet während der Entwicklung zwischendurch mal sein Programm. So lassen sich Fehler rechtzeitig entdecken und vermeiden.

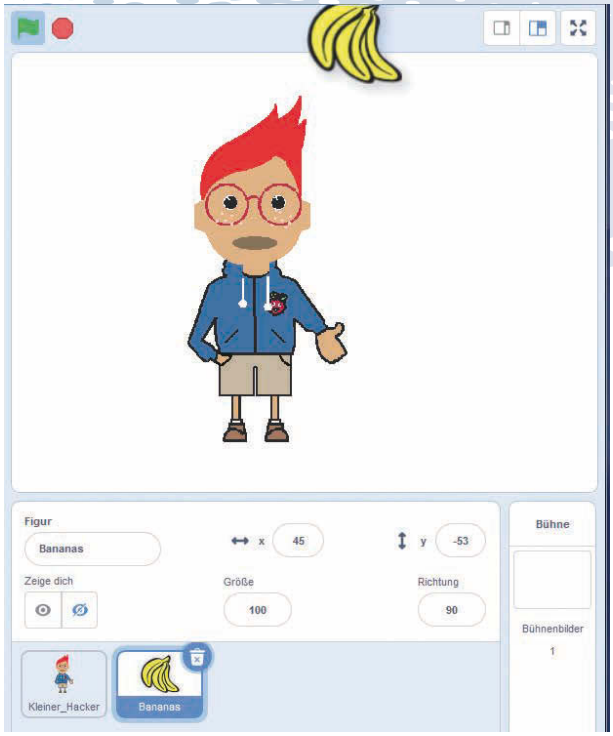
Klickst du auf das grüne Fähnchen, ist der kleine Hacker traurig. Nimm mit der Maus die Bananen und schiebe sie mit gedrückter Maustaste so, dass der kleine Hacker sie berührt. Lass dort die Maustaste los, und der kleine Hacker wird fröhlich.

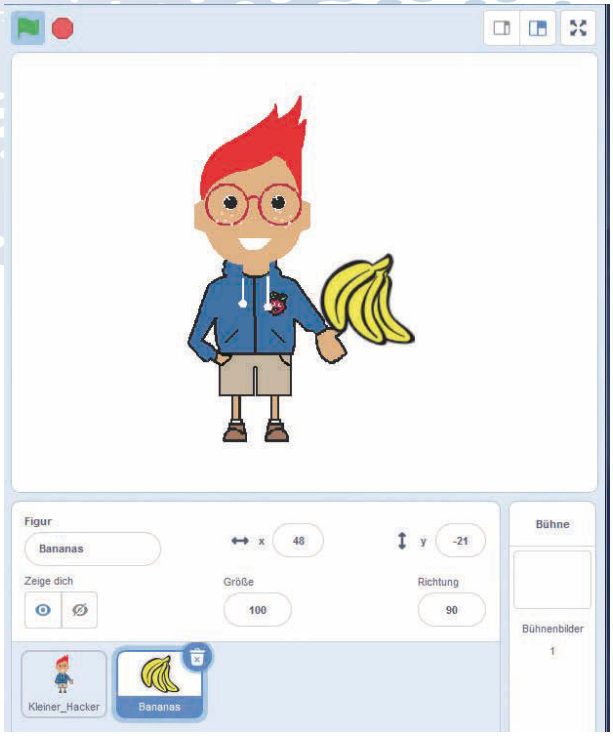




DIE BANANEN FALLEN HERUNTER

Jetzt sollen die Bananen automatisch von oben herunterfallen, und der kleine Hacker fängt sie auf. Ziehe die Bananen an eine Position rechts oberhalb des kleinen Hackers. Du kannst sie auch etwas über den Rand aus der Bühne hinausziehen. Wenn du sie dort loslässt, bleiben sie an der Stelle, der Rand der Bühne verdeckt sie aber teilweise. So können Objekte von außen ins Bild fliegen.





1 Bis jetzt ist das Skriptfenster noch leer, wenn du auf die Bananen klickst. Baue wieder eine Endlosschleife, die startet, wenn auf das grüne Fähnchen geklickt wird.

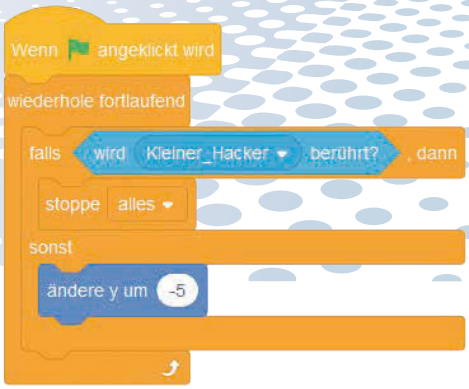
2 Auch innerhalb dieser Endlosschleife findet eine ***falls ... sonst***-Abfrage statt, die prüft, ob der kleine Hacker die Bananen bereits gefangen hat.

3 Die Figur ***Kleiner_Hacker*** führt bereits eine Aktion aus, wenn sie die Bananen berührt. Umgekehrt bekommen die Bananen jetzt auch noch Programmblöcke, falls sie den kleinen Hacker berühren.

4 Wenn die Bananen die Figur berühren, soll das Programm beendet werden. Eine Endlosschleife endet normalerweise nie, der Block ***stoppe alles*** macht es aber möglich, so ein Programm trotzdem zu beenden.

5 Berühren die Bananen den kleinen Hacker nicht, fallen sie um fünf Schritte nach unten. Danach startet die Endlosschleife die ***falls ... sonst***-Abfrage erneut.

3 Der kleine Hacker und die Bananen



Starte das Programm mit einem Klick auf das grüne Fähnchen.

Wie erwartet, fallen die Bananen nach unten bis auf die Hand des kleinen Hackers. Im gleichen Moment setzt er wieder sein fröhliches Lächeln auf, und das Programm wird beendet, was daran zu erkennen ist, dass das rote Stoppsymbol leuchtet und die grüne Fahne wieder ausgeschaltet wird.

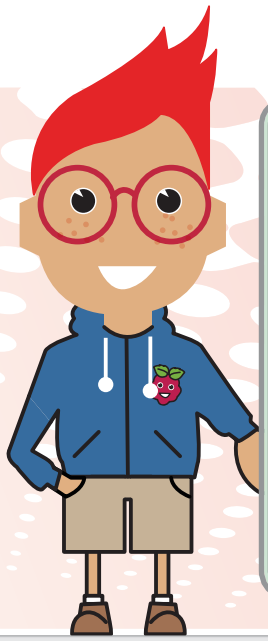
DIE BANANEN FALLEN NOCH ECHTER HERUNTER

Um das Programm wieder neu zu starten, musst du die Bananen in ihre Ausgangsposition zurückbringen. Das wird die nächste Programmversion automatisch erledigen.

Schiebe die Bananen noch einmal in die Ausgangsposition. Auf der Figurenpalette kannst du die aktuellen Koordinaten ablesen.

Ziehe einen blauen Block ***gehe zu x: ... y: ...*** direkt hinter den Startblock im Programm. Die Koordinaten werden in diesem Block automatisch eingetragen.

Wenn du jetzt auf das grüne Fähnchen klickst, fallen die Bananen, wie beim letzten Mal, wieder herunter. Erst beim nächsten Start des Programms passiert etwas Neues. Die Bananen starten automatisch in der Ausgangsposition.

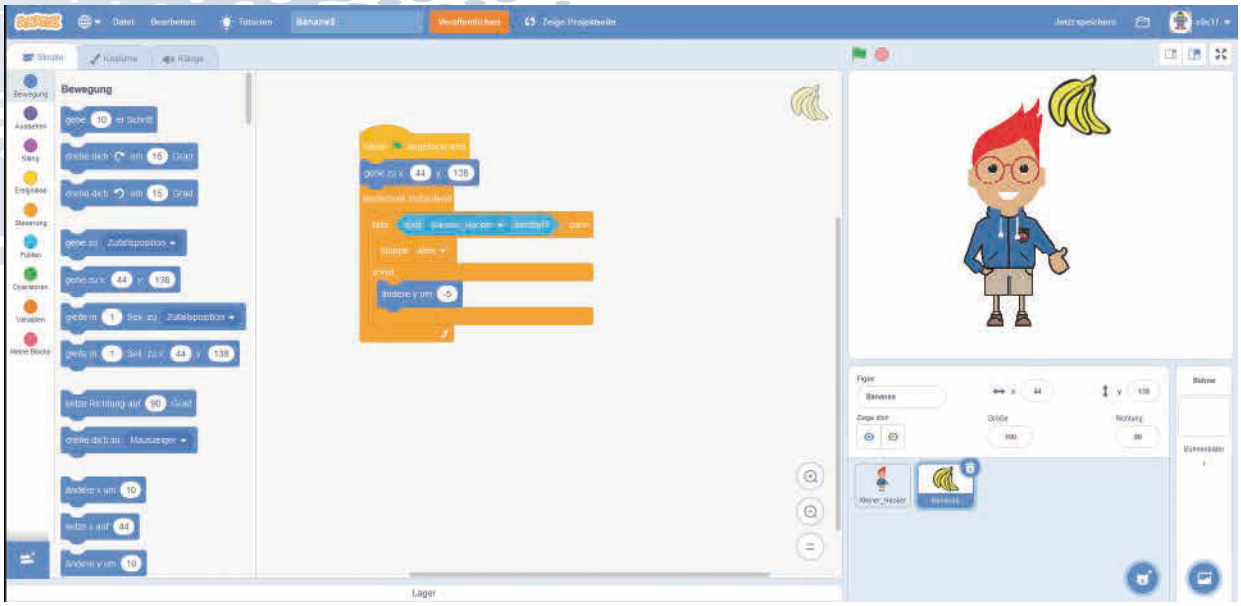


WENN EINE FIGUR AUS DEM BILD GERÄT ...

... ist sie noch nicht verloren. Du kannst sie zwar nicht mehr mit der Maus greifen, aber mit einem einzelnen Programmblock wieder ins Bild bringen. Trage in die beiden Zahlenfelder im Block *gehe zu x: ...y: ...* auf der blauen Blockpalette *Bewegung* die Zahl *0* ein und klicke einmal auf den Block. Du brauchst den Block dazu nicht in das Skriptfenster zu ziehen. Die Figur wird, egal wo sie vorher war, in die Mitte der Scratch-Bühne bewegt.

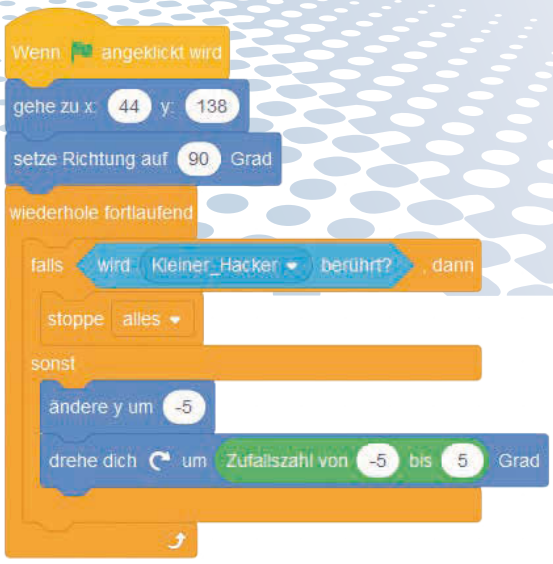
In Wirklichkeit fallen Gegenstände einfach senkrecht herunter. In Trickfilmen wackeln sie meistens beim Fallen, was irgendwie besser aussieht, obwohl es physikalisch nicht stimmt – außer bei starkem Wind.

Mit zwei zusätzlichen Programmblöcken lassen wir die Bananen im freien Fall noch zufällig ein bisschen wackeln:



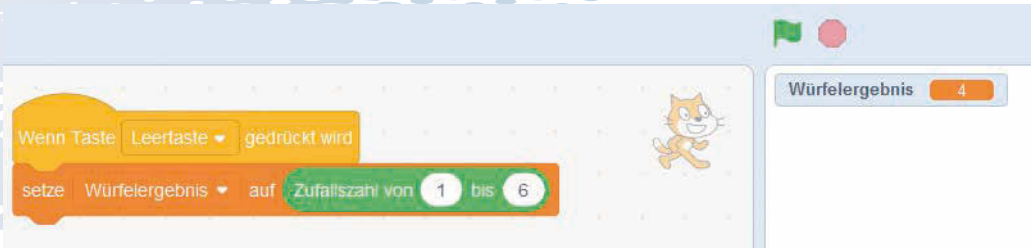
1 Ziehe in den unteren Programmblock der ***falls ... sonst***-Abfrage hinter die Bewegung noch einen ***drehe dich...***-Block, in dem der Drehwinkel zwischen ***-5*** und ***5*** Grad zufällig festgelegt wird.

2 Da die Bananen jetzt in einem zufälligen Winkel in der Hand des kleinen Hackers ankommen, müssen sie beim nächsten Start wieder gerade ausgerichtet werden. Das erledigt ein Block ***setze Richtung auf 90 Grad*** am Anfang, nachdem die Bananen an die Ausgangsposition geschoben wurden.



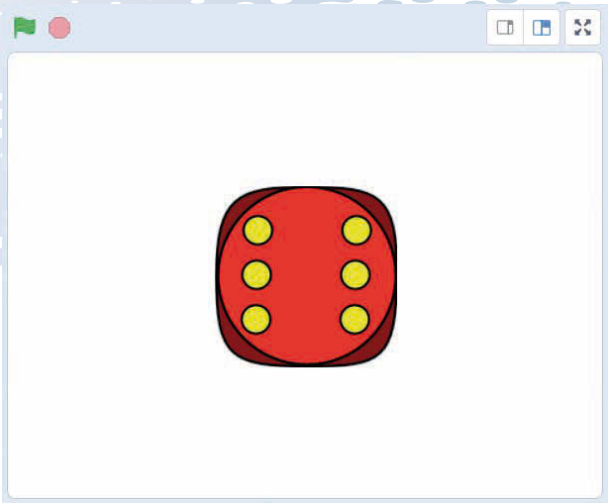
4 Spielwürfel

Für viele Spiele braucht man einen Würfel, aber oft ist gerade keiner griffbereit. Das nächste Programm zeigt, wie einfach es ist, mit Scratch einen Würfel zu programmieren. Natürlich könntest du auch einfach ein Programm ohne jegliche Grafik schreiben, das in einer Variablen eine Zahl zwischen 1 und 6 schreibt:



Dieses Programm erfüllt die gestellte Aufgabe, schön anzusehen ist es aber wirklich nicht. Großartig zu erklären, wie es funktioniert, brauchen wir jedoch auch nicht ...

Das nächste Programm ist kaum aufwendiger programmiert, zeigt aber einen „echten" Würfel in ansprechender Vektorgrafik, der, wenn man ihn anklickt, jedes Mal eine neue Zahl anzeigt.



WÜRFEL ZEICHNEN

Diesmal werden wir die Grafik nicht importieren, sondern sie in Scratch selbst zeichnen.

1. Lege in Scratch ein neues Projekt an und lösche die Katze, indem du auf die Figur ***Figur1*** in der Figurenliste und dann auf das blaue Papierkorbsymbol klickst.

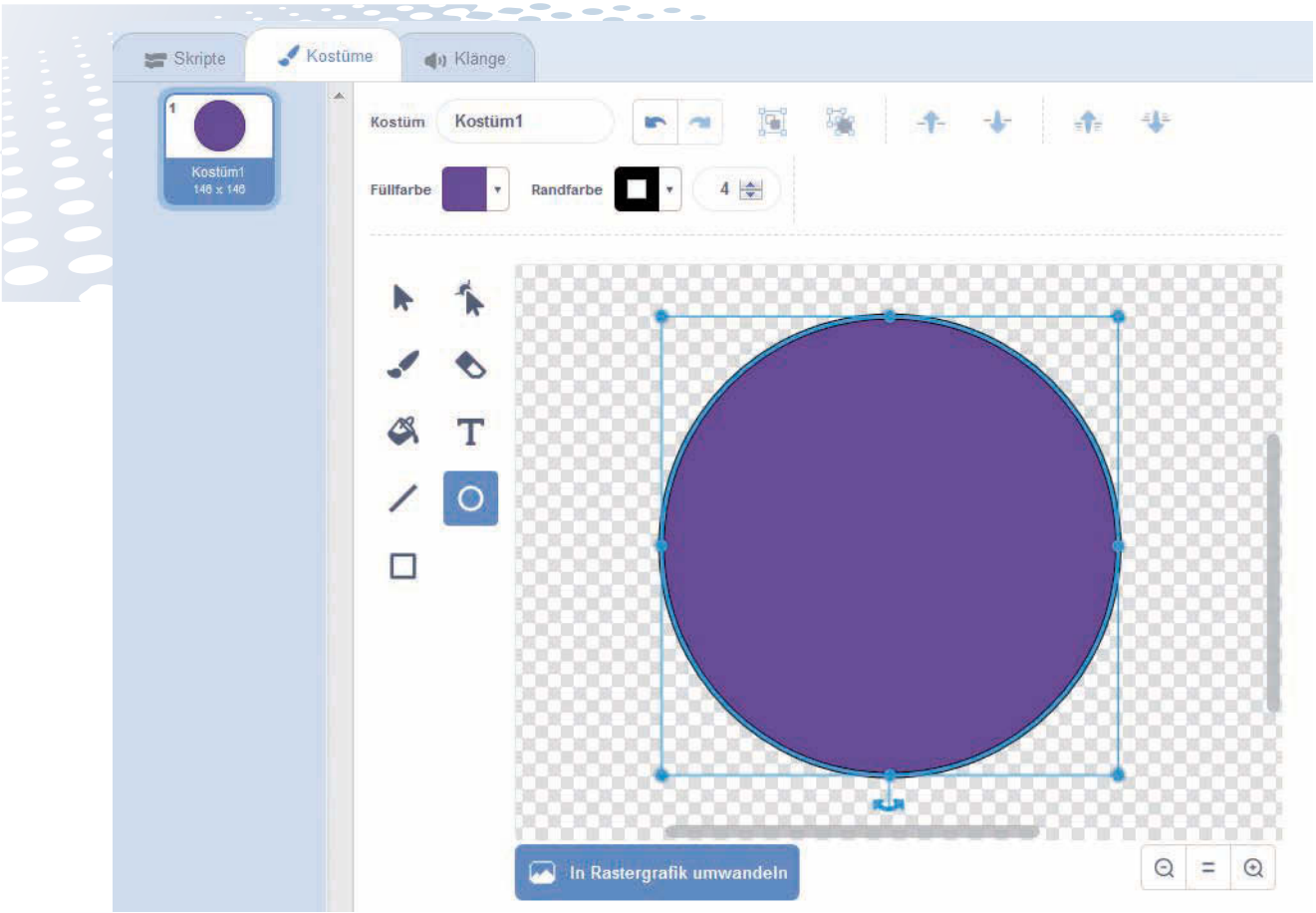
2. Fahre mit der Maus auf das Symbol ***Figur wählen*** und klicke dann auf ***Malen***.

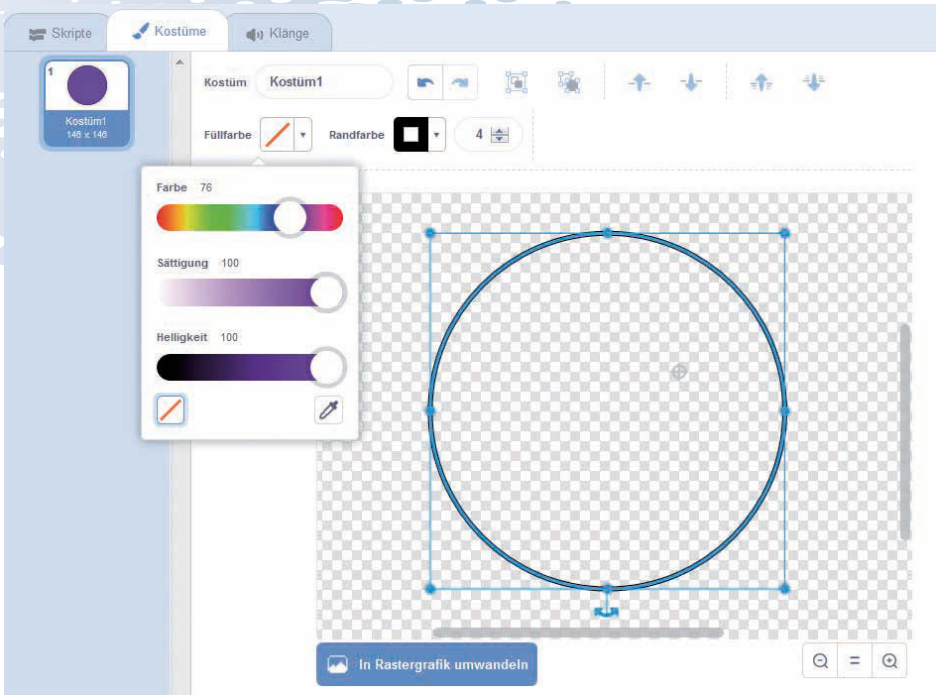
3. Das Grafikprogramm startet automatisch im Vektorgrafikmodus. Vektorgrafiken haben gegenüber Rastergrafiken den großen Vorteil, dass sich die einzelnen Objekte nachträglich noch

verändern und verformen lassen. Außerdem können Vektorgrafiken, ohne pixelig zu werden, beliebig vergrößert oder verkleinert werden. Zoome unten rechts einmal. So kannst du besser zeichnen, denn das Bild im Grafikbereich ist jetzt doppelt so groß wie die Figur auf der Bühne.

4 Klicke auf das Kreissymbol und halte die Umschalt-Taste gedrückt. Mit dem Kreissymbol kannst du beliebige Ellipsen zeichnen, bei gedrückter Umschalt-Taste werden es echte Kreise.

5 Klicke auf das Symbol ***Füllfarbe*** und schalte sie mit einem Klick auf den roten Schrägstrich unten links aus.

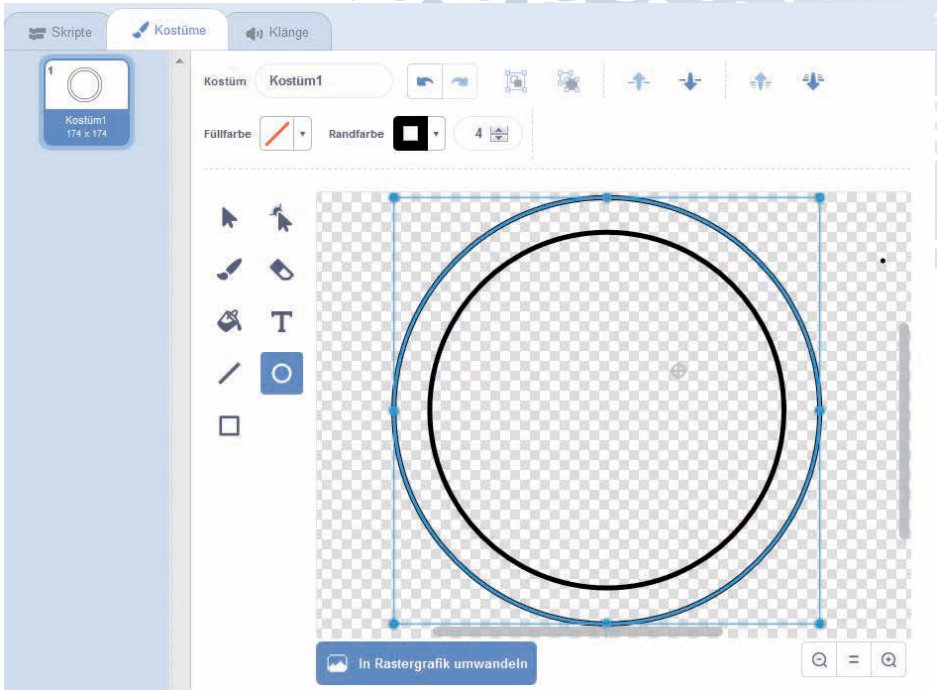


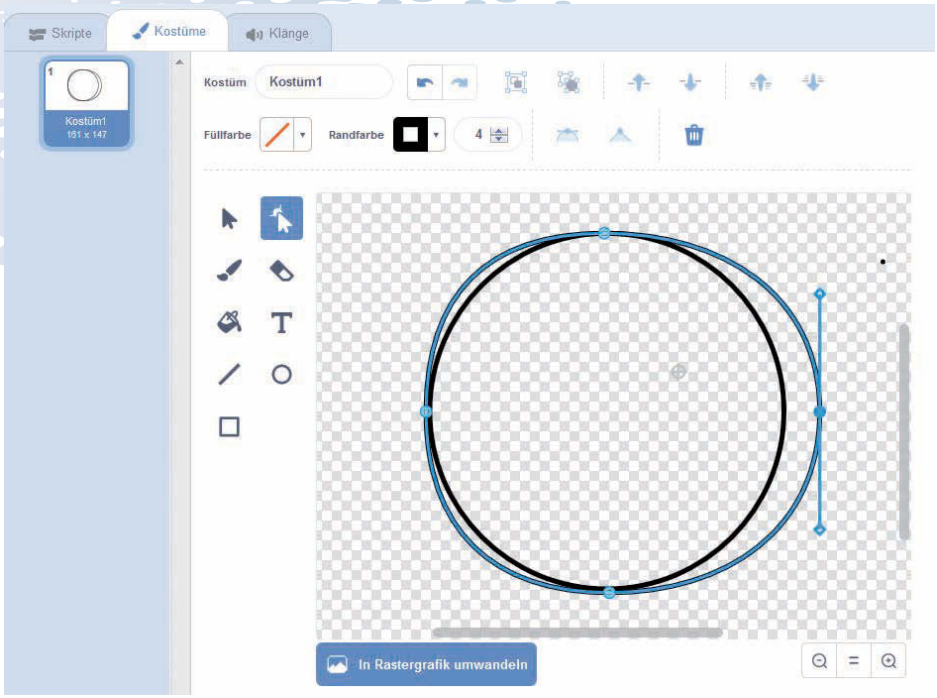


8 Klicke, solange der neue Kreis ausgewählt ist, auf das Symbol ***Verformen***. Auf der Kreislinie erscheinen runde Griffpunkte, mit denen du den Kreis verformen kannst. Ziehe den oberen, den unteren, den linken und den rechten Punkt nach innen auf den kleineren Kreis, um die typische Draufsicht eines Spielwürfels zu erzeugen.

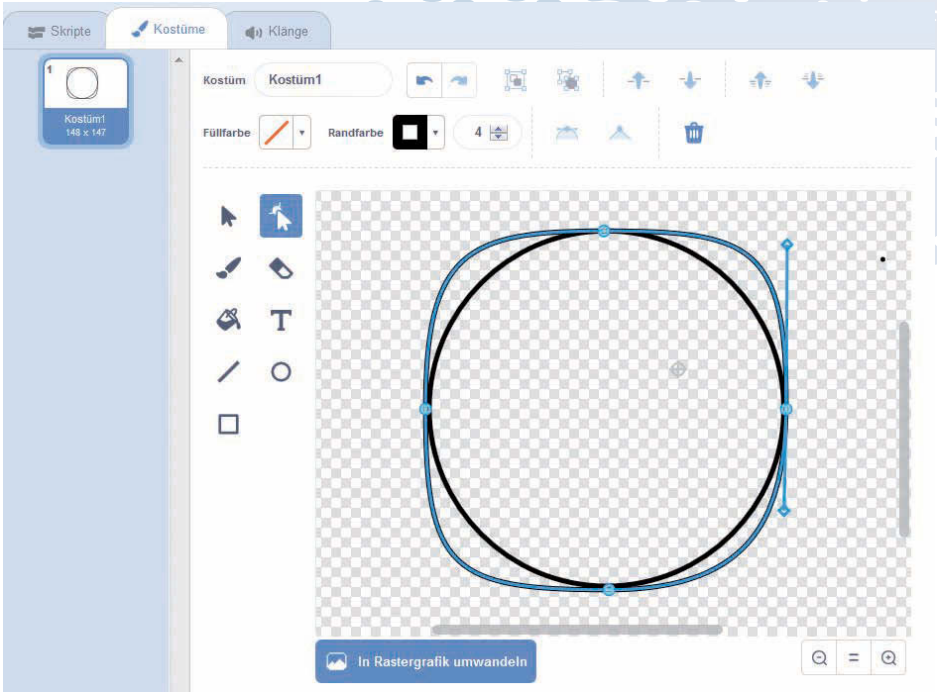
6 Klickst du auf einen der Griffe in den Ecken, kannst du den Durchmesser des Kreises durch Ziehen verändern, ziehst du den Kreis in der Mitte, kannst du ihn im Ganzen verschieben.

7 Klicke, nachdem der Kreis fertig ist, irgendwo auf eine freie Fläche im Grafikbereich, um ihn loszulassen. Zeichne einen weiteren Kreis mit gleichem Mittelpunkt und einem etwas größeren Durchmesser.

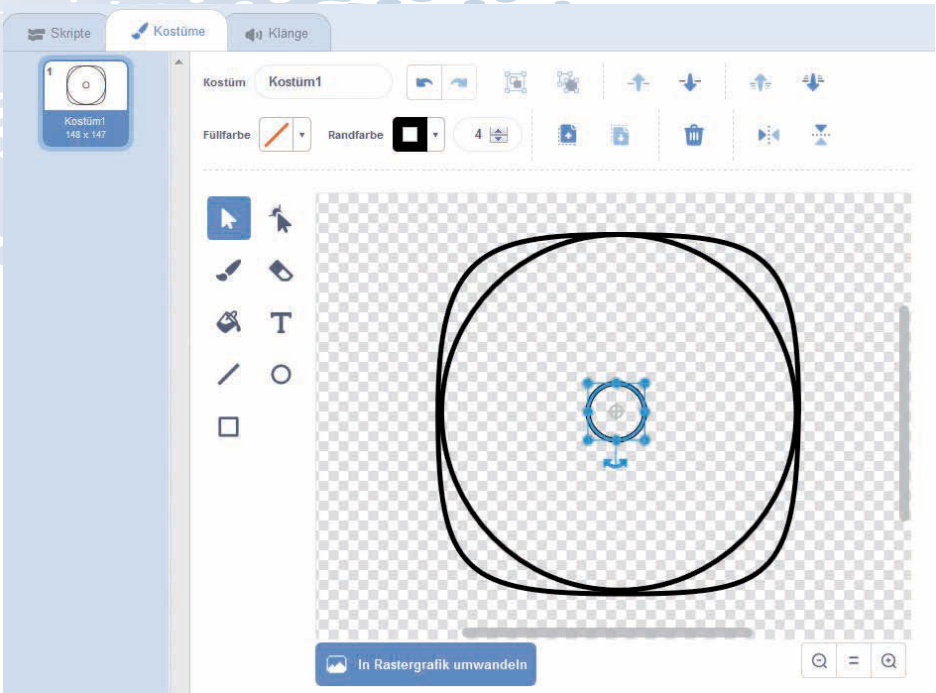




9 Beim Ziehen erscheint am Griff eine Linie. Ziehe sie an den Enden in die Länge, um den Würfel noch etwas quadratischer zu formen.



10 Zeichne einen weiteren kleinen Kreis in die Mitte, der das Würfelauge einer gewürfelten 1 darstellt. Dieses Würfelauge wird für alle ungeraden Zahlen gebraucht.

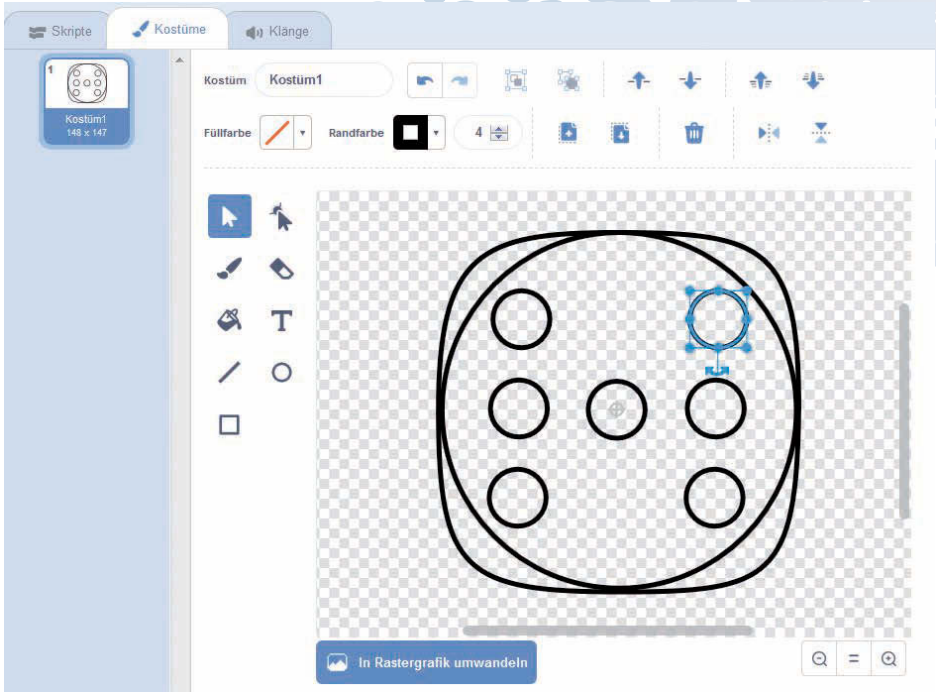


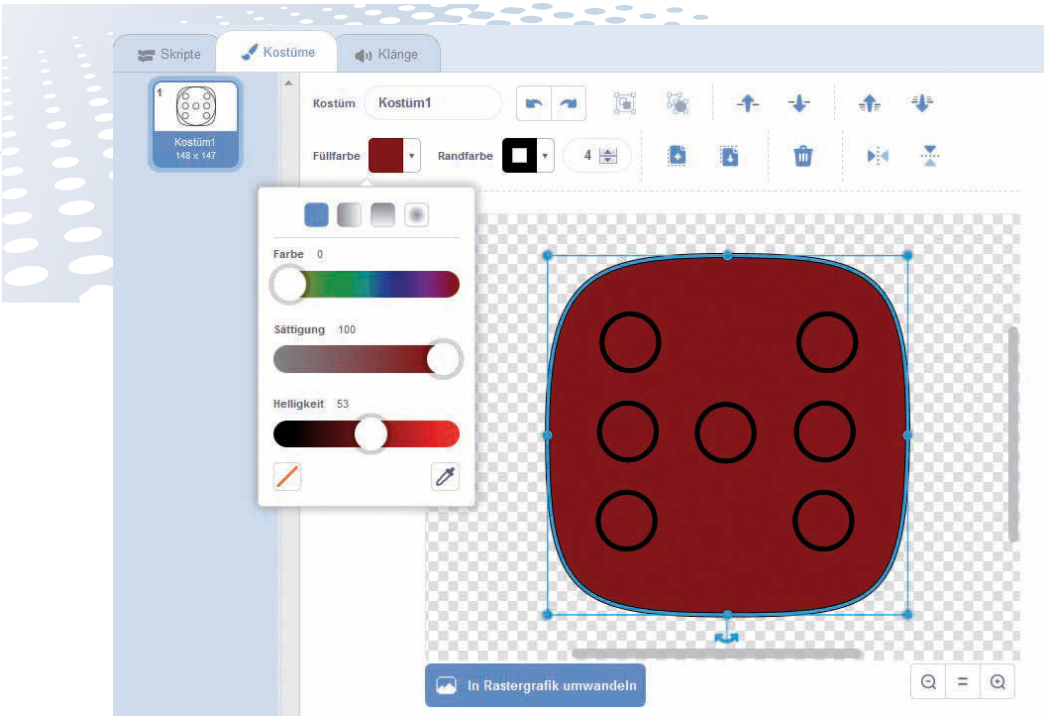
kannst du mehrere Objekte gleichzeitig kopieren und wieder einfügen.

12 Fülle den Würfel als Nächstes mit Farbe. Wähle mit dem Symbol ***Auswählen*** die äußere Umrandung des Würfels. Wähle dann mit dem Symbol ***Füllfarbe*** ein dunkles Rot aus. Die ausgewählte Fläche wird eingefärbt.

13 Beim Füllen des äußeren Bereichs des Würfels verschwindet der innere Kreis. Das liegt daran, dass

11 Damit alle Würfelaugen gleich groß werden, kopiere eines, anstatt alle einzeln zu zeichnen. Zum Kopieren kannst du die unter Windows bekannten Tastenkombinationen Strg + C und Strg + V oder die Symbole ***Kopieren*** und ***Einfügen*** verwenden. Damit wird ein ausgewähltes Vektorobjekt, wie zum Beispiel der Kreis, dupliziert. Anschließend kannst du die Kopie an die gewünschte Stelle schieben. Mit gedrückter Umschalt-Taste



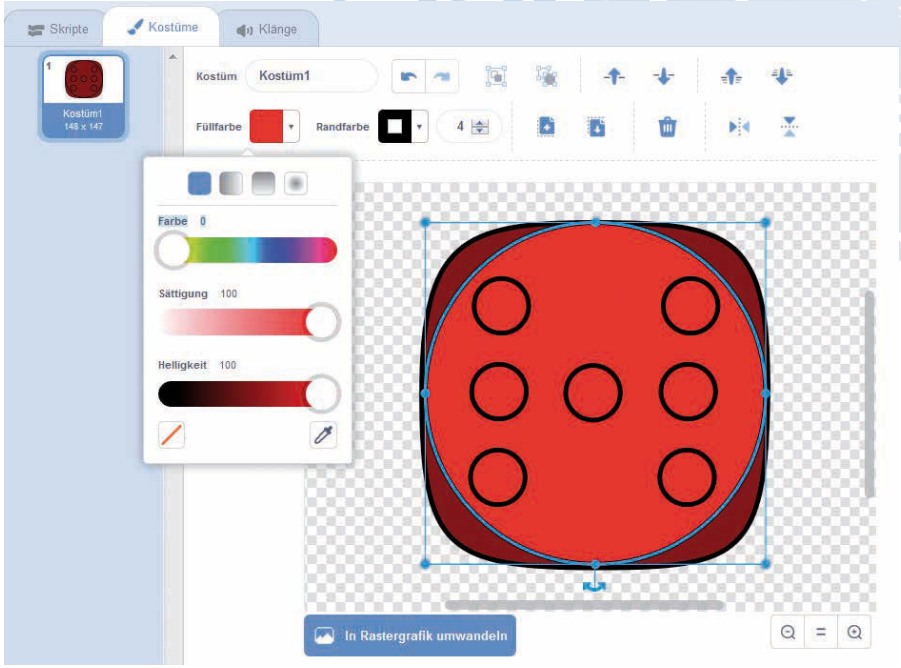


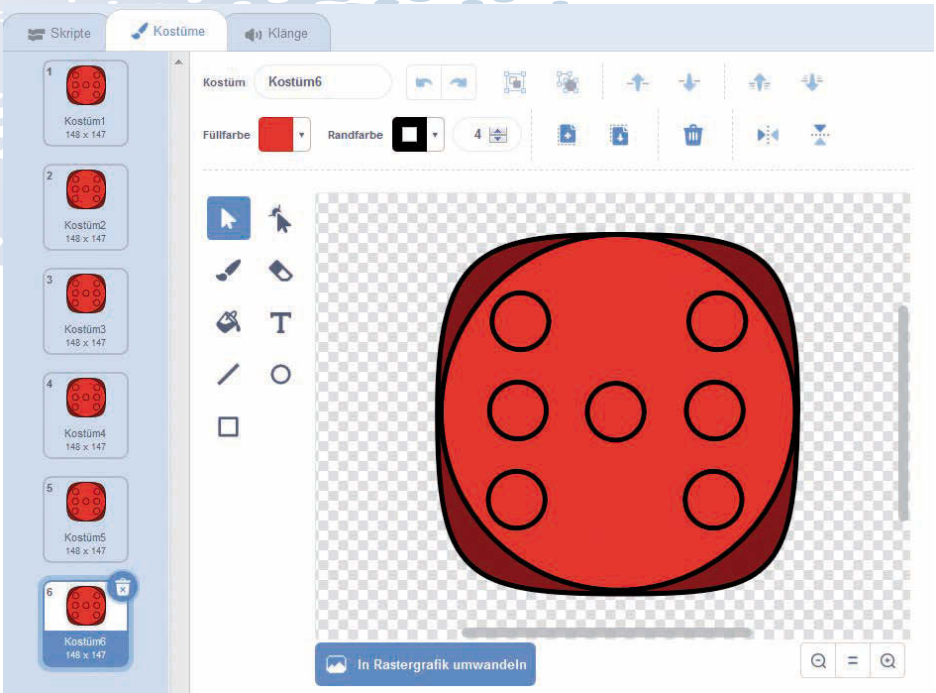
wurden später gezeichnet, liegen also weiter oben und sind vor der roten Fläche weiterhin zu sehen.

15 Das Programm zum Würfeln wird später einfach unterschiedliche Kostüme des Würfels anzeigen, auf denen die Würfelzahlen 1 bis 6 erscheinen. Dupliziere das aktuelle Kostüm fünfmal, bevor du anfängst, die Würfelaugen auszumalen. Klicke dazu mit der rechten Maustaste auf das ***Kostüm1*** in der Spalte mit den Kostü-

dieser zuerst gezeichnet wurde. Vektorobjekte liegen wie Folien übereinander. Solange sie nicht mit einer Farbe gefüllt sind, sieht man die darunterliegenden Ebenen. Klicke, solange die äußere Umgrenzungslinie des Würfels ausgewählt ist, auf das Symbol ***nach hinten***. Damit wandert die rot gefärbte Fläche nach hinten, und der große Kreis ist wieder zu sehen.

14 Wähle den Kreis aus und fülle ihn mit einem kräftigeren, helleren Rotton. Die kleinen Kreise der Würfelaugen

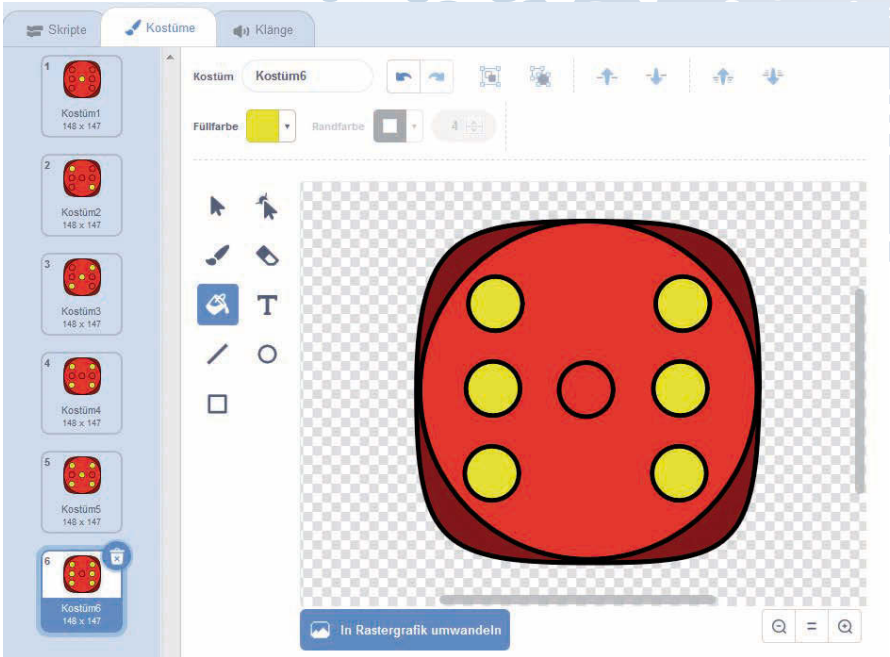




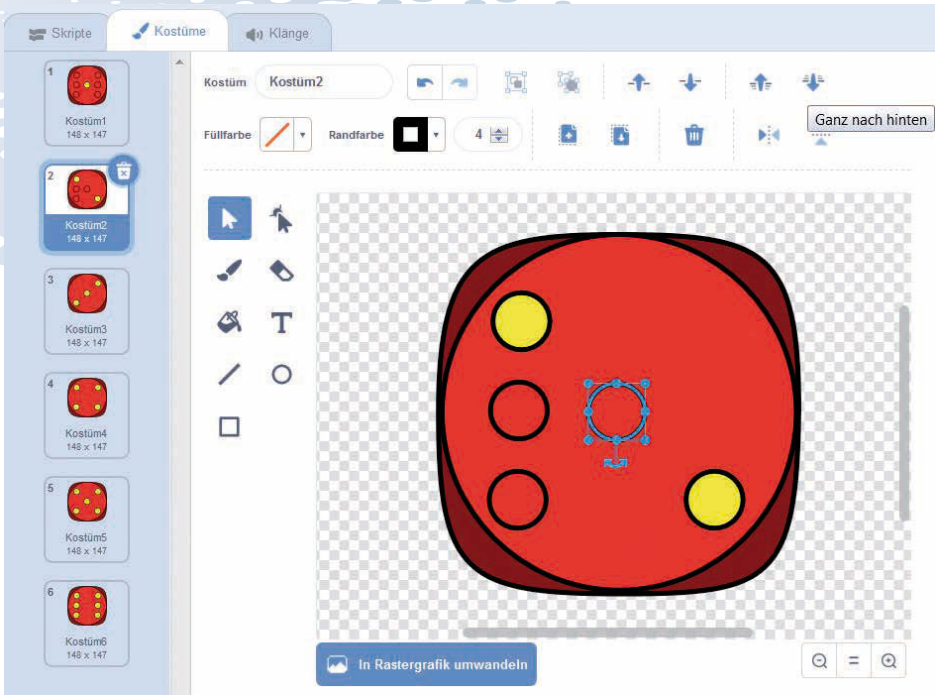
die nicht benutzten Würfelaugen auf den Kostümen stehen lassen oder einfach löschen. Du kannst sie auch auswählen und mit dem Symbol ***Ganz nach hinten*** ganz unten platzieren, sodass sie hinter der rot gefüllten Fläche verschwinden, aber jederzeit wieder hervorgeholt werden können. Eine Ebene nach hinten reicht nicht aus, da für jedes neu gezeichnete Vektorobjekt eine neue Ebene angelegt wird. Das zuletzt gezeichnete Würfelauge liegt also sieben Ebenen vor der roten Fläche des Würfels.

men und wähle im Menü ***Duplizieren***.

16 Wähle nacheinander in allen Kostümen die entsprechenden Würfelaugen und fülle sie mit der Farbe Gelb, um die Zahlen von 1 bis 6 darzustellen. Besonders einfach geht das, wenn du einmal die Füllfarbe auswählst und dann auf das Farbeimersymbol klickst. Dann brauchst du die gewünschten Würfelaugen nur noch anzuklicken.



17 Je nach persönlichem Geschmack kannst du

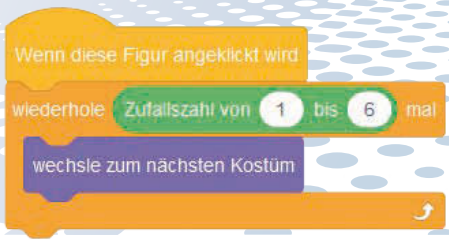


Fähnchen wird bei diesem Programm nicht gebraucht. Klicke einfach auf den Würfel.

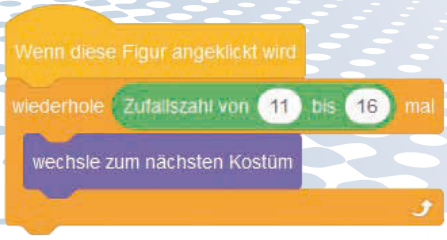
Bei diesem Zufallsgenerator treten alle sechs Ergebnisse mit der gleichen Wahrscheinlichkeit auf. Allerdings kann es passieren, dass man kein wirkliches Würfeln bemerkt, sondern das nächste Ergebnis sofort angezeigt wird. Das ist immer dann der Fall, wenn der Zufallsgenerator eine 1 liefert und somit direkt das nächste Kostüm gezeigt wird. Um diesen Effekt zu verhindern, ändere den Zufallsgenerator zum Beispiel auf ***Zufallszahl von 11 bis 16***.

DAS PROGRAMM FÜR DEN WÜRFEL

Das Programm für den Würfel ist sehr einfach und deutlich schneller zusammengebaut, als der Würfel gezeichnet ist.



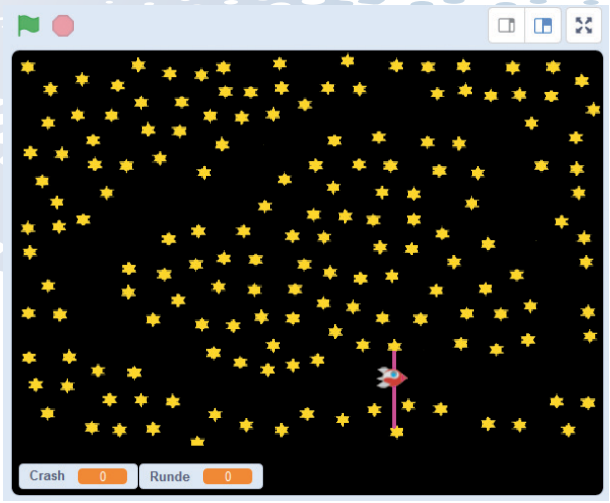
Wird der Würfel angeklickt, startet eine Schleife, die zufällig das jeweils nächste Kostüm des Würfels von 1 bis 6 anzeigt. Das letzte Kostüm, das stehen bleibt, zeigt das Würfelergebnis. Das grüne



Jetzt werden immer mindestens zehn Kostüme angezeigt und wieder weitergeschaltet, bevor frühestens das elfte endgültig stehen bleibt. Achte bei der Einstellung des Zufallsgenerators darauf, dass der Bereich genau sechs mögliche Zahlen enthält. Wie bei einem echten Würfel soll jede Zahl die gleiche Wahrscheinlichkeit haben.

5 Space Race – oder auf Deutsch: Raumschiffrennen

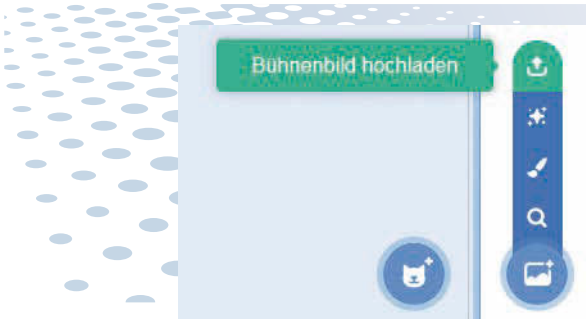
Ein Raumschiff rast durchs All und darf dabei an keinem Stern anstoßen. Wie viele Runden mit möglichst wenigen Crashs schaffst du? In diesem Projekt programmieren wir ein schnelles Rennspiel.



DER STERNENHIMMEL

Bei den Downloads zum Buch findest du ein Bild ***sternenhimmel.png***, das du als Hintergrund verwenden kannst.

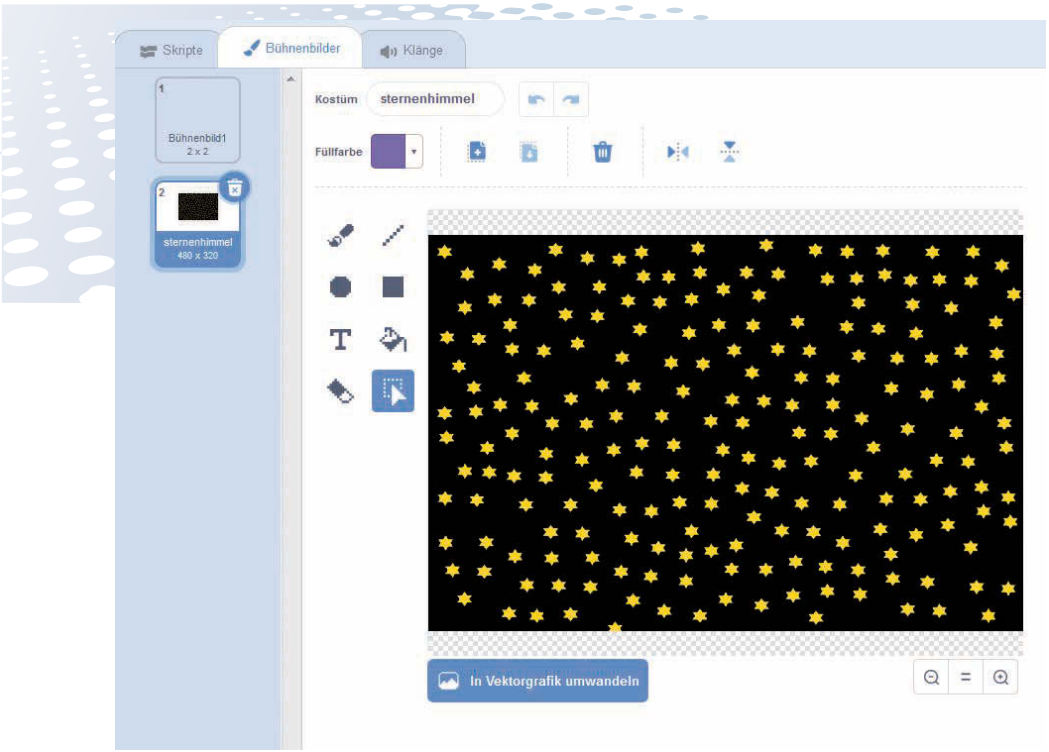
1 Starte ein neues Projekt in Scratch, klicke in der Figurenpalette auf die Bühne und fahre mit der Maus auf das Symbol ***Bühnenbild wählen***. Klicke dann auf ***Bühnenbild hochladen***. Wähle hier den Sternenhimmel aus.

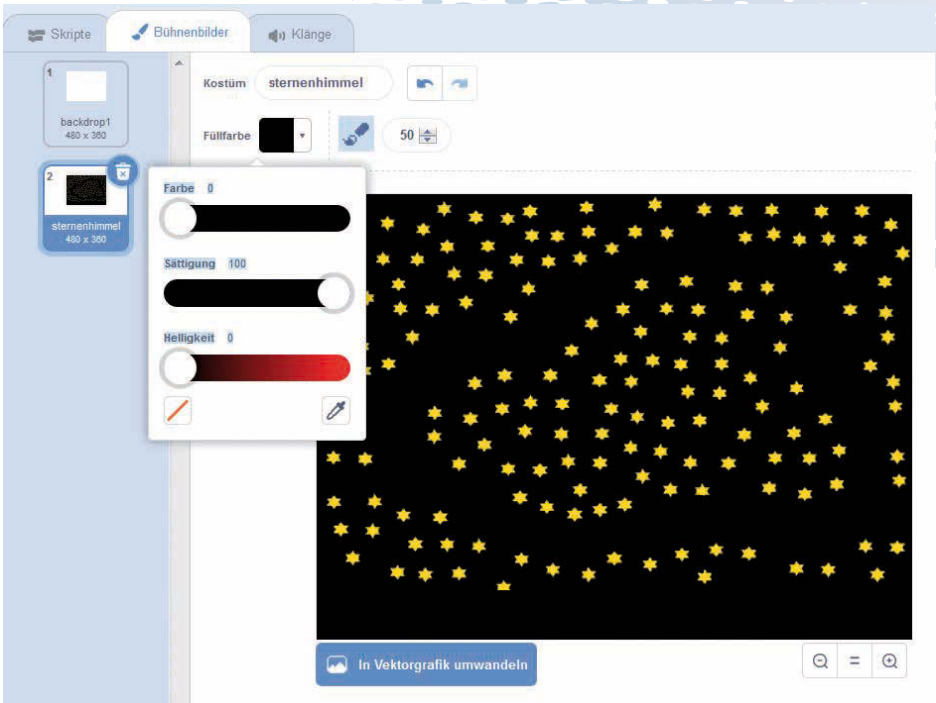


2 Der Sternenhimmel erscheint als Bühnenbild und öffnet sich gleichzeitig im Grafikprogramm.

3 Der Sternenhimmel ist gleichmäßig mit Sternen übersät. Übermale einige Sterne mit einem dicken schwarzen Pinsel, um Platz für einen Rundkurs zu machen, auf dem das Raumschiff fliegen kann. Wähle dazu das Pinselsymbol und stelle die Breite auf etwa ***50***.

4 Lösche gleich noch die Katze, sie kommt in diesem Spiel nicht vor.


Skripte
Bühnenbilder
Klänge
1
Bühnenbild1
2 x 2
2
sternenhimmel
480 x 320
Kostüm
sternenhimmel
Füllfarbe
In Vektorgrafik umwandeln


Skripte
Bühnenbilder
Klänge
1
backdrop1
480 x 360
2
sternenhimmel
480 x 360
Kostüm
sternenhimmel
Füllfarbe
50
Farbe 0
Sättigung 100
Helligkeit 0
In Vektorgrafik umwandeln

5 Space Race – oder auf Deutsch: Raumschiffrennen

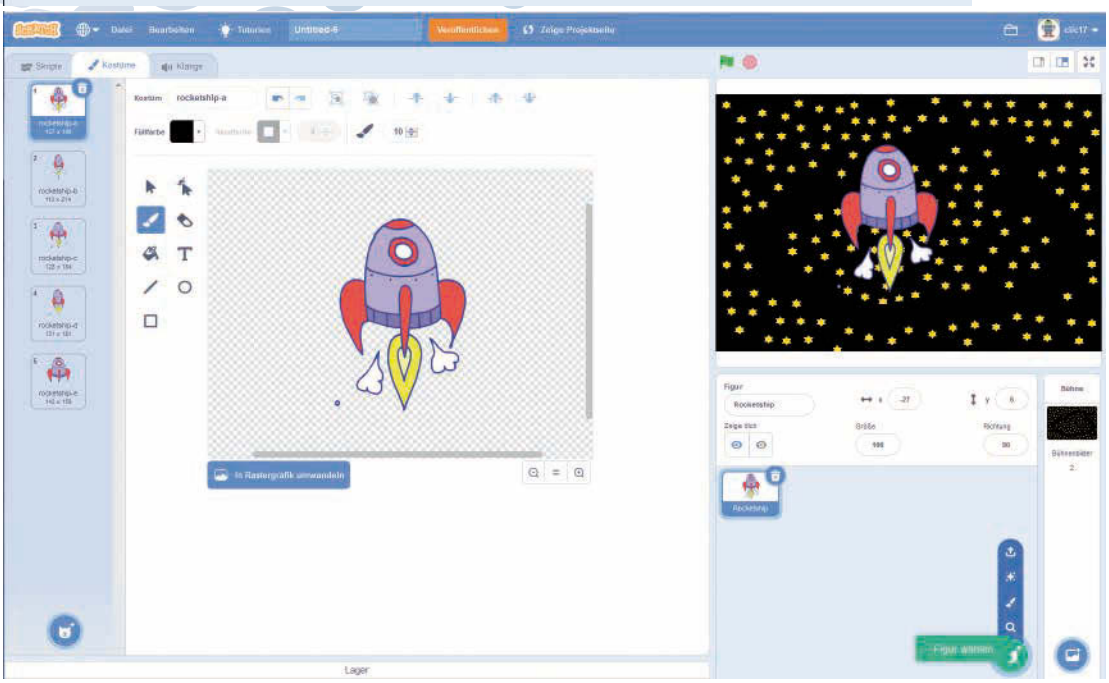
DAS RAUMSCHIFF

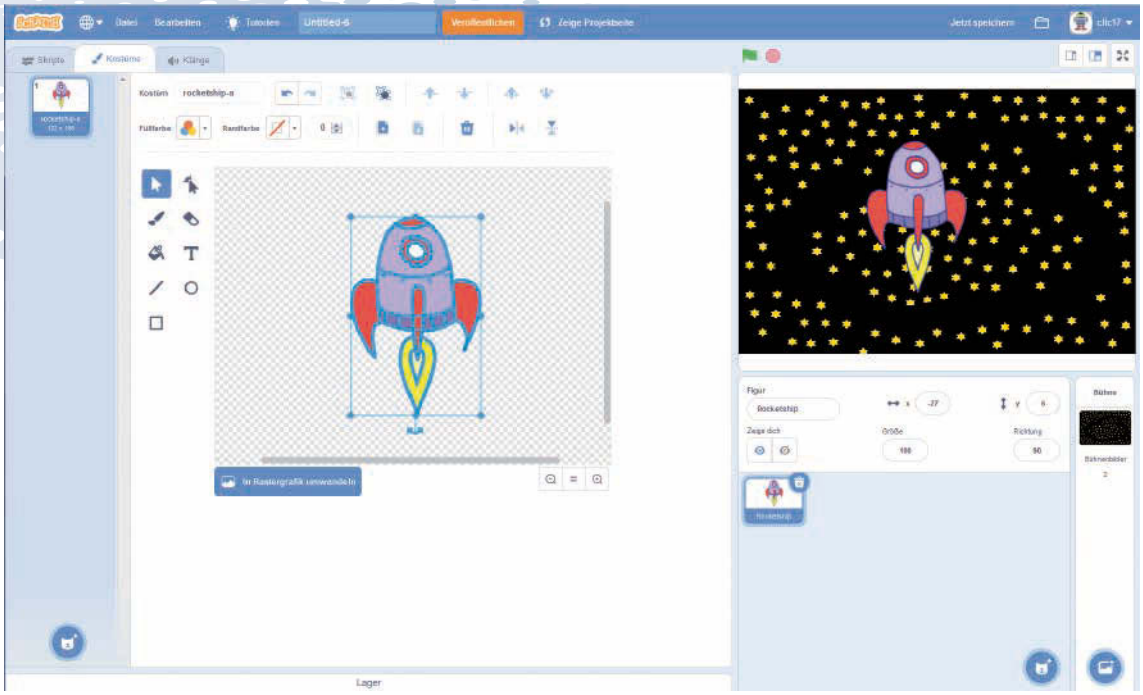
1 Scratch liefert bereits ein Raumschiff mit, das sich für dieses Spiel leicht modifizieren lässt. Klicke auf das Symbol ***Figur wählen***. Suche im Suchfeld die Figur ***Rocketship***.

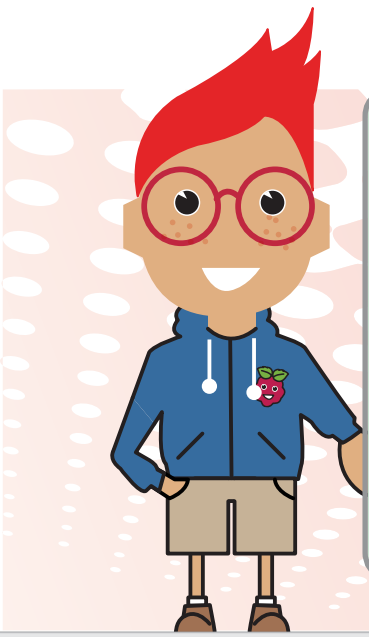
2 Das Raumschiff ist als Vektorgrafik gezeichnet. Lösche alle Kostüme des Raumschiffs mit Ausnahme des ersten, da wir sie in diesem Spiel nicht brauchen.

3 Das übrig gebliebene Kostüm besteht aus zwei getrennten Objekten. Wähle die Wolken aus und lösche sie mit dem Papierkorbsymbol. Jetzt ist das Raumschiff nur noch ein Objekt, das sich leichter bearbeiten lässt.

4 Das Raumschiff ist für unser Spiel viel zu groß. Ziehe es an einer der Ecken kleiner, bis es auf der Bühne auf den zu fliegenden Kurs passt. Fasse es jetzt an dem runden Griff unten mit der Maus an und drehe es in die Waagerechte mit Flugrichtung nach rechts. Das Raumschiff fliegt im Spiel im Kreis, daher mag man auf den ersten Blick denken, die Ausrichtung der Figur wäre egal – das ist sie aber nicht.





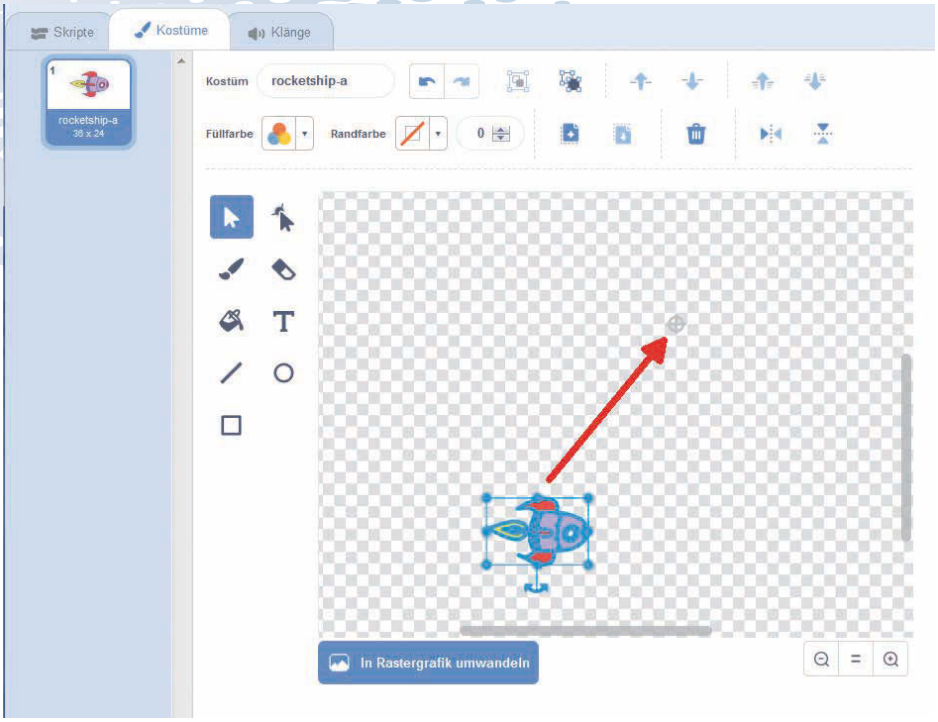


BEWEGUNGEN UND RICHTUNGEN IM SCRATCH-KOORDINATENSYSTEM

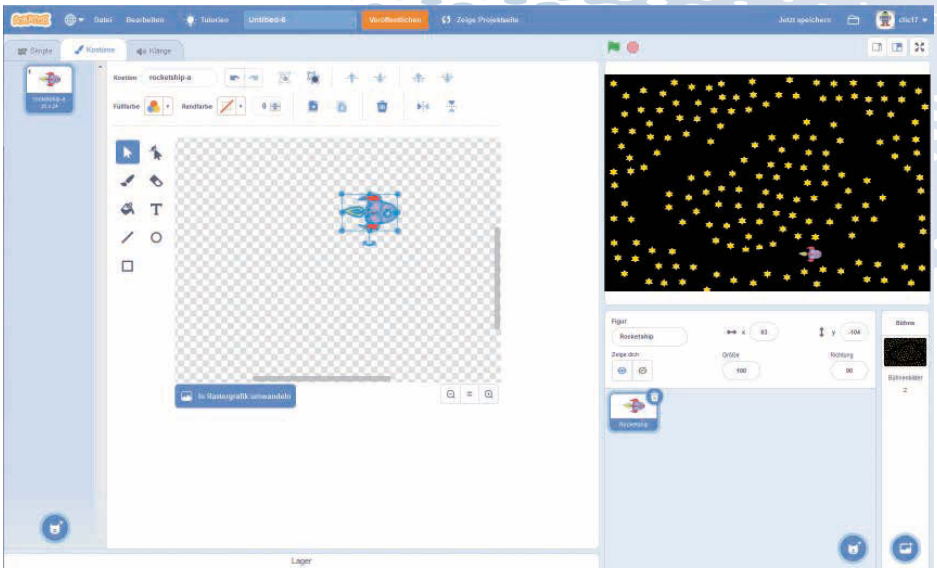
Die Bewegungsrichtung einer Figur zeigt standardmäßig immer nach rechts, wie an der Katze zu sehen ist. Das Raumschiff würde sich also seitlich zur eigentlichen Flugrichtung bewegen. Indem du das Kostüm drehst, kann sich das Raumschiff weiterhin nach rechts bewegen, es sieht aber aus, als würde es richtig fliegen. Drehst du dann die ganze Figur auf der Scratch-Bühne, kann das Raumschiff auch nach oben und in alle anderen Richtungen fliegen, da das interne Koordinatensystem der Figur mit gedreht wird.

Beim Drehen ist außerdem der Drehpunkt wichtig, der für jede Figur festgelegt ist. Er befand sich vor der Verkleinerung mitten im Raumschiff und ist jetzt nach außen gerutscht. Er wird als kleines Kreuz im Grafikfenster dargestellt.

5 Space Race – oder auf Deutsch: Raumschiffrennen



5 Verschiebe das Raumschiff so, dass sein Mittelpunkt auf dem im Grafikfenster gezeigten Drehpunkt liegt. Eventuell musst du etwas zoomen, um den Drehpunkt zu finden.



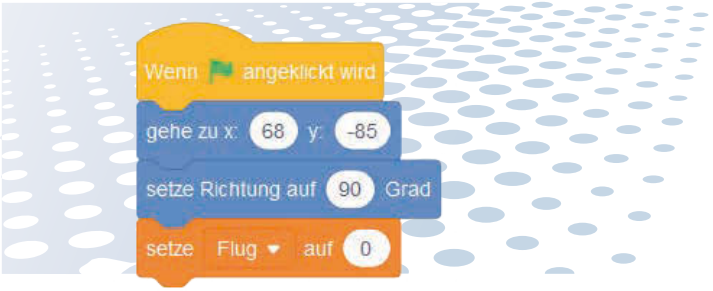
6 Ziehe das Raumschiff auf der Bühne an die Startposition. In der Figurenliste siehst du die Koordinaten und kannst auch die Drehrichtung einstellen.

DIE STEUERUNG

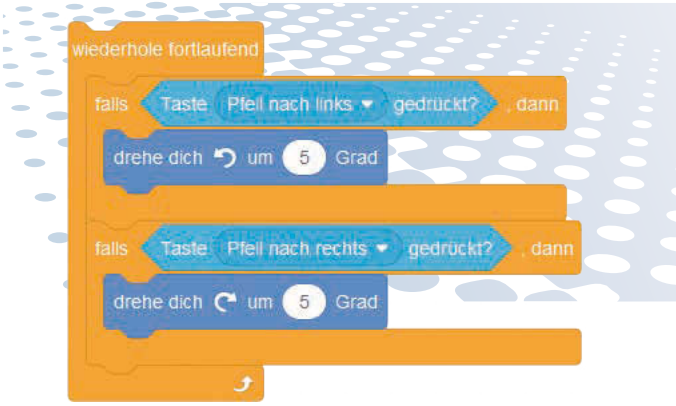
Das Raumschiff wird mit den Pfeiltasten gesteuert. Pfeil nach links und Pfeil nach rechts lenken nach links und rechts, Pfeil nach oben beschleunigt, Pfeil nach unten bremst.

1 Wenn du später im Spiel auf das grüne Fähnchen klickst, soll das Raumschiff an die Startposition gesetzt werden, egal wo es sich gerade befindet. Ziehe dazu zwei Blöcke ***gehe zu x: ... y: ...*** und ***setze Richtung auf 90 Grad*** ins Skriptfenster. Wenn das Raumschiff an der richtigen Stelle steht, werden die Koordinaten automatisch in den Programmblock übernommen.

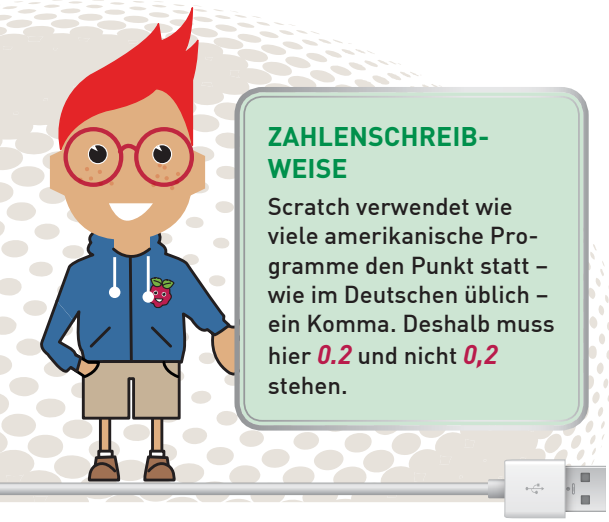
2 Zur Einstellung der Fluggeschwindigkeit verwenden wir eine Variable. Lege auf der Blockpalette ***Variablen*** mit einem Klick auf ***Neue Variable*** eine neue Variable mit Namen ***Flug*** an und hänge anschließend den Block ***setze Flug auf 0*** an das Programm an. Am Anfang steht das Raumschiff still, hat also eine Fluggeschwindigkeit von ***0***.



3 Anschließend folgt eine ***wiederhole fortlaufend***-Schleife, in der die Pfeiltasten abgefragt werden. Baue hier zunächst hintereinander zwei ***falls ..., dann***-Blöcke ein, die prüfen, ob die Tasten ***Pfeil nach links*** oder ***Pfeil nach rechts*** gedrückt wurden. Wurde die Taste ***Pfeil nach links*** gedrückt, soll sich das Raumschiff um 5 Grad nach links (gegen den Uhrzeigersinn) drehen, bei der Taste ***Pfeil nach rechts*** um 5 Grad nach rechts (im Uhrzeigersinn).



5 Space Race – oder auf Deutsch: Raumschiffrennen



ZAHLENSCHREIBWEISE

Scratch verwendet wie viele amerikanische Programme den Punkt statt – wie im Deutschen üblich – ein Komma. Deshalb muss hier ***0.2*** und nicht ***0,2*** stehen.

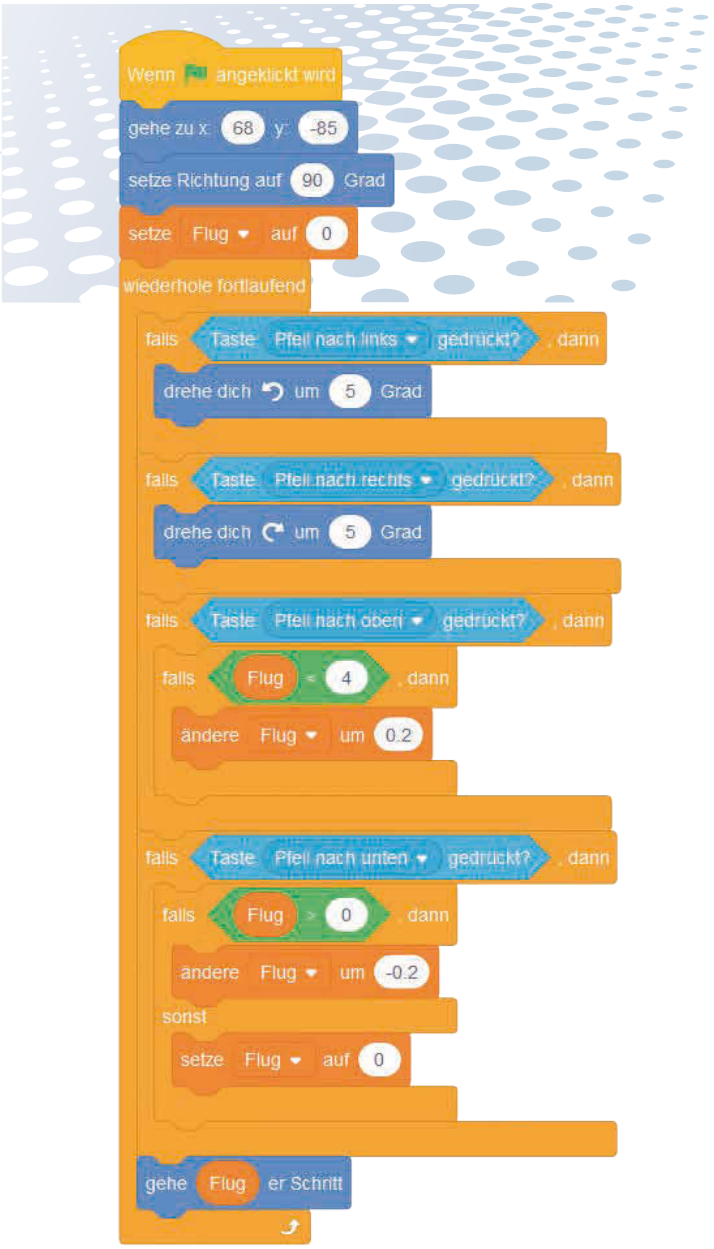
4 Beim Drücken der Taste ***Pfeil nach oben*** soll das Raumschiff beschleunigen. Die Geschwindigkeit, also eigentlich die Länge eines einzelnen Bewegungsschritts, wird in der Variablen ***Flug*** gespeichert. Bei jedem Tastendruck wird diese Geschwindigkeit um ***0.2*** erhöht.

5 Wenn das Raumschiff zu schnell fliegt, gerät es leicht außer Kontrolle. Deshalb begrenzen wir die Geschwindigkeit auf ***4***. Nur solange die Variable ***Flug*** kleiner als ***4*** ist, soll die Geschwindigkeit erhöht werden. Um das zu prüfen, kommt in die ***falls ..., dann***-Abfrage mit der Tastaturprüfung eine weitere ***falls ..., dann***-Abfrage.

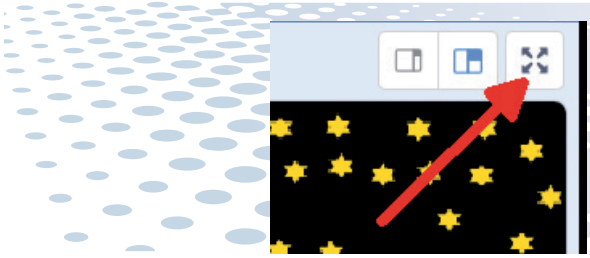
6 Auf die gleiche Weise soll die Taste ***Pfeil nach unten*** das Raumschiff abbremsen. Es soll aber nicht rückwärts fliegen, wenn die Geschwindigkeit immer kleiner wird, sondern auch bei hektischem Drücken der Taste mit einer Geschwindigkeit von ***0*** zum Stehen kommen. Um eine Schubumkehr wirklich auszuschließen, verwenden wir eine ***falls ... sonst***-Abfrage, die das Raumschiff abbremst, solange die Geschwindigkeit größer als ***0*** ist, und im theoretisch gar nicht möglichen Fall eine negative Geschwindigkeit sicherheitshalber genau auf ***0*** setzt.

7 Nachdem alle Tasten in der Endlosschleife abgefragt wurden, soll sich das Raumschiff in jedem Schleifendurchlauf auch noch einen Schritt bewegen. Das gilt ebenfalls, wenn keine der Tasten gedrückt wurde. Die Schrittlänge entspricht der Variablen ***Flug*** und ist umso größer, je schneller das Raumschiff fliegt.

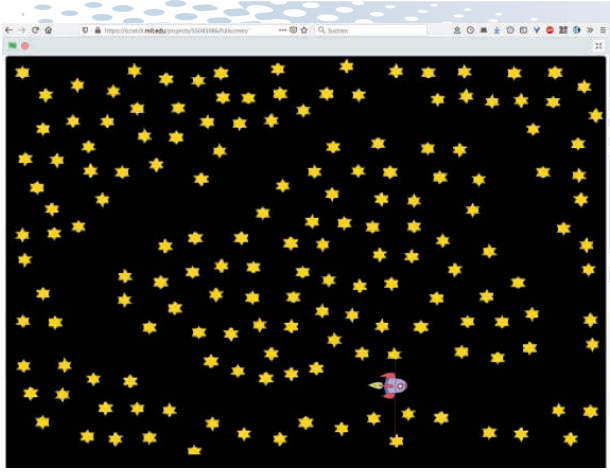
8 Das Programm sieht jetzt so aus:



9 Starte das Programm mit einem Klick auf das grüne Fähnchen. Jetzt kannst du das Raumschiff mit den Pfeiltasten fliegen. Dabei kannst du einfach über die Sterne fliegen. Das Raumschiff fliegt unbeirrt weiter.



10 Mit dem Symbol rechts oben kannst du die Scratch-Bühne auf das ganze Browserfenster vergrößern. Beim Spielen brauchst du das Skriptfenster und die Figurenliste schließlich nicht.

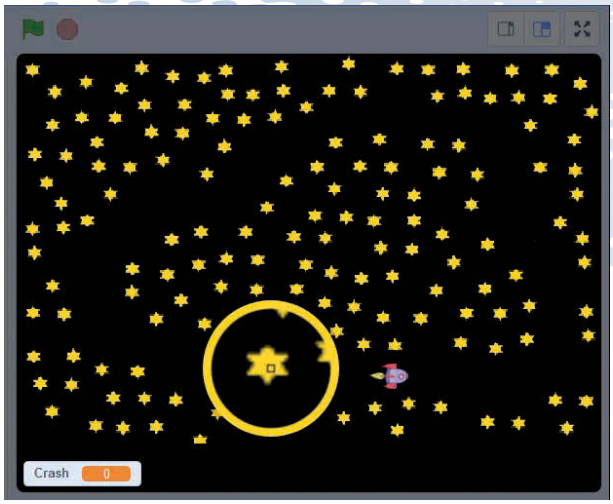


KOLLISIONSERKENNUNG

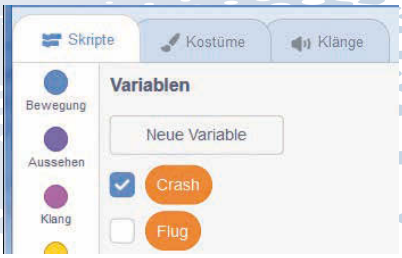
Die Spielregel besagt: Jedes Mal, wenn das Raumschiff an einen Stern stößt, kommt es zum Stillstand und bewegt sich ein kleines Stück zurück, damit du Gelegenheit hast, zu lenken und in die richtige Richtung wieder zu beschleunigen. Außerdem sollen die Crashs gezählt werden. Informatiker und Physiker sprechen von „Kollisionserkennung“, wenn es darum geht, festzustellen, ob sich

5 Space Race – oder auf Deutsch: Raumschiffrennen

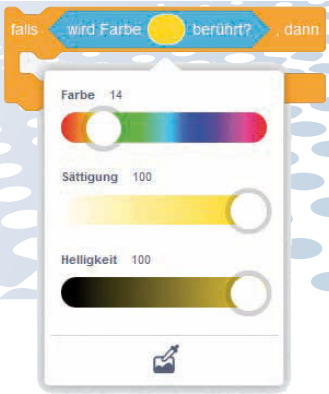
zwei Objekte in der Realität oder virtuell auf dem Bildschirm berühren. Scratch liefert dafür vorgefertigte Blöcke mit, die diese Kollisionserkennung ganz einfach machen.



1 Lege als Erstes auf der Blockpalette ***Variablen*** eine neue Variable ***Crash*** an, die die Zusammenstöße mit den Sternen zählt. Achte dabei darauf, dass das kleine Kästchen links neben dem Variablennamen angekreuzt ist. Damit wird diese Variable auf der Bühne angezeigt. Schiebe dieses Anzeigefeld in eine der Ecken, wo es die Flugbahn des Raumschiffs nicht behindert.



2 Baue in die Endlosschleife zusätzlich zu den Tastaturabfragen eine Abfrage ein, die prüft, ob das Raumschiff die Farbe der Sterne berührt. Klicke auf das farbige Feld im Block ***wird Farbe ... berührt***. Ein Feld zur Auswahl der Farbe erscheint. Klicke auf die Farbpipette ganz unten in diesem Feld. Klicke dann auf einen der Sterne auf der Bühne, und das Farbfeld nimmt dessen gelbe Farbe an.



3 Bei jedem Crash sollen drei Aktionen ablaufen:

- Der ***Crash***-Zähler wird um 1 hochgezählt.
- Das Raumschiff wird sofort auf 0 abgebremst
- Das Raumschiff bewegt sich um 10 Koordinateneinheiten rückwärts, um leichter wieder freizukommen.

4 Diese drei Aktionen werden durch drei Blöcke innerhalb der ***falls ..., dann***-Abfrage erledigt.

5 Am Anfang des Programms muss der ***Crash***-Zähler noch auf ***0*** gesetzt werden, damit du mit einem Klick auf das grüne Fähnchen wieder bei 0 anfangen kannst. Das Programm sieht jetzt so aus:

6 Starte das Programm mit einem Klick auf das grüne Fähnchen. Jetzt kannst du das Raumschiff mit den Pfeiltasten fliegen. Bei jedem Zusammenstoß mit einem Stern wird der ***Crash***-Zähler auf der Bühne um 1 hochgezählt.

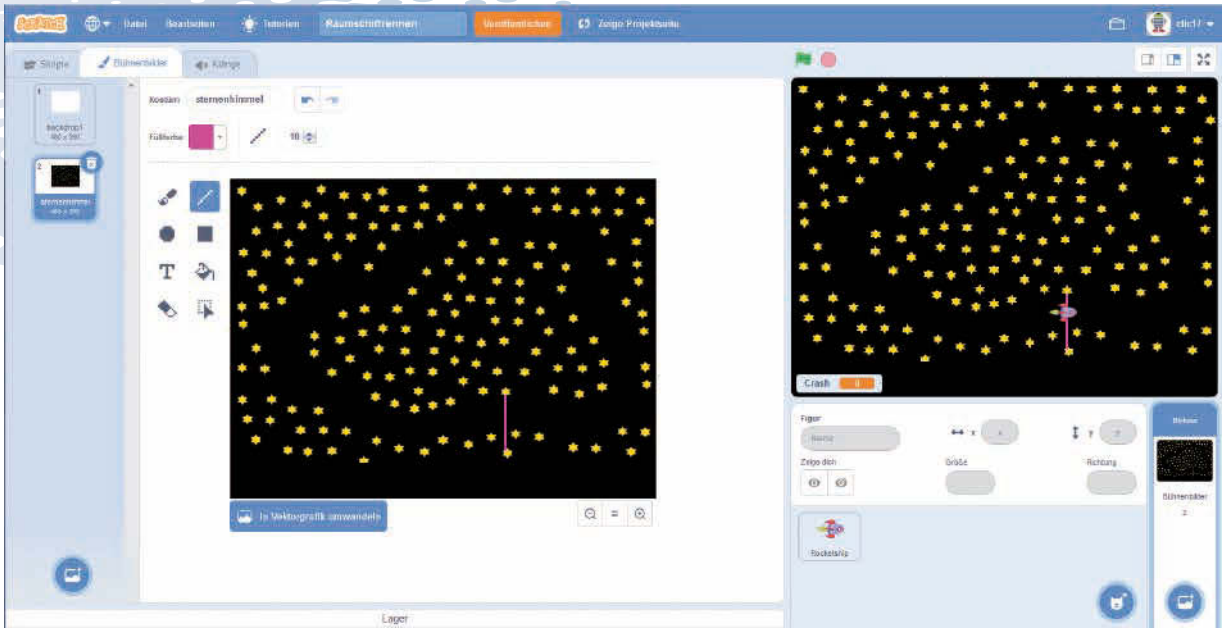
DER RUNDENZÄHLER

Ein Laserstrahl zwischen zwei Sternen dient als Start- und Ziellinie für das Raumschiffrennen. Jedes Mal, wenn das Raumschiff durch diesen Laserstrahl fliegt, wird ein Rundenzähler um 1 erhöht.

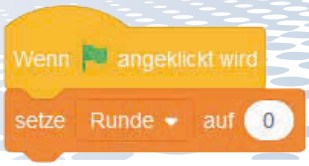
1 Klicke in der Figurenliste auf die Bühne und schalte oben auf die Seite ***Bühnenbilder***. Male in einer kräftigen Laserfarbe eine gut erkennbare Linie zwischen zwei Sterne an die Stelle, an der das Raumschiff beim Start steht.

2 Lege nun auf der Blockpalette ***Variablen*** eine weitere Variable namens ***Runde*** an, die die geflogenen Runden zählt. Auch diese Variable soll auf der Bühne zu sehen sein.

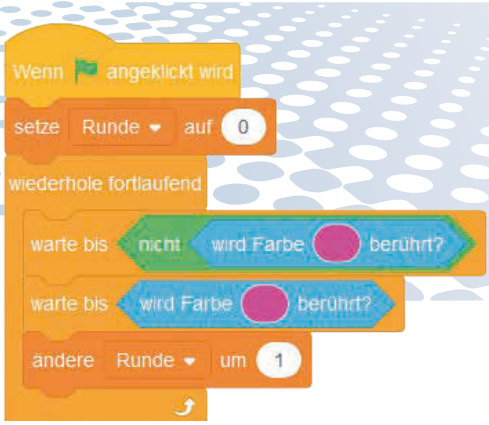
5 Space Race – oder auf Deutsch: Raumschiffrennen



3 Für den Rundenzähler verwenden wir ein eigenes Skript, das bei Klick auf das grüne Fähnchen ebenfalls gestartet wird und als Erstes den Rundenzähler auf ***0*** setzt.

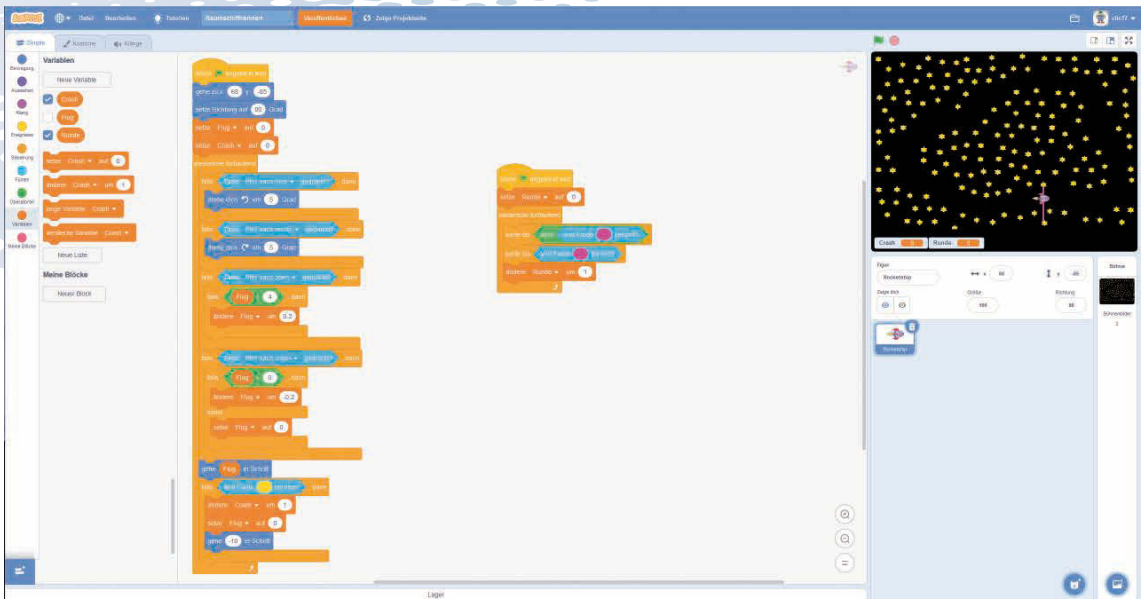


4 Um eine Runde zu zählen, kann nicht einfach nur überprüft werden, ob das Raumschiff den Laserstrahl berührt. Für den Flug durch den Strahl braucht es etwas Zeit. Es würden also bei jedem Durchflug mehrere Runden schnell hintereinander gezählt. Diese Aufgabe lässt sich mit ***warte bis ...***-Blöcken elegant lösen. Das Programm wartet am Anfang, bis das Raumschiff über die Startlinie geflogen ist, den Laserstrahl also nicht mehr berührt. Danach wartet es wieder, bis das Raumschiff den Laserstrahl berührt, und zählt dann eine Runde. Das Ganze wiederholt sich endlos und zählt jedes Mal eine Runde, wenn das Raumschiff erneut in den Laserstrahl fliegt, nachdem es ihn zwischendurch verlassen hat.



5 Der Operator ***nicht ...*** kann zu jeder beliebigen Abfrage feststellen, ob sie ***nicht wahr*** ist. Er liefert also in dem Moment das Ergebnis ***wahr***,

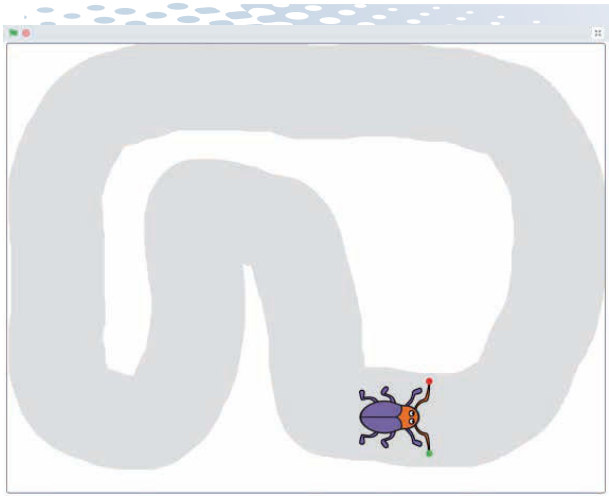
wenn das Raumschiff den Laserstrahl verlassen hat.



Starte das Programm mit einem Klick auf das grüne Fähnchen und fliege möglichst viele Runden mit möglichst wenigen Crashs. Mit der Pfeiltaste nach unten kannst du das Raumschiff jederzeit zum Stehen bringen, um dir eine kurze Verschnaufpause zu gönnen. Der Rundenzähler zählt danach weiter.

Erst wenn du das Programm mit dem Stoppsymbol anhältst und danach mit dem grünen Fähnchen neu startest, beginnen alle Zähler wieder bei ***0***.

6 Ein Käfer sucht sich seinen Weg

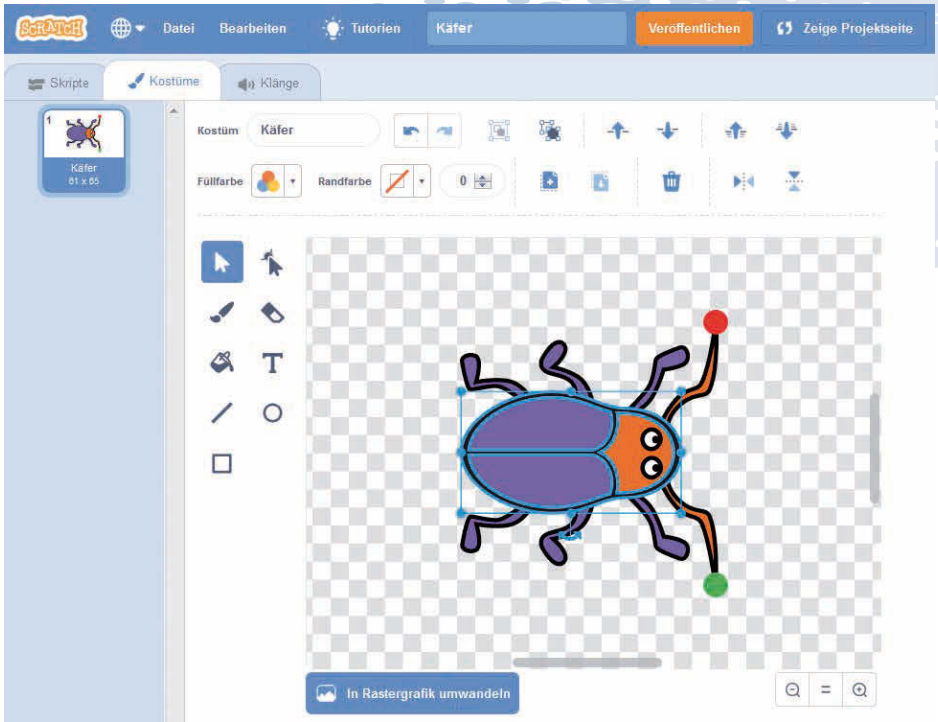


Bei der Steuerung von Robotern wird oft das Verhalten von Insekten als Vorbild genommen. Eine beliebte Aufgabe für Selbstbauroboter ist, entlang einer Linie einen Weg zu gehen oder zu fahren. Ein kleines Scratch-Programm simuliert mit einem Käfer, wie so etwas funktioniert.

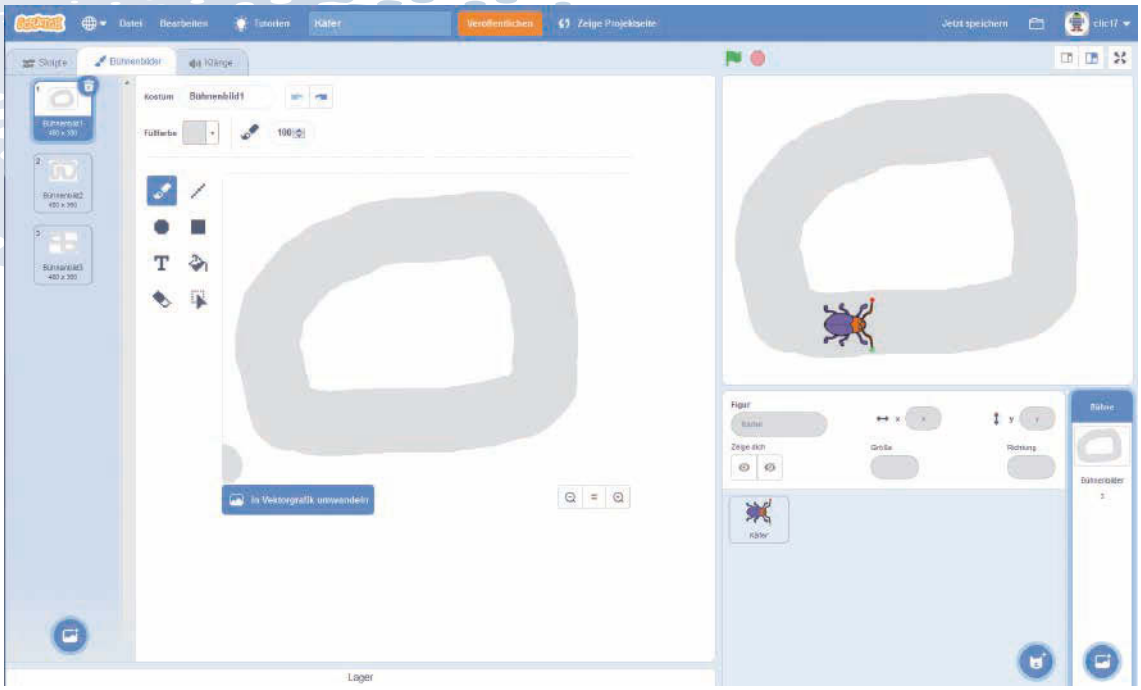
1 Starte ein neues Projekt in Scratch und lösche als Erstes die Katze.

2 Wir verwenden als Figur eine etwas abgewandelte Form des Käfers ***Beetle*** aus Scratch, den du als Datei ***Käfer.svg*** bei den Downloads findest. Klicke beim Symbol ***Figur wählen*** auf ***Figur hochladen*** und lade den Käfer in das Projekt.

3 Natürlich kannst du auch selbst einen Käfer malen. Achte dabei darauf, dass die Achse des Käfers genau auf dem Nullpunkt liegt. Außerdem braucht der Käfer links einen roten Fühler und rechts einen grünen. Mit diesen Fühlern erkennt er, wenn er vom Weg abweicht.

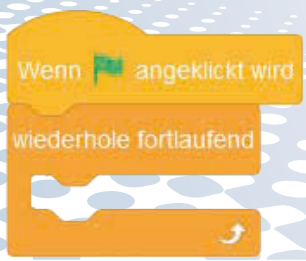


4 Schalte jetzt auf das Bühnenbild um, wähle eine graue Farbe und die breiteste Stiftdicke. Male damit einen geschlossenen Linienzug, auf dem sich der Käfer bewegen soll.



5 Setze den Käfer auf den Weg. Der Weg muss immer etwas breiter sein, als der Käfer zwischen den beiden Fühlern ist. Sollte dein Käfer zu breit sein, mache entweder den Weg breiter oder verkleinere den Käfer über den Faktor ***Größe*** in der Figurenpalette. ***100*** entspricht der Originalgröße, also 100 %.

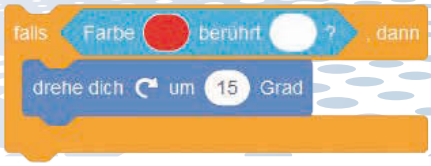
6 Erstelle für den Käfer ein Programm, das bei Klick auf das grüne Fähnchen automatisch eine ***wiederhole fortlaufend***-Schleife startet.



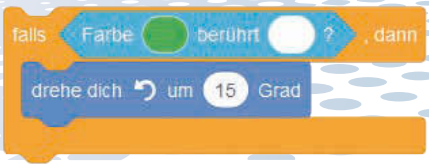
6 Ein Käfer sucht sich seinen Weg

7 In jedem Schleifendurchlauf geht der Käfer einen 5er-Schritt.

8 Anschließend wird überprüft, ob er vom Weg abgekommen ist. Berührt die rote Farbe des linken Fühlers den weißen Hintergrund, läuft der Käfer zu dicht am linken Wegesrand. Er muss sich um 15 Grad nach rechts drehen, um wieder in Richtung der Mitte des Wegs zu laufen.



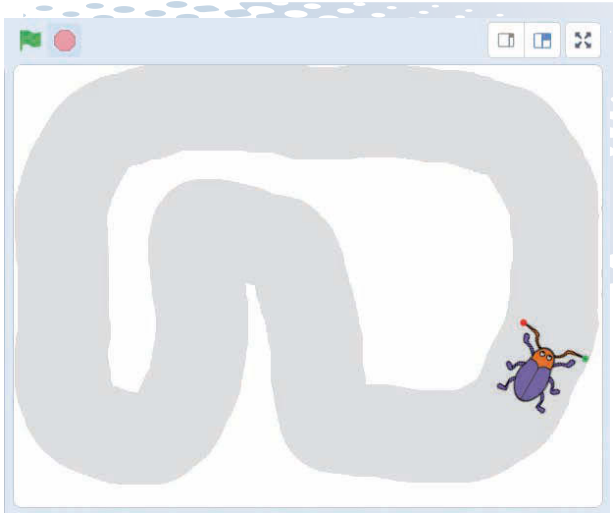
9 Berührt dagegen die grüne Farbe des rechten Fühlers den weißen Hintergrund, läuft der Käfer zu dicht am rechten Wegesrand. Er muss sich um 15 Grad nach links drehen, um wieder in Richtung der Mitte des Wegs zu laufen.



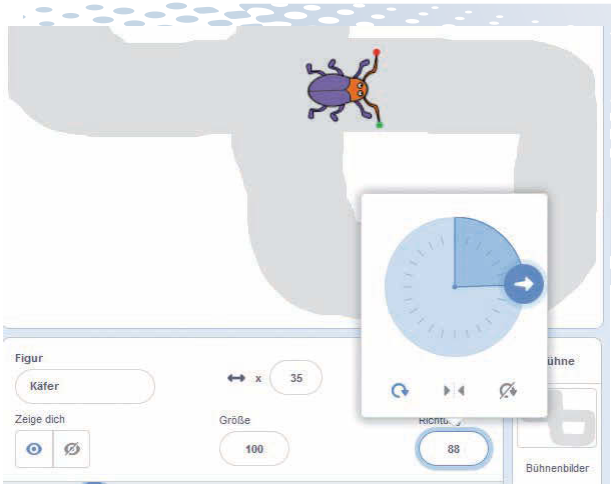
10 Danach startet die Endlosschleife wieder, und der Käfer geht den nächsten Schritt.

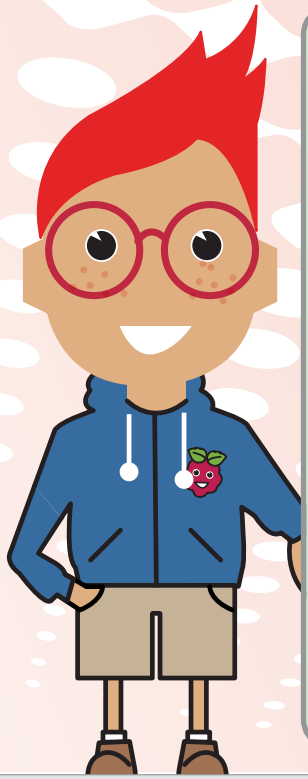
11 Setze den Käfer auf den Weg und klicke auf das grüne Fähnchen. Er läuft los und folgt dem Weg.

12 Jetzt kannst du verschiedene Bühnenbilder malen und mit dem Käfer ausprobieren. Je nachdem, an welche Stelle du den Käfer setzt, läuft er links herum oder rechts herum. Er versucht immer, nach rechts in Richtung 90 Grad loszulaufen.



13 Sogar Kreuzungen sind möglich, vorausgesetzt, der Käfer kann geradeaus über die Kreuzung laufen. Bei solchen Kreuzungen braucht er seine Fühler nicht einzusetzen und kann sich deshalb auch nicht „verlaufen“.





KÄFER EINFANGEN

Sollte der Käfer wirklich einmal vom Weg abkommen, was passieren kann, wenn dieser zu schmal oder zu dicht am Rand der Bühne liegt, sodass der Käfer keine weiße Fläche neben dem Weg mehr findet, kannst du ihn wieder einfangen:

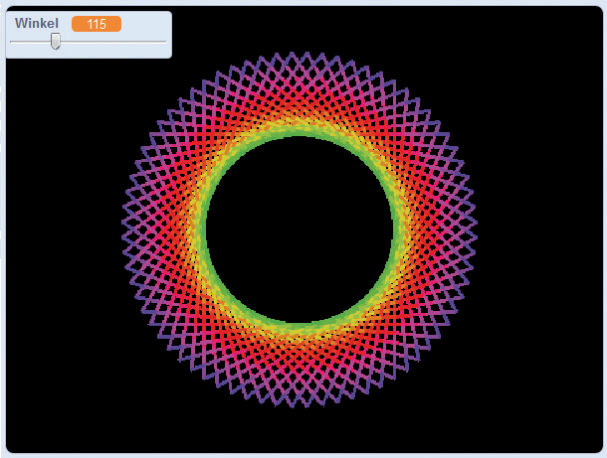
Klicke dazu zuerst auf das rote Stoppsymbol, um den Käfer anzuhalten.

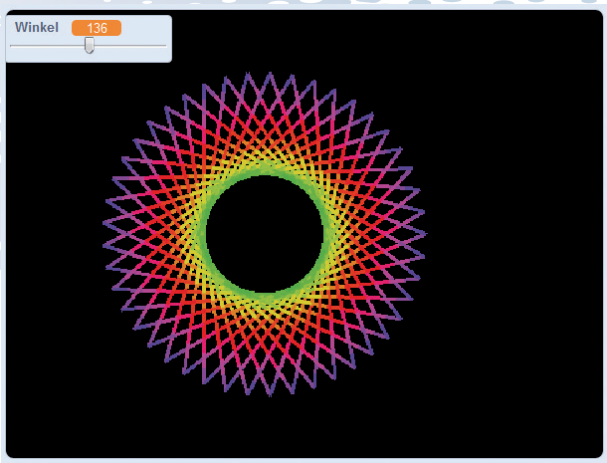
Trage dann in die beiden Felder *x* und *y* auf der Figurenpalette eine *0* ein, um den Käfer zurück in die Mitte der Bühne zu bringen. Von dort kannst du ihn leicht wieder auf den Weg setzen.

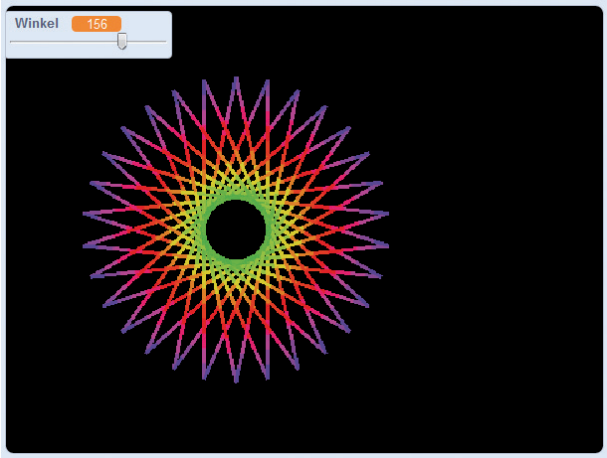
Klicke auf das Feld *Richtung*, um den Käfer zu drehen. Hier kannst du einen beliebigen Winkel einstellen, damit der Käfer wieder auf dem Weg startet.

7 Scratch malt Retro-Computergrafiken

In den 1970er-Jahren, in den Anfangszeiten der privat genutzten Computer, als die Grafikfähigkeiten noch ihre Grenzen hatten, waren kreisförmige Bilder, die aus vielen geraden Linien zusammengesetzt sind, groß in Mode.







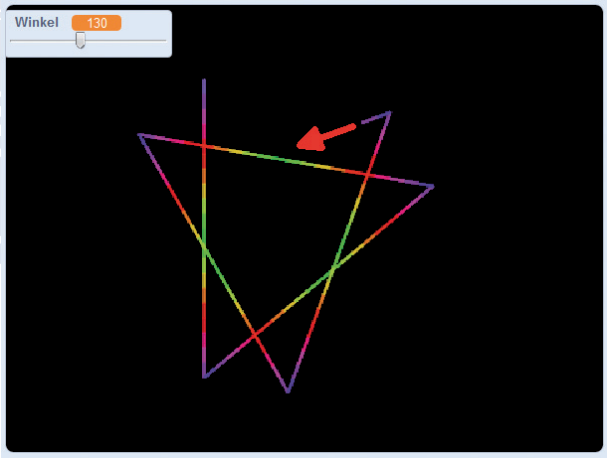
Mit diesem Scratch-Projekt kannst du solche Grafiken selbst erstellen. Scratch bietet Grafikfunktionen, die deutlich einfacher zu programmieren sind als die komplizierten Algorithmen der damaligen Zeit.

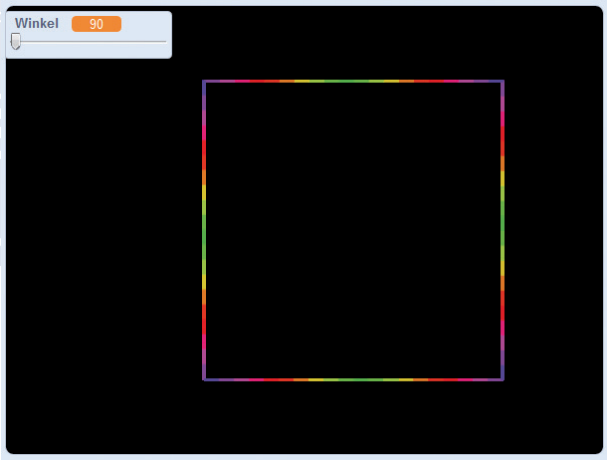
SO ENTSTEHEN DIE GRAFIKEN

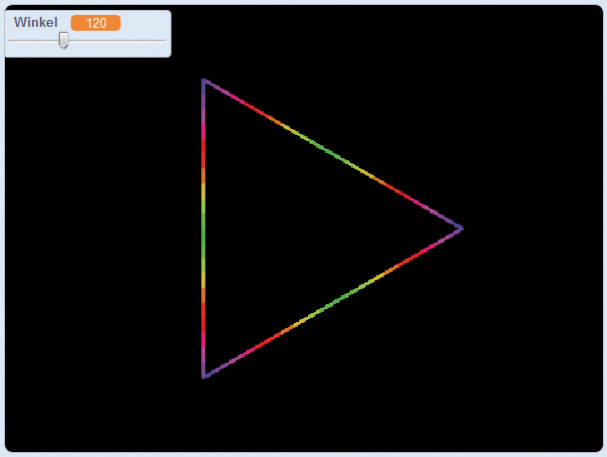
Beim Zeichnen solcher Grafiken werden immer die gleichen Schritte wiederholt:

1. Zeichne eine gerade Linie in einer bestimmten Länge.
2. Drehe dich um einen bestimmten Winkel.
3. Zeichne dann wieder eine gerade Linie der gleichen Länge.
4. Drehe dich wieder um den gleichen Winkel usw. usw.

Bei Drehwinkeln, die ein ganzzahliger Teiler von 360 Grad sind, was im Bereich zwischen 90 Grad und 180 Grad nur auf 90 Grad und 120 Grad zutrifft, entstehen einfache geometrische Figuren, da hier die vierte bzw. dritte Linie wieder am Anfangspunkt der ersten ankommt.





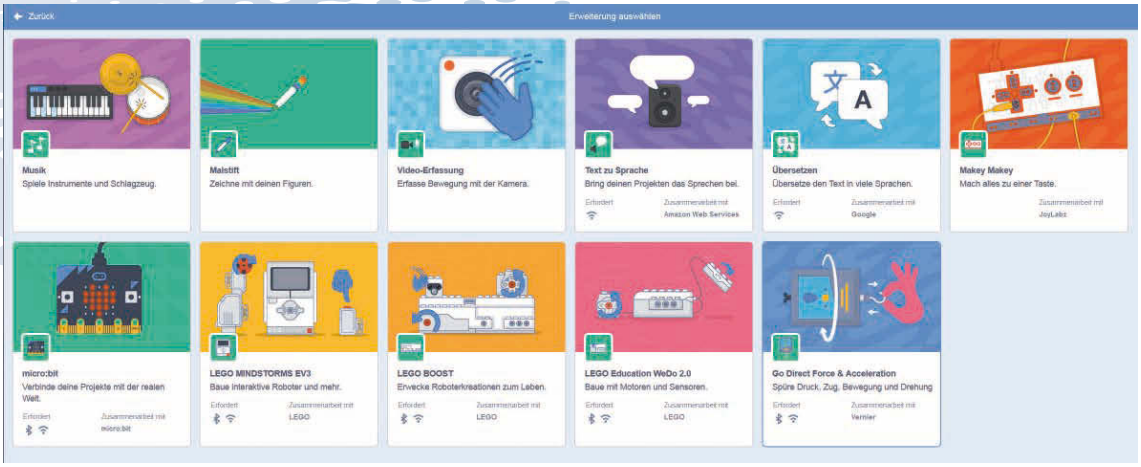


ERWEITERUNGEN INSTALLIEREN

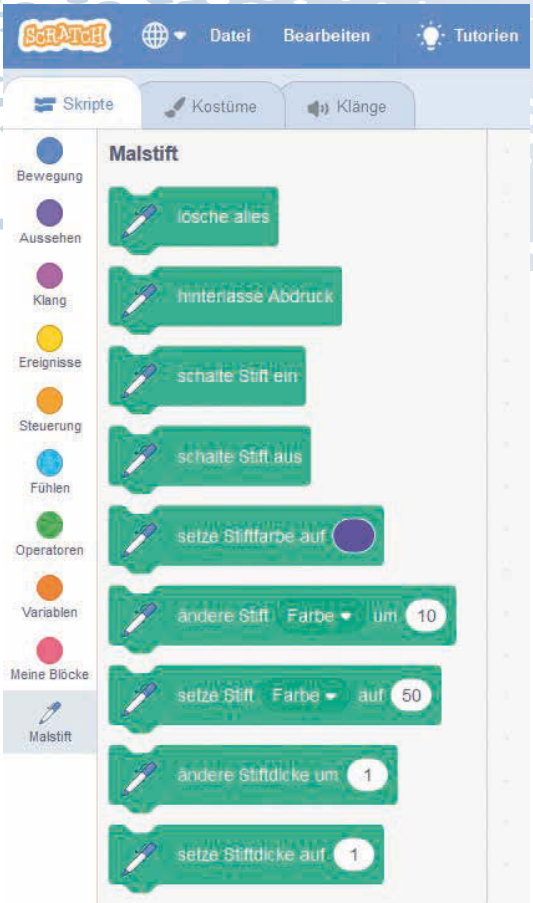
Mit Erweiterungen kannst du Scratch zusätzliche Funktionen geben. Scratch liefert bereits einige solche Erweiterungen mit, die mit einem Klick ganz einfach installiert werden können.

Klicke unten links auf das Symbol ***Erweiterung hinzufügen***. Es erscheint eine Liste aller verfügbaren Erweiterungen.

7 Scratch malt Retro-Computergrafiken



Installiere mit einem Klick die Erweiterung ***Malstift***. Du brauchst sie für das Grafikprogramm. Jede Erweiterung liefert zusätzliche Blöcke, die ganz unten auf der Blockpalette angehängt werden.



DAS ERSTE GRAFIKPROGRAMM

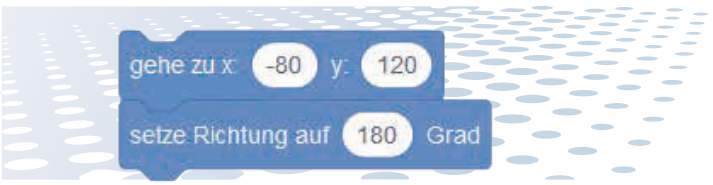
Die Scratch-Erweiterung ***Malstift*** verwendet die sogenannte Turtle-Grafik. Turtle ist das englische Wort für Schildkröte. Eine Schildkröte oder irgendeine andere Figur, z. B. die Scratch-Katze, bekommt einen farbigen Stift, mit dem jede ihrer Bewegungen auf dem Hintergrund eine Spur hinterlässt. Die Figur kann sich beim Malen auch verstecken, also unsichtbar werden, und trotzdem eine Spur hinterlassen. Die Scratch-Katze würde auch gar nicht ins Bild der Retro-Computergrafiken passen.

0 Färbe als Erstes den Hintergrund schwarz, dann erscheinen die farbigen Linien leuchtender, und das Ganze sieht stilechter aus. Klicke dazu in der Figurenliste auf die Bühne, schalte auf die Seite ***Bühnenbilder*** und wähle dort

die Füllfarbe Schwarz. Nimm dir dann den Farbeimer und klicke im Grafikbereich in das Zeichenfenster. Es färbt sich komplett schwarz.

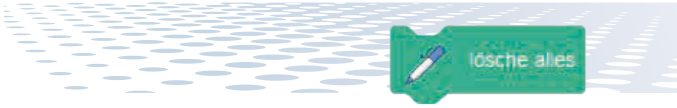
2 Als Allererstes nach dem Klicken auf das grüne Fähnchen soll sich die Katze verstecken. Wir brauchen sie zwar zum Zeichnen, sie soll aber nicht zu sehen sein. Auf der Blockpalette ***Aussehen*** gibt es dazu den Block ***verstecke dich***.

3 Die Grafik beginnt mit einer senkrechten Linie von oben nach unten. Wir beginnen an der Position ***x:-80, y:120***, und jede Linie ist 240 Einheiten lang. Zwei Bewegungsblöcke setzen die unsichtbare Katze an diese Position und die Richtung auf 180 Grad (nach unten).



4 Möchtest du sehen, wo sich eine versteckte Figur gerade befindet, klicke in der Figurenpalette auf das Auge bei ***Zeige dich***. Umgekehrt kannst du sichtbare Figuren mit dem durchgestrichenen Auge verstecken.

5 Bevor die unsichtbare Katze die erste Linie malt, sollen sämtliche Spuren früherer Zeichnungen weggewischt werden, damit die neue Grafik jedes Mal wieder auf einer leeren Bühne beginnen kann. Hänge dazu einen Block ***lösche alles*** von der Blockpalette ***Malstift*** aus der Erweiterung an das Programm an.

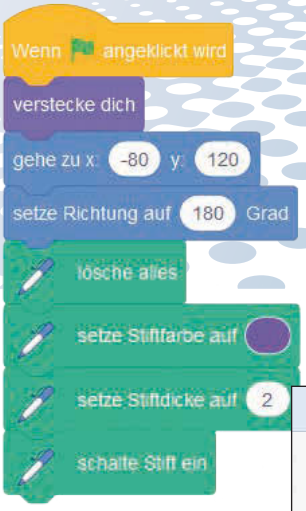


6 Setze die Stiftfarbe auf Violett. Sie wird später beim Malen noch verändert. Ziehe dazu einen Block ***setze Stiftfarbe auf ...*** an das Programm. Klicke in das Farbfeld dieses Blocks und wähle die Farbe Violett.

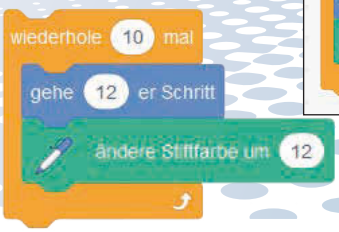
7 Setze jetzt mit einem Block ***setze Stiftdicke auf ...*** die Stiftdicke auf ***2***, damit die Linien klarer zu erkennen sind. Die Standardstiftdicke von ***1*** ist sehr dünn.

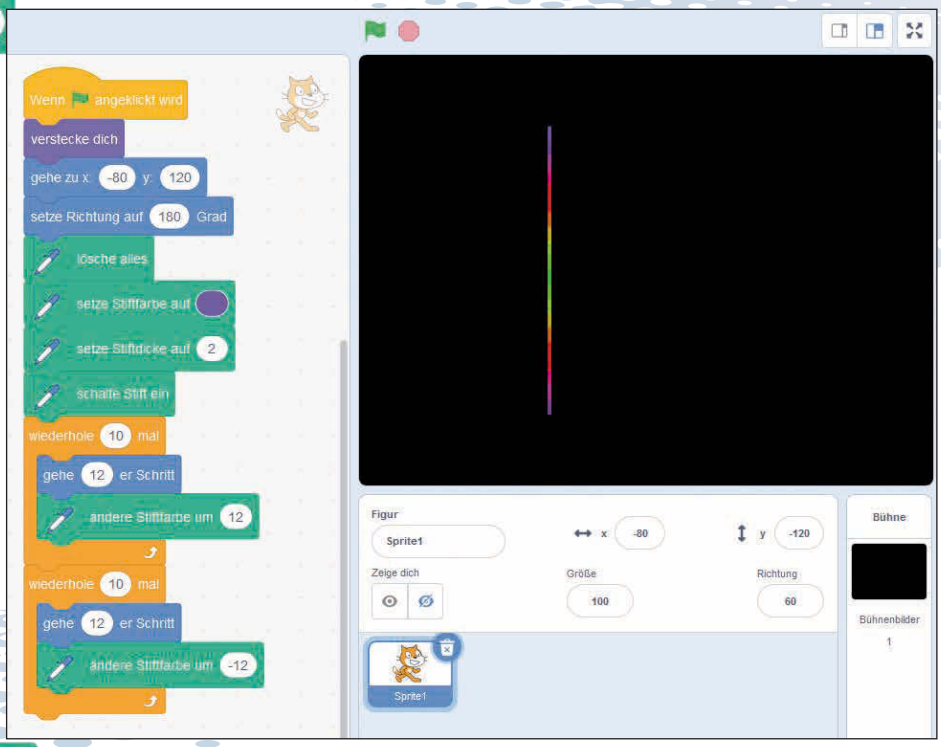
8 Jetzt muss der Stift mit dem Block ***schalte Stift ein*** noch eingeschaltet werden, damit die unsichtbare Katze bei ihrer Bewegung auch

wirklich malt. Mit dem Block ***schalte Stift aus*** kannst du den Stift jederzeit wieder ausschalten, um eine Linie zu unterbrechen.



9 Jetzt könntest du mit einer einfachen Bewegung der unsichtbaren Katze eine Linie malen. Da jede Linie aber einen Farbverlauf bekommen soll, der später die Grafiken so schön bunt macht, bauen wir zwei Schleifen ins Programm ein.

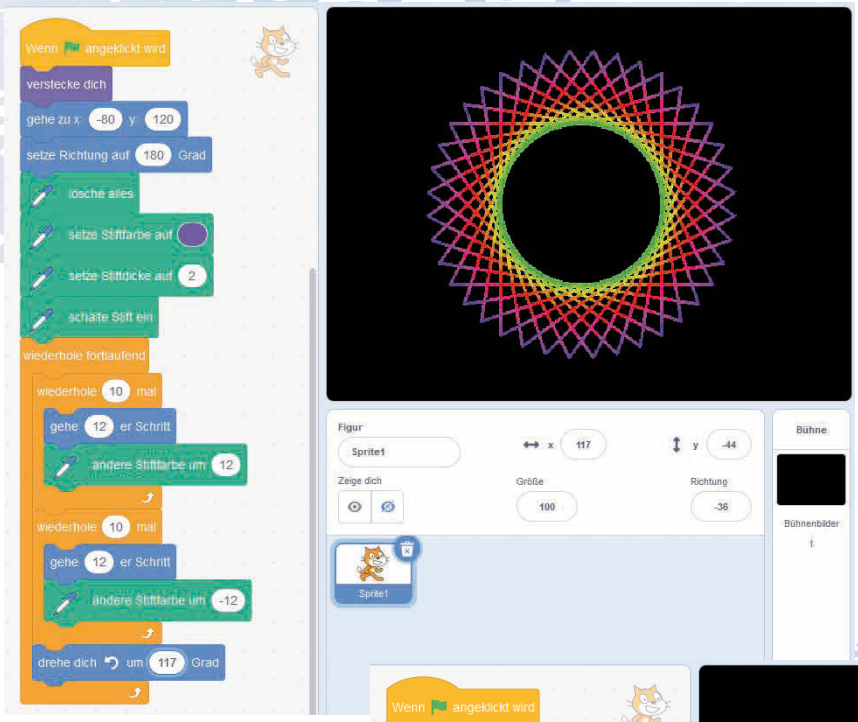




10 Diese Schleife malt ***10*** aufeinanderfolgende Linien, die jeweils ***12*** Einheiten lang sind. Nach jeder Linie wird die Stiftfarbe um ***12*** verändert. Über die Zahl im Block ***ändere Stiftfarbe um ...*** wird der Farbton entlang eines Regenbogens verändert. Je kleiner die Zahl, umso kleiner ist die Veränderung gegenüber der bisherigen Farbe. Nach 120 (= 10 * 12) Farbschritten wird aus dem Violett ein kräftiges Grün.

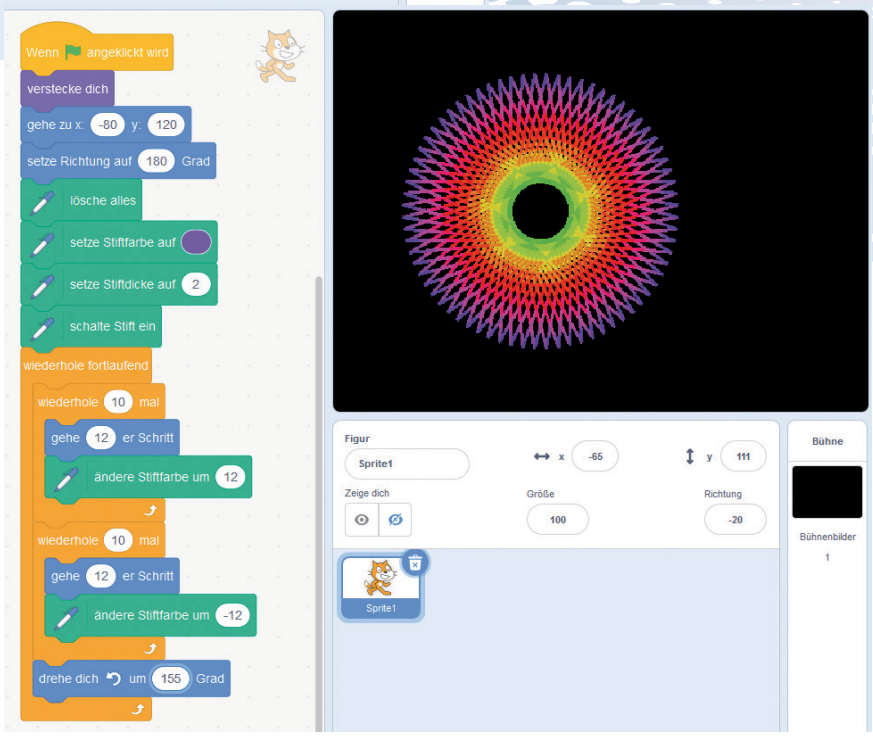
11 Die Linie hat jetzt eine Länge von 120 Einheiten, die Hälfte der für die Grafik gewünschten Länge. Eine zweite, ähnliche Schleife malt die zweite Hälfte der Linie, wobei sich die Farbe wieder Schritt für Schritt von Grün zurück nach Violett ändert.

12 Das Programm soll aber nicht nur eine Linie zeichnen, sondern eine runde Grafik aus vielen Linien. Dazu wird das Zeichnen einer Linie



über eine Endlosschleife wiederholt. Jede Linie beginnt am Endpunkt der zuletzt gezeichneten, da sich die unsichtbare Katze zwischendurch nicht bewegt. Die Katze dreht sich nur jedes Mal am Ende einer Linie um den gleichen Winkel.

13 Probiere unterschiedliche Winkel im Block ***drehe dich um ... Grad*** aus. Du wirst sehen, es entstehen völlig verschiedene Muster. Je größer der Winkel, desto kleiner wird der schwarze Kreis in der Mitte der Grafik.



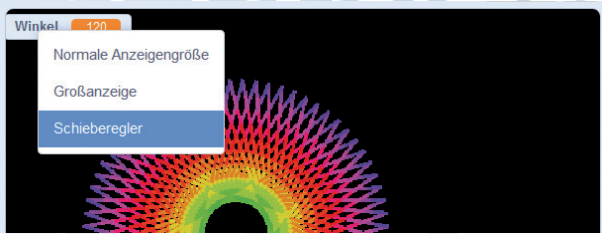
WINKEL ÜBER VARIABLE EINSTELLEN

In einem echten Programm wird man keinem Anwender zumuten, im Programmcode irgendwelche Werte zu verändern. Viel eleganter ist es, solche veränderbaren Werte, wie hier den Winkel, direkt auf der Benutzeroberfläche einzustellen.

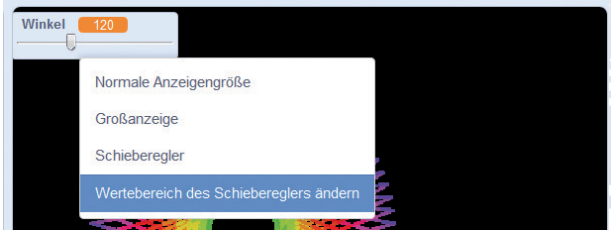
1 Lege auf der Blockpalette ***Variablen*** eine neue Variable ***Winkel*** an, die den Drehwinkel des Malstifts zwischen zwei Linien festlegt. Diese Variable soll auf der Bühne zu sehen sein. Das Kästchen links neben der Variablen auf der Blockpalette muss angekreuzt sein.

2 Ziehe die Variable ***Winkel*** in das Zahlenfeld im ***drehe dich um ... Grad***-Block.

3 Klicke mit der rechten Maustaste auf das Anzeigefeld der Variablen auf der Bühne und wähle im Menü ***Schieberegler***. Damit erstellst du einen Schieberegler, um den Wert der Variablen, während das Programm läuft, mit der Maus zu ändern.

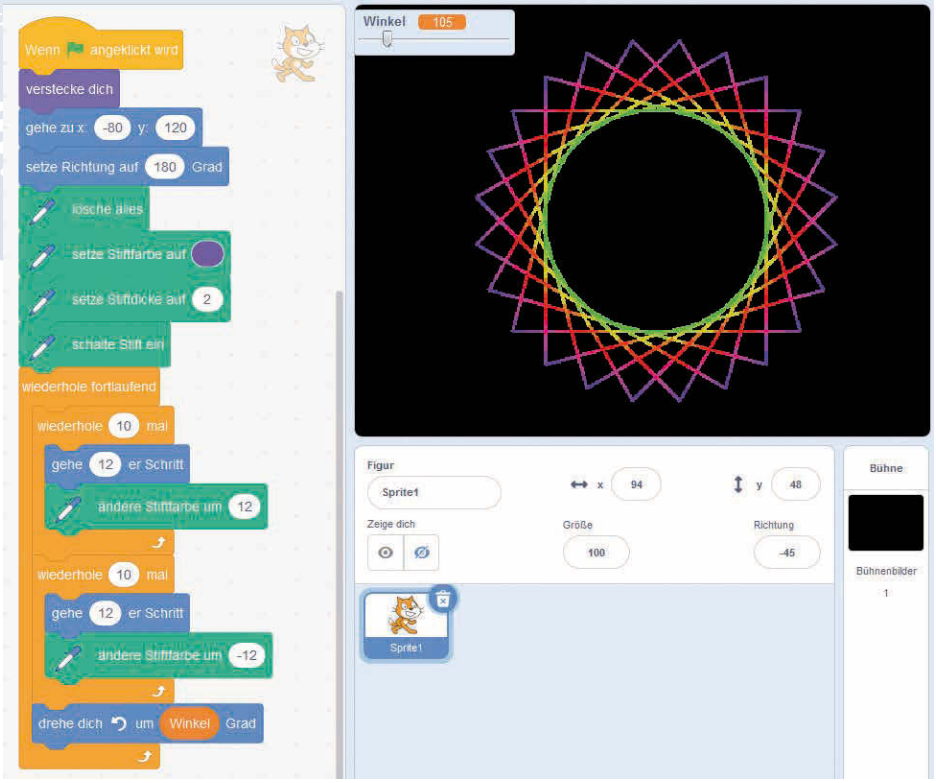


4 Klicke noch einmal mit der rechten Maustaste auf das Anzeigefeld der Variablen und wähle jetzt ***Wertebereich des Schiebereglers ändern***. Stelle hier einen Bereich von ***90*** bis ***180*** ein. Das bedeutet, dass du später in diesem Bereich mit dem Schieberegler einen Wert einstellen kannst. Andere Werte ergeben bei dieser Art von Grafik kein sinnvolles Ergebnis.



5 Stelle mit dem Schieberegler einen Wert ein und starte das Programm mit einem Klick auf das grüne Fähnchen.

6 Um einen anderen Wert auszuprobieren, halte das Programm mit einem Klick auf das rote Stoppsymbol an, stelle den neuen Wert ein und starte es mit dem grünen Fähnchen wieder. Beim Neustart des Programms werden alle Malspuren weggewischt, damit die neue Grafik auf der schwarzen Bühne gut zu erkennen ist.



0 Lege auf der Blockpalette ***Variablen*** eine neue Variable ***Winkel_alt*** an, die sich den Drehwinkel des Malstifts merkt. Diese Variable soll auf der Bühne nicht zu sehen sein. Das Kästchen links neben der Variablen auf der Blockpalette darf also nicht angekreuzt sein.

WINKEL INTERAKTIV EINSTELLEN

Jedes Mal erst das Programm anhalten zu müssen, um einen anderen Winkel auszuprobieren, ist nicht besonders komfortabel. Ein Benutzer, der mit der Technik des Programms nicht vertraut ist, wird auch nicht verstehen, warum er das tun sollte. Deutlich benutzerfreundlicher ist es, wenn das Programm automatisch richtig reagieren würde, wenn man einen anderen Winkel einstellt.

2 Diese Variable wird am Anfang des Programms auf ***0*** gesetzt.

3 Innerhalb der Endlosschleife, die die Grafik erzeugt, steht jetzt eine ***falls ..., dann ... sonst***-Abfrage, die prüft, ob der aktuell über den Schieberegler eingestellte Winkel gleich dem zuletzt in der Variablen ***Winkel_alt*** gemerkten Winkel ist.

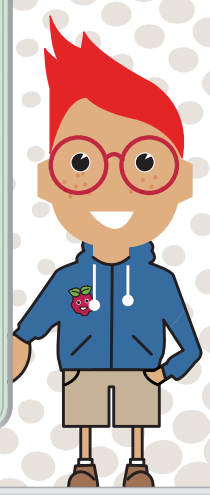
4 Sind beide Winkel gleich, malen die bereits bekannten Schleifen eine insgesamt 240 Einheiten lange Linie mit Farbverlauf. Danach dreht sich die unsichtbare Katze um den eingestellten Winkel, und die nächsten Linien werden gezeichnet. Bis hier läuft alles wie bereits bekannt.

5 Ist der aktuell über den Schieberegler eingestellte Winkel aber nicht gleich dem zuletzt in der Variablen ***Winkel_alt*** gemerkten Winkel, hat der Benutzer des Programms über den Schieberegler einen neuen Winkel eingestellt. In diesem Fall wird der zweite Block in der ***falls ..., dann ... sonst***-Abfrage ausgeführt.

Hier passiert all das, was in der letzten Programmversion noch am Anfang erledigt wurde, um eine neue Grafik zu starten. Dabei wird die unsichtbare Katze an den Anfangspunkt gesetzt, alle Malspuren werden gelöscht, Stiftfarbe und Stiftdicke werden eingestellt. Danach wird die Variable ***Winkel_alt*** auf den aktuellen Winkel gesetzt, und im nächsten Durchlauf der Endlosschleife wird mit dem Malen begonnen.

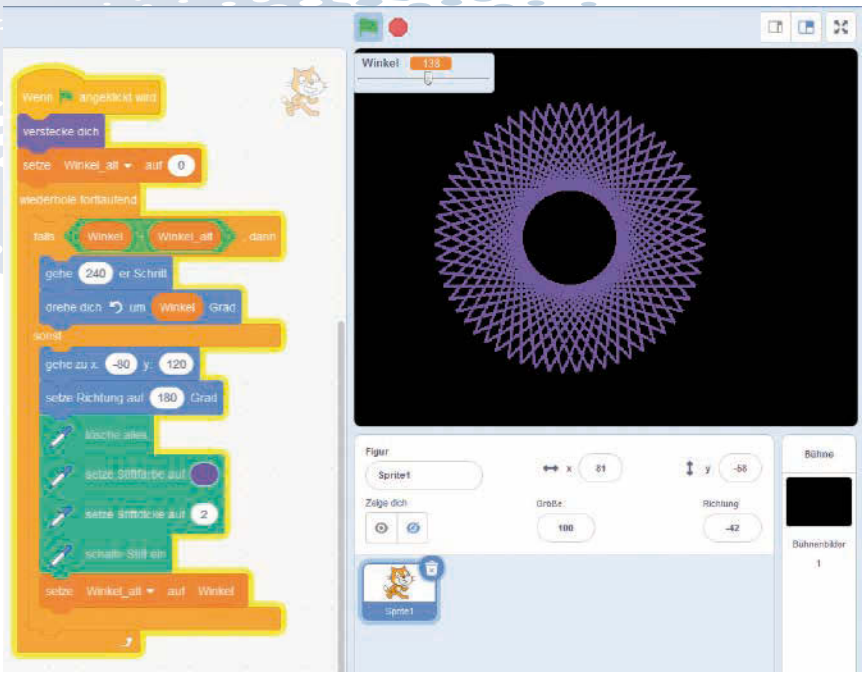
INITIALISIERUNG AM ANFANG FÄLLT WEG

Die Initialisierung der Grafik am Programmanfang ist jetzt nicht mehr nötig. Da die Variable ***Winkel_alt*** beim Start auf ***0*** gesetzt wird, kann sie nie den gleichen Wert wie der über den Schieberegler eingestellte Winkel haben. Also wird gleich beim ersten Durchlauf der Endlosschleife der zweite Block in der ***falls ..., dann ... sonst***-Abfrage ausgeführt, der alle Einstellungen der Grafik für ein neues Bild zurücksetzt.



SCHNELLERE GRAFIK

Es ist etwas mühsam auszuprobieren, wie sich verschiedene Winkel auf die Grafik auswirken, da der Aufbau eines neuen Bilds immer einige Zeit in Anspruch nimmt. Grund ist der Farbverlauf auf den Linien. Generell gilt für jedes Grafikprogramm: Es dauert immer gleich lang, eine Linie zu zeichnen, egal wie lang sie ist. Das Programm



zeichnet aber zurzeit 20 aufeinanderfolgende Linienstücke in unterschiedlichen Farben, die dann wie eine Linie aussehen.

Wenn du also auf den Farbverlauf verzichtest und statt 20 Linien jedes Mal nur eine zeichnest, ist das Programm etwa 20-mal so schnell.

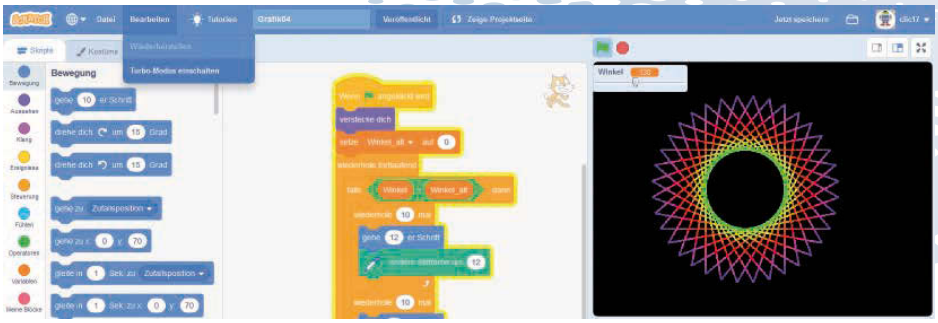
Ersetze die Schleifen im ersten Block in der ***falls ..., dann ... sonst***-Abfrage durch eine einfache Bewegung um 240 Einheiten. Dann kannst du wirklich interaktiv mit der Winkeleinstellung spielen und schnell sehen, welche Winkel interessante Grafiken ergeben.

DER TURBO-MODUS

So weit zur Programmiertheorie ... In Wirklichkeit ist jeder PC heute wesentlich schneller, als nötig ist, um ein paar Linien ohne sichtbaren Zeitverlust zu malen. Scratch hat extra Bremsen eingebaut, damit es dir beim Programmieren nicht davonläuft und du die Programmschritte mitverfolgen kannst.

Wenn du im Menü unter ***Bearbeiten*** den ***Turbo-Modus*** einschaltest, wird diese Bremse gelöst, und alles läuft richtig schnell.

Jetzt kannst du wirklich interaktiv mit dem Schieberegler verschiedene Winkel ausprobieren und dabei zusehen, wie sich die Grafik in voller Farbenpracht verändert.



8 Musik mit Scratch

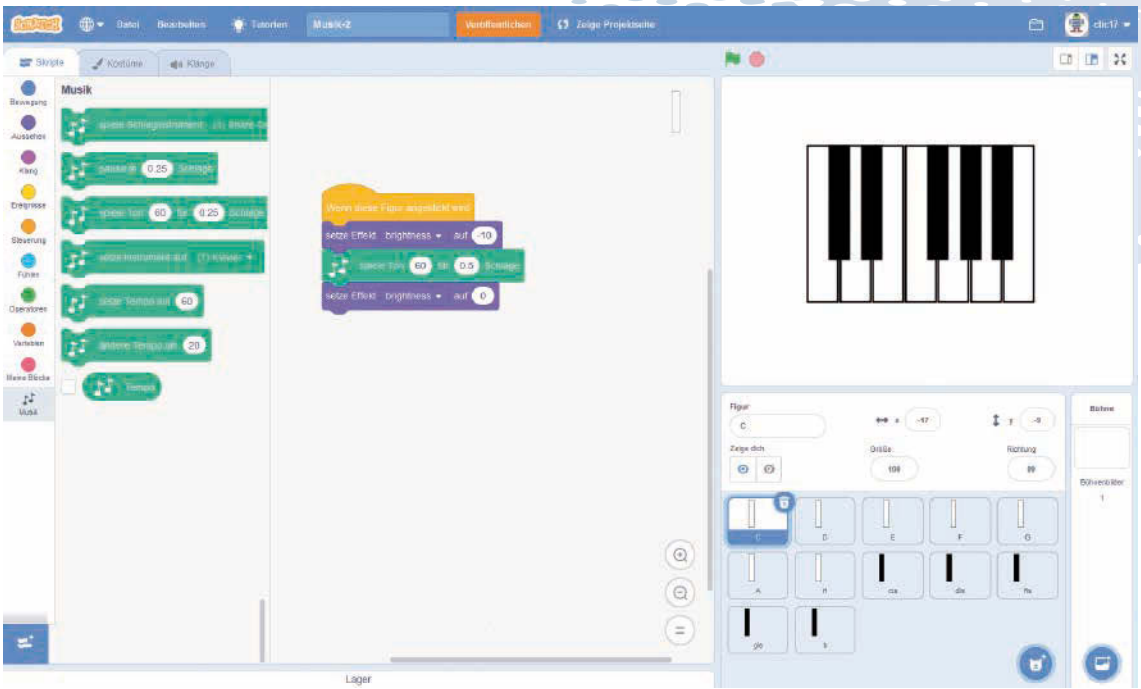
Mit einer zusätzlichen Scratch-Erweiterung kannst du auch Musik machen.

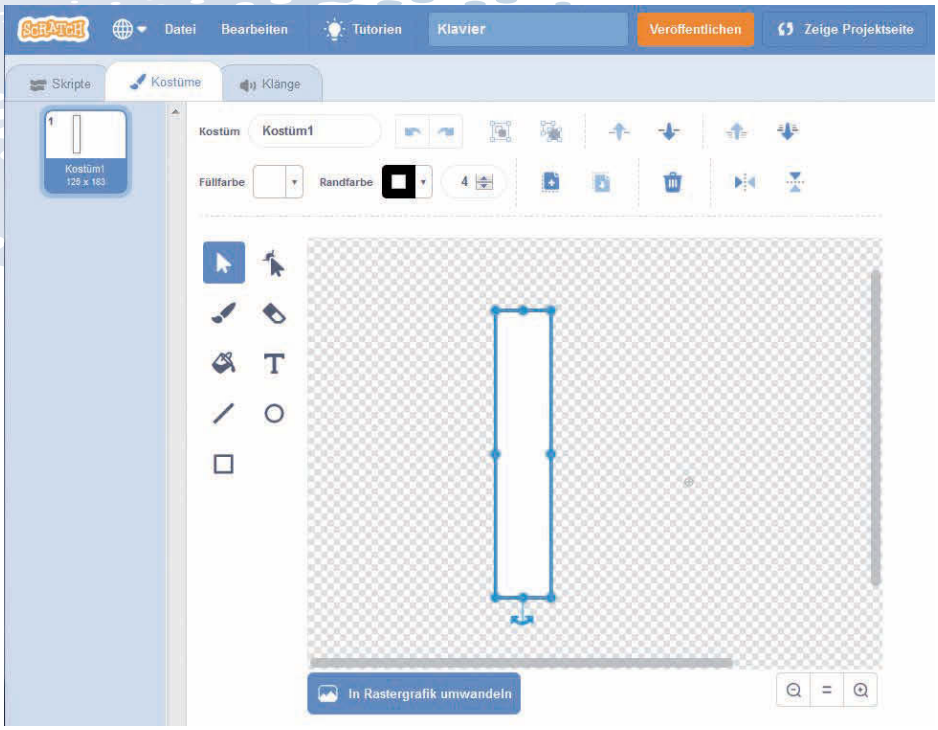
Klicke unten links auf das Symbol ***Erweiterung hinzufügen***. Installiere die Erweiterung ***Musik***.

In diesem Projekt zeichnen wir ein kleines Klavier, das Töne von sich gibt.

1 Starte ein neues Projekt in Scratch und lösche als Erstes die Katze. In diesem Projekt wirst du zahlreiche andere Figuren benötigen, genauer gesagt, für jede Klaviertaste eine Figur, die Katze brauchst du aber nicht.

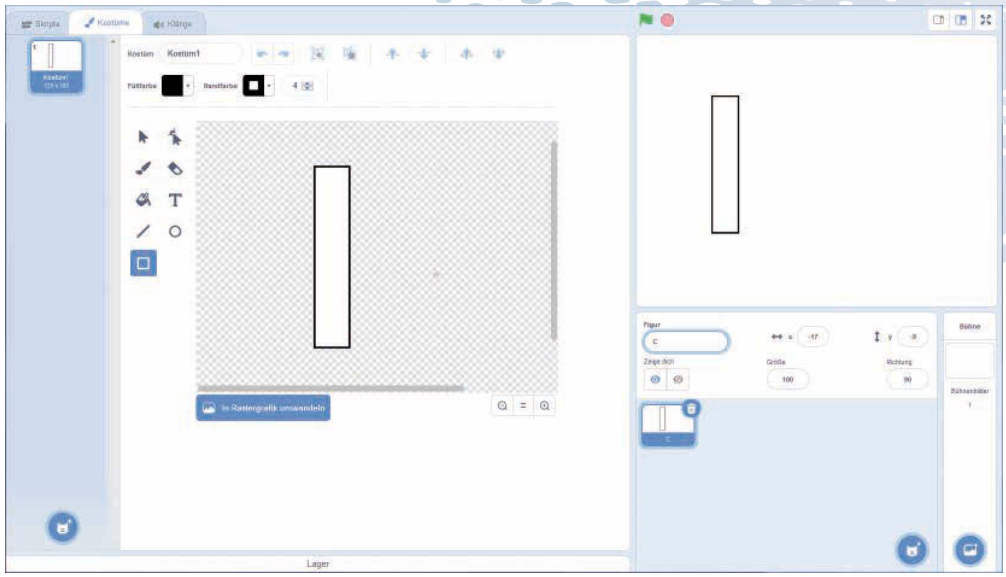
2 Zeichne jetzt die erste weiße Klaviertaste. Fahre dazu in der Figurenpalette mit der Maus auf ***Figur wählen*** und klicke auf ***Malen***. Wähle als Füllfarbe Weiß und als Randfarbe Schwarz. Zeichne dann ein schmales Rechteck.

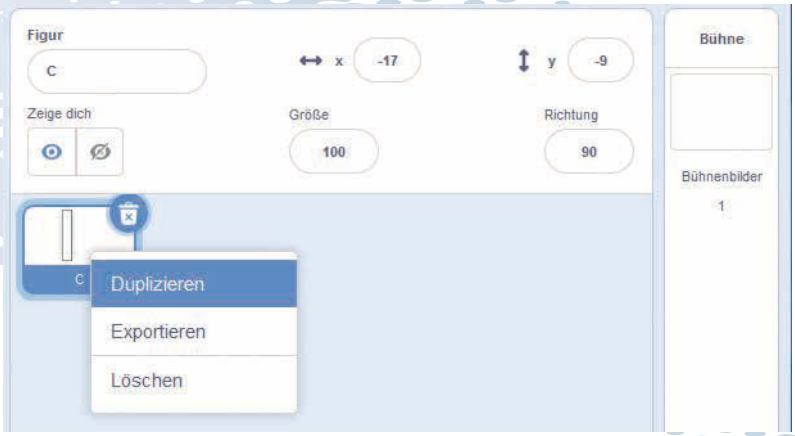




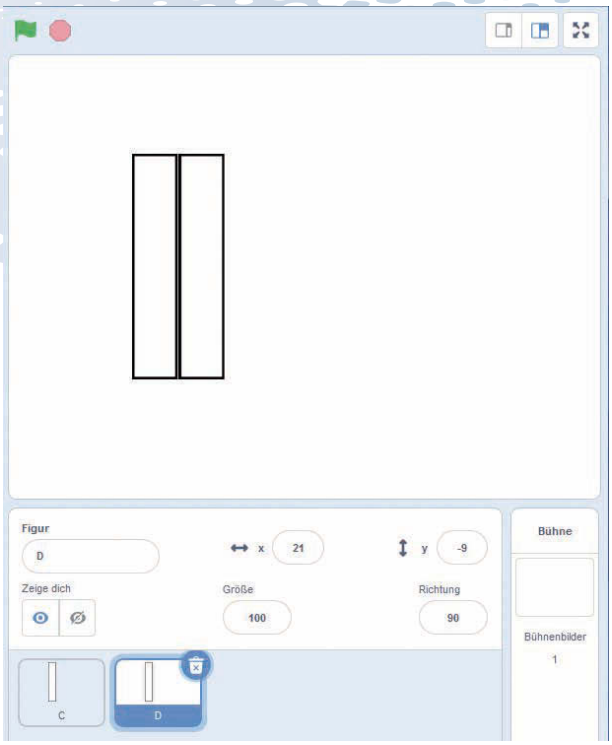
9 Unser Klavier soll sieben weiße und fünf schwarze Tasten haben. Die nächsten sechs weißen Tasten kannst du einfach aus der ersten Taste duplizieren. Klicke dazu mit der rechten Maustaste auf die Figur ***C*** in der Figurenpalette und wähle im Menü ***Duplizieren***.

8 Gib der Taste gleich noch einen Namen. Bei vielen Figuren in einem Programm kommt man sonst schnell durcheinander. Ändere in der Figurenpalette im Namensfeld den Namen von ***Figur1*** auf ***C***. Das ist der Ton, den die erste Taste spielen soll.

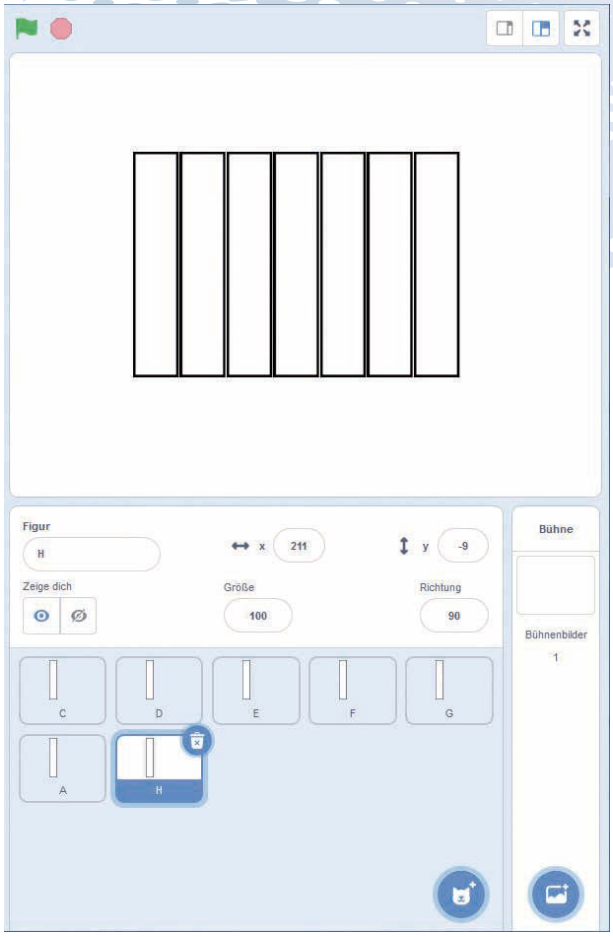




5 Benenne die duplizierte Taste in ***D*** um und schiebe sie mit winzigem Abstand rechts neben die erste Taste.



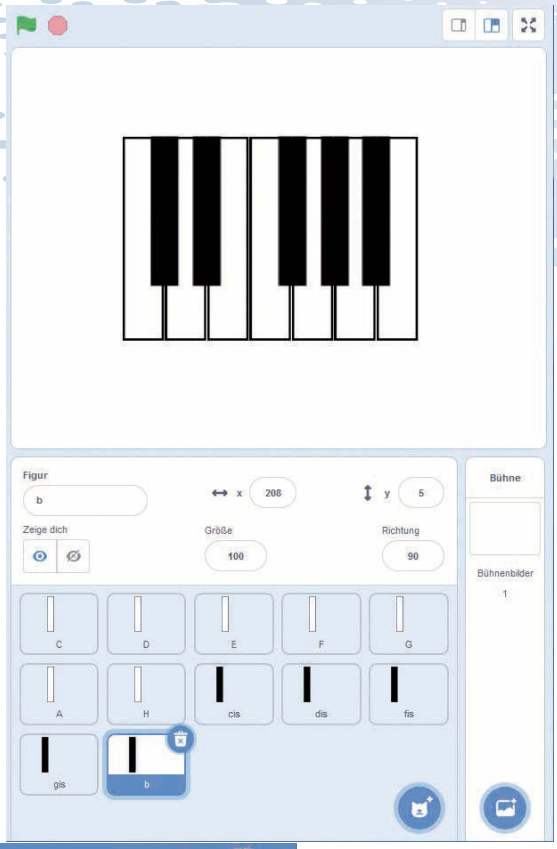
6 Dupliziere auf die gleiche Weise weitere Tasten und nenne sie ***E***, ***F***, ***G***, ***A*** und ***H***.

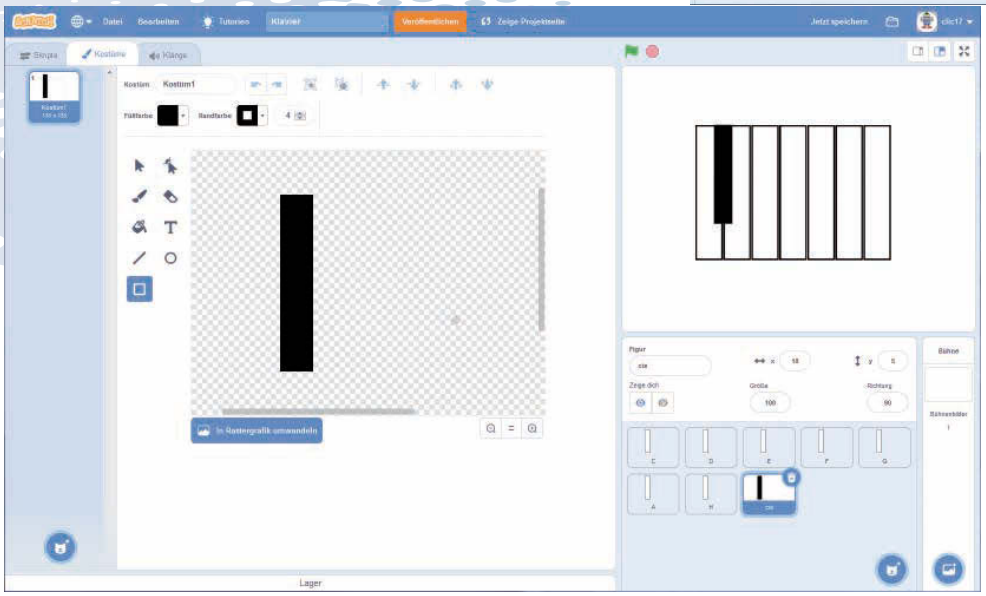


7 Sollte eine Taste beim Duplizieren außerhalb der Bühne landen, kannst du sie nicht greifen, um sie an die richtige Position zu schieben. Trage in diesem Fall auf der Figurenpalette in den Feldern ***x*** und ***y*** zweimal die ***0*** ein. Die aktuell ausgewählte Taste rutscht in den sichtbaren Bereich der Bühne.

8 Zeichne jetzt die erste schwarze Taste als ausgefülltes Rechteck und etwas kleiner als die weißen Tasten. Schiebe diese Taste zwischen die ersten beiden weißen Tasten, sodass die oberen Kanten miteinander abschließen.

9 Benenne diese Figur in ***cis*** um und dupliziere daraus vier weitere schwarze Tasten. Baue diese wie abgebildet zwischen den weißen Tasten ein und nenne sie ***dis***, ***fis***, ***gis*** und ***b***.





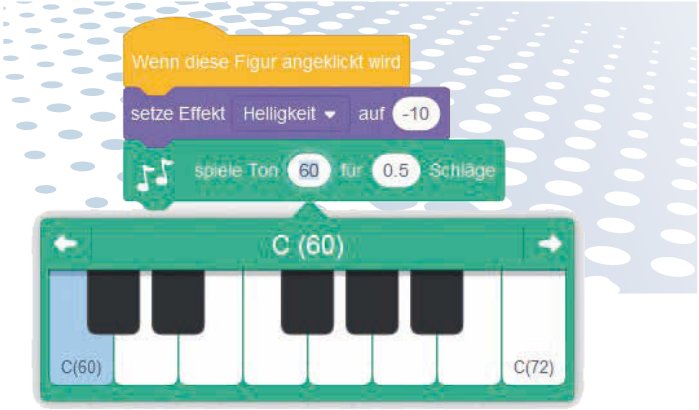
8 Musik mit Scratch

10 Jede Taste bekommt jetzt ihre eigenen Programmblöcke. Diese sind für alle Tasten ähnlich und spielen beim Anklicken der Taste einen bestimmten Ton ab. Die Programmblöcke starten diesmal nicht bei Klick auf das grüne Fähnchen, sondern jedes Mal, wenn die entsprechende Taste angeklickt wird. Markiere in der Figurenliste die Taste ***C*** und ziehe aus der Blockpalette ***Ereignisse*** den Block ***Wenn diese Figur angeklickt wird*** ins Skriptfenster.

11 Alle Blöcke darunter werden immer dann ausgeführt, wenn die Figur – hier die Taste ***C*** – angeklickt wird. Damit man sieht, dass eine Klaviertaste gedrückt wurde, soll sie kurz etwas abgedunkelt erscheinen. Hänge dazu aus der Blockpalette ***Aussehen*** den Block ***setze Effekt ... auf ...*** an das Programm, wähle im Listenfeld ***Helligkeit*** und trage in das Zahlenfeld ***-10*** ein. Damit verdunkelt sich die Figur.

12 Danach soll Scratch den richtigen Ton erzeugen. Ziehe dazu aus der Erweiterung ***Musik*** einen Block ***spiele Ton... für... Schläge*** in das Programm.

13 Klicke einfach mal auf diesen Block im Programmfenster – der erste Ton erklingt. Klicke dann in das Zahlenfeld hinter ***Ton***. Es erscheint eine Klaviatur, auf der du den gewünschten Ton auswählen kannst. Die erste Klaviertaste ***C*** soll den Ton ***C(60)*** spielen.

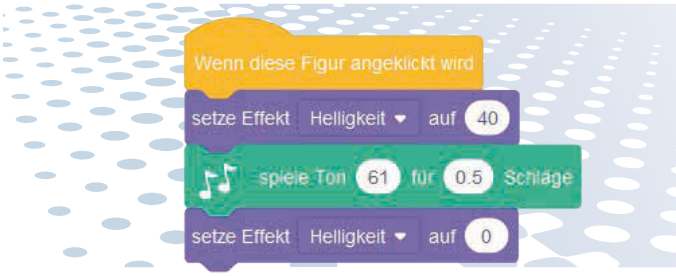


14 Wurde der Ton abgespielt, soll die Taste wieder normal und nicht abgedunkelt dargestellt werden. Hänge dazu einen weiteren Block ***setze Effekt Helligkeit auf ...*** an das Programm und verwende dort wieder den Wert ***0***.

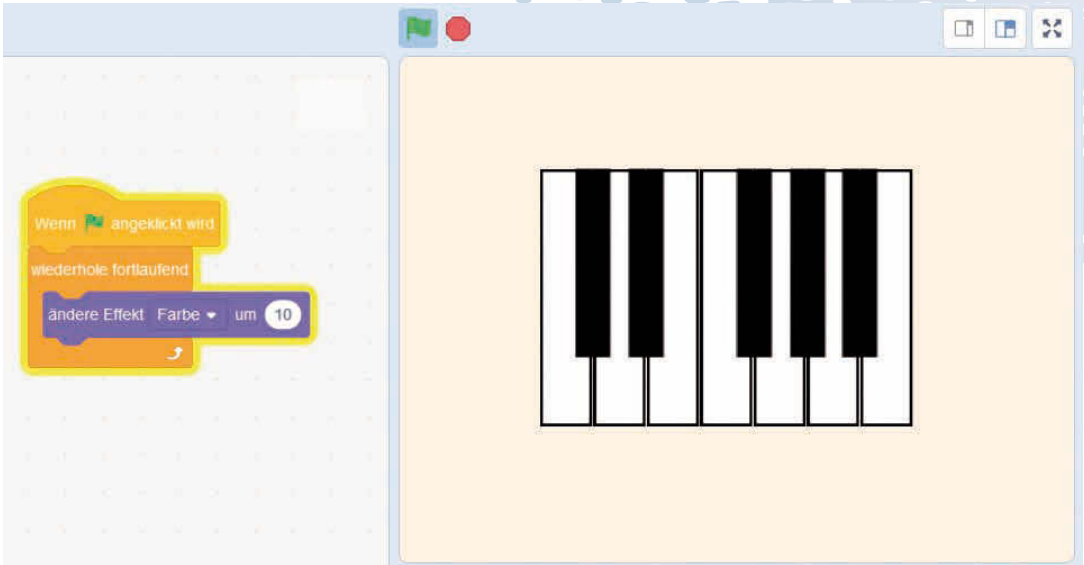
15 Lege die gleichen Blöcke für alle weißen Klaviertasten an. Welche Taste welchen Ton spielen soll, ist in der Klaviertastenabbildung des Blocks ***spiele Ton ... für ... Schläge*** gut zu erkennen. Die Bezeichnungen der Töne lauten in Scratch manchmal etwas anders als im deutschen Sprachgebrauch.

Ton	Tastenfarbe	Nummer	Scratch
c	Weiß	60	C
cis	Schwarz	61	C#
d	Weiß	62	D
dis	Schwarz	63	Eb
e	Weiß	64	E
f	Weiß	65	F
fis	Schwarz	66	F#
g	Weiß	67	G
gis	Schwarz	68	G#
a	Weiß	69	A
b	Schwarz	70	Bb
h	Weiß	71	B
c	Weiß	72	C

16 Die schwarzen Klaviertasten müssen beim Anklicken etwas heller und nicht dunkler werden, damit sie gut zu erkennen sind. Setze hier die Helligkeit auf ***40*** statt auf ***-10***.

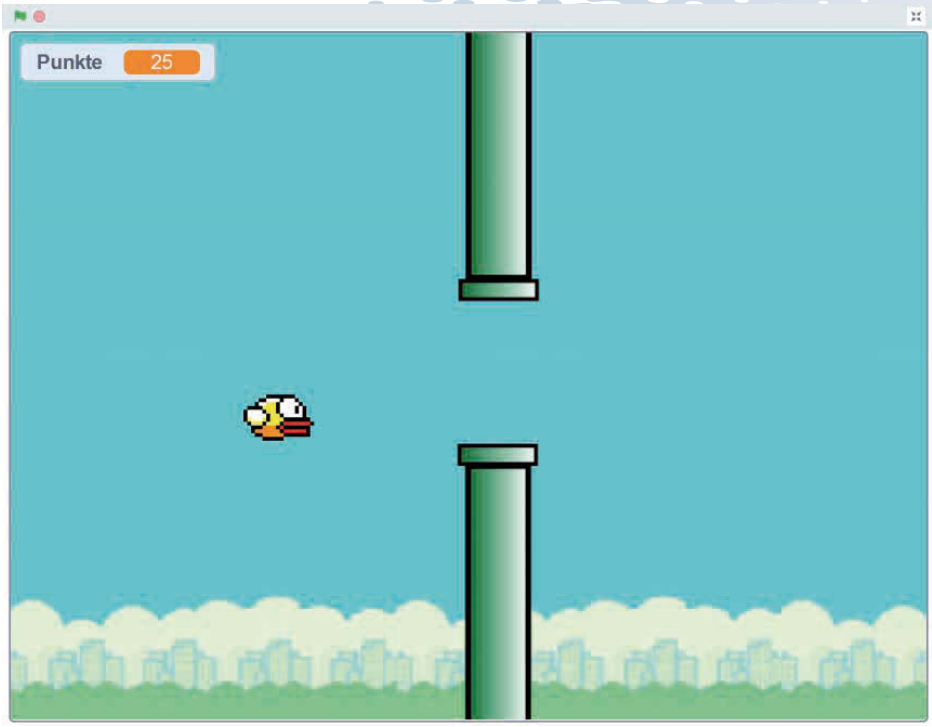


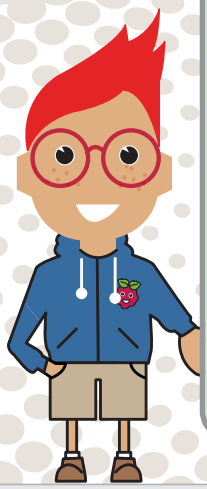
17 Damit der Hintergrund beim Musizieren nicht so langweilig weiß aussieht, lege einen kleinen Farbeffekt auf die Bühne. Markiere dazu in der Figurenliste rechts außen die Bühne. Auch hier kannst du im Skriptfenster Programmblöcke einfügen, das funktioniert nicht nur bei Figuren. Das abgebildete Programm verändert ständig die Farbe des Bühnenbilds.



9 Flappy Bird

Flappy Bird war das Kultspiel des Sommers 2013. Leider hat der Entwickler am 9. Februar 2014 das Spiel kommentarlos aus allen App-Stores entfernt. Seitdem haben zahlreiche Entwickler das Spiel für verschiedenste Computerplattformen, Smartphones und Spielkonsolen nachgebaut. Wir werden jetzt eine etwas vereinfachte Version des Spiels in Scratch nachbauen.



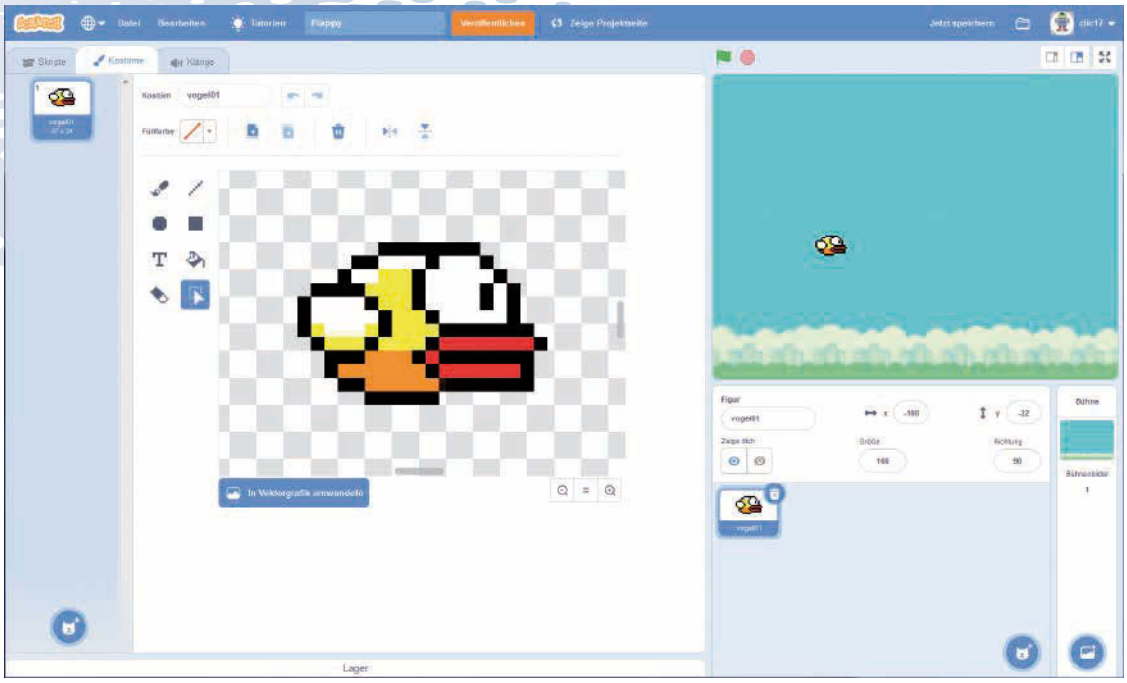


DIE SPIELREGELN

Der kleine Vogel Flappy Bird muss zwischen Rohren hindurchfliegen, die ihm zufällig entgegenkommen. Dabei wird er mit einer Taste in den Himmel aufsteigen und dann langsam wieder heruntersegeln. Jedes Mal, wenn er zwischen zwei Rohren hindurchfliegt, gibt es einen Punkt. Stößt er jedoch an ein Rohr, ist das Spiel zu Ende.

1 Lege in Scratch ein neues Projekt an, klicke in der Figurenliste auf die Bühne und dann auf das Symbol ***Bühnenbild hochladen***. Lade das Hintergrundbild ***flappy-bird-buehne.png***, das dem Hintergrund des Originalspiels sehr ähnlich sieht.

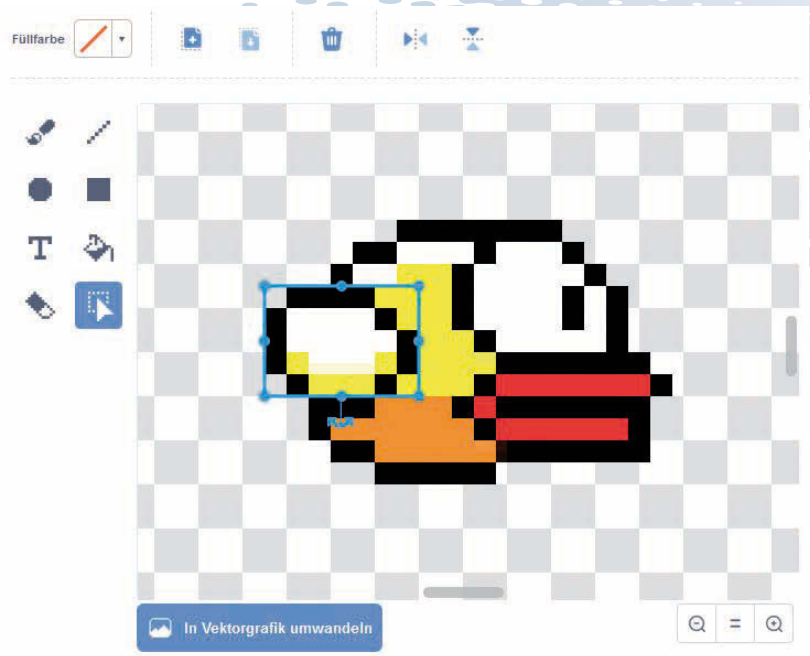
2 Lösche die Katze und klicke in der Figurenliste auf das Symbol ***Figur hochladen***. Lade den Vogel ***vogel01.png*** aus den Downloads zu diesem Buch.



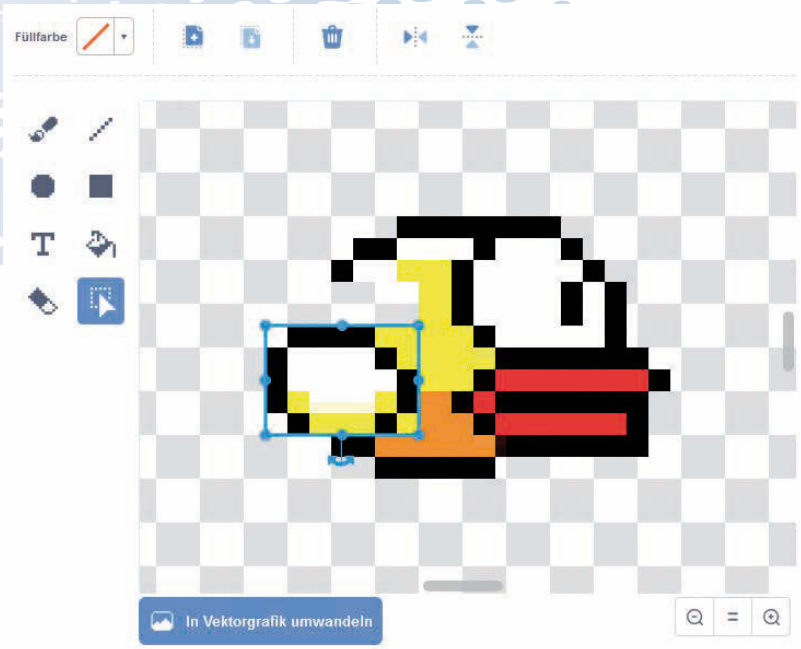
DER VOGEL FLIEGT

1 Damit der Vogel beim Fliegen mit den Flügeln schlagen kann, bekommt er noch ein zweites Kostüm. Klicke dazu mit der rechten Maustaste auf das Kostüm und wähle im Menü ***Duplizieren***.

2 Wähle mit dem ***Auswählen***-Werkzeug in diesem Kostüm den Flügel aus und schiebe ihn um zwei Rastereinheiten nach unten. Die Figur ist als Rastergrafik importiert. Deshalb gibt es etwas andere Bearbeitungswerkzeuge als bei Vektorgrafiken.

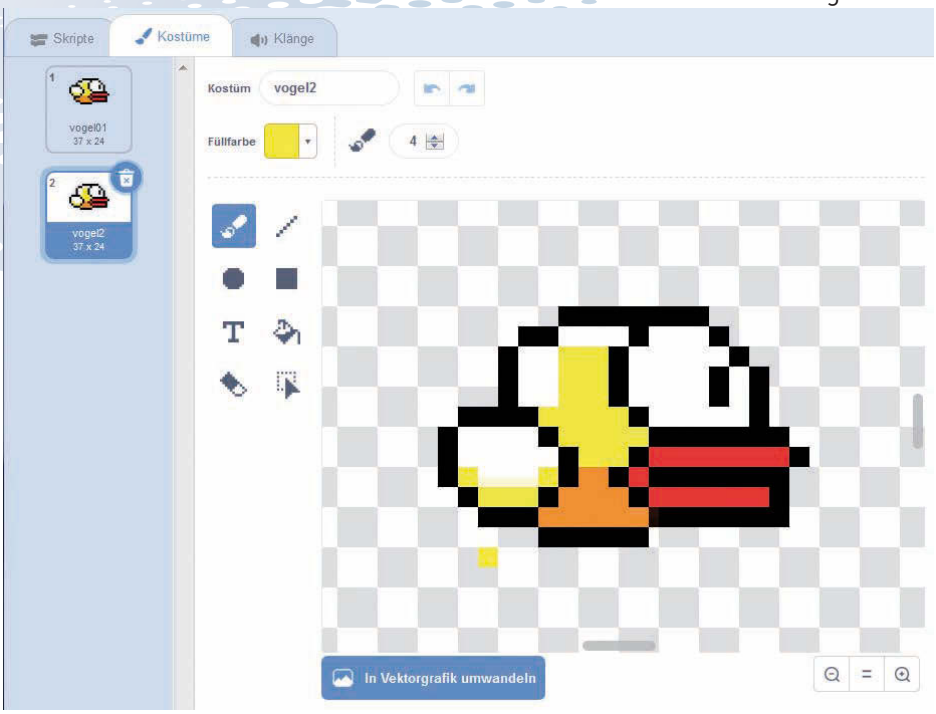


9 Flappy Bird



3 Male jetzt mit dem Pinsel-Werkzeug die fehlenden Grafikteile nach. Mit der passenden Linienbreite wie auf der Abbildung ist der Pinsel genauso groß wie ein Kästchen in der Grafik des Vogels. So kannst du mit ein paar Klicks die fehlenden Teile nachmalen. Mit dem Pipettensymbol im Auswahlfeld ***Füllfarbe*** holst du die benötigten Farben direkt aus dem Bild.

4 Baue ein kleines Programm mit einer Endlosschleife. Sie soll bei Klick auf das grüne Fähnchen gestartet werden und ständig das Kostüm des Vogels wechseln. Dadurch sieht es aus, als schlage er schnell mit den Flügeln.



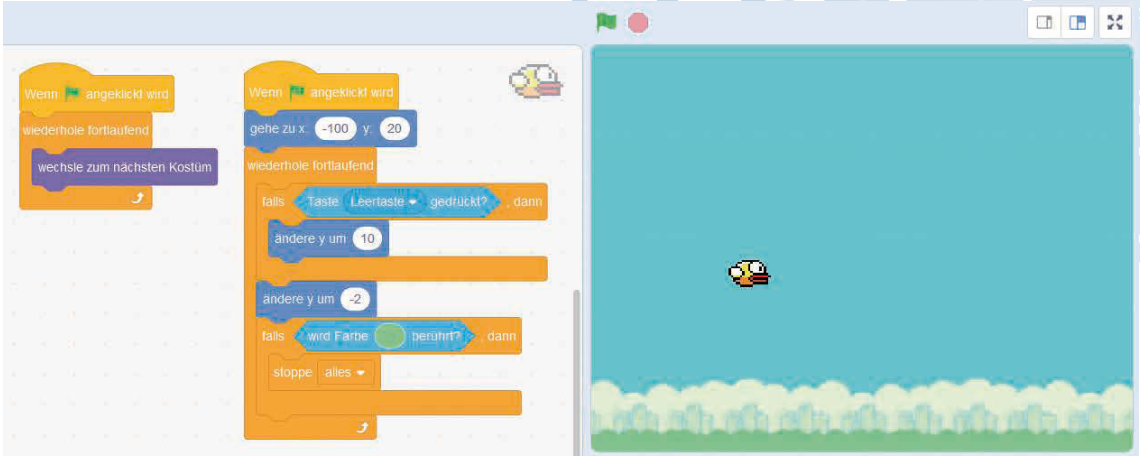
5 Mit einem Druck auf die Leertaste soll der Vogel ein Stück aufsteigen und dann langsam wieder absinken. Allerdings darf er nicht auf der Wiese landen, sonst ist das Spiel vorbei. Erstelle dazu ein zweites Programm für den Vogel, das ebenfalls mit einem Klick auf das grüne Fähnchen startet. Als Erstes wird der Vogel in seine Startposition gebracht.

6 Danach beginnt eine Endlosschleife, die die drei möglichen Spielsituationen verarbeitet: Aufsteigen, Sinken und Spielende bei Berühren der Wiese. Eine ***falls … dann***-Abfrage prüft, ob die Leertaste gedrückt wurde. Wenn ja, bewegt sich der Vogel um zehn Einheiten nach oben.

7 In jedem Schleifendurchlauf soll er ohne irgendeine Bedingung um zwei Einheiten sinken. Füge dazu noch einen Block ***ändere y um -2*** in die Endlosschleife ein.

8 Eine weitere ***falls … dann***-Abfrage prüft, ob der Vogel die Wiese berührt. Verwende hier einen ***wird Farbe … berührt***-Block und wähle im Farbfeld die grüne Farbe der Wiese. Wenn diese Farbe berührt wird, soll das Spiel anhalten. Baue dazu in die Abfrage einen ***stoppe alles***-Block.

9 Jetzt kannst du mit einem Klick auf das grüne Fähnchen das Spiel bereits starten und mit der Leertaste ausprobieren, wie sich der Vogel verhält.



9 Flappy Bird

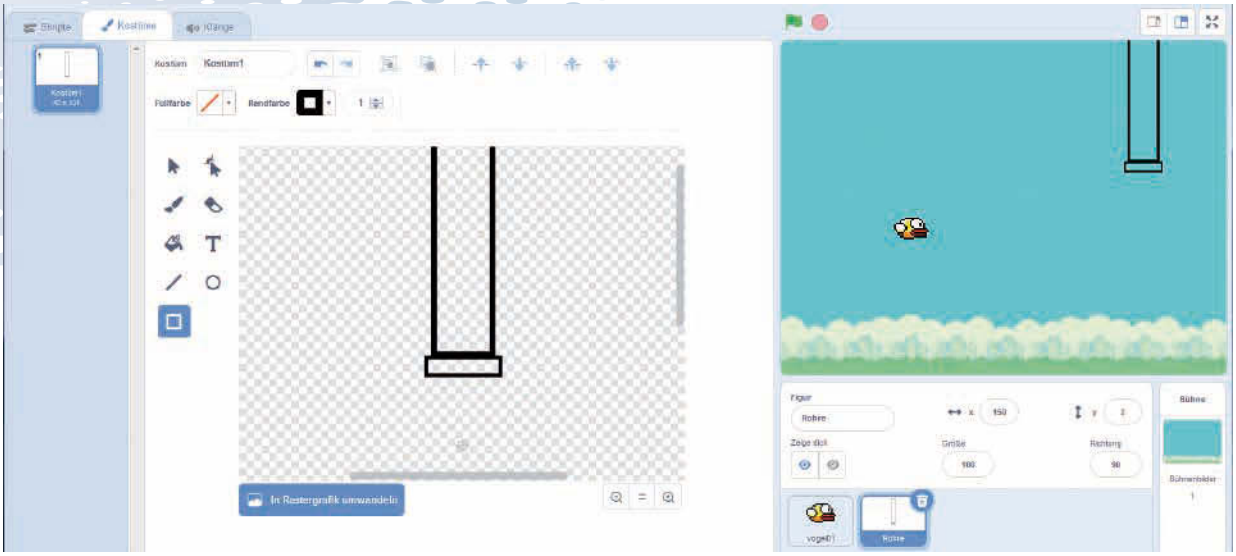
DIE ROHRE KOMMEN

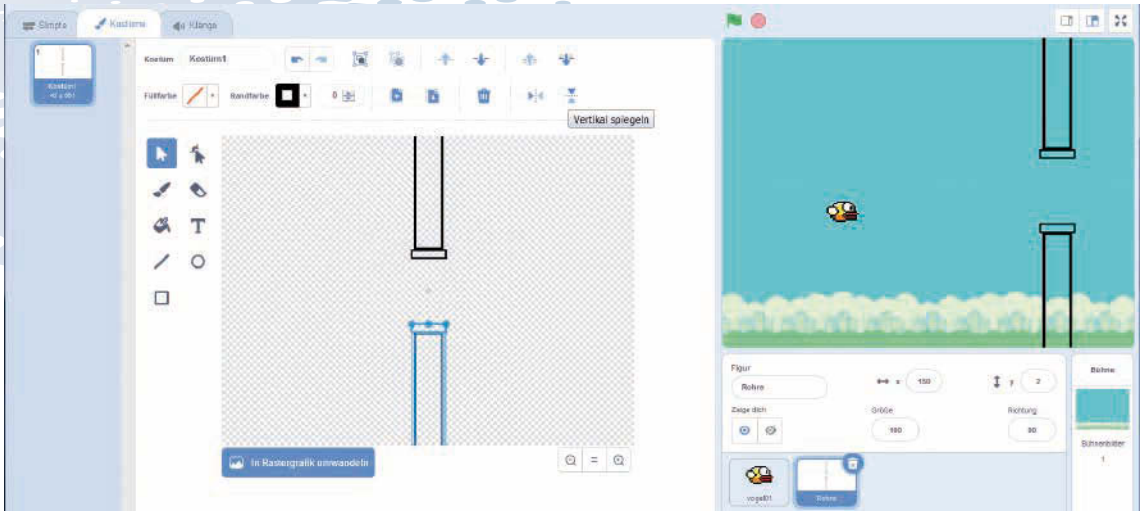
Natürlich ist das Spiel so noch nicht besonders spannend. Interessant wird es erst, wenn die Rohre ins Spiel kommen, zwischen denen der Vogel hindurchfliegen muss. Warum ein Vogel zwischen zwei Rohrenden durchfliegen muss, wird wohl ein Geheimnis des Flappy-Bird-Entwicklers Dong Nguyen bleiben. Um das Spiel möglichst realistisch nachzubilden, verwenden wir ebenfalls zwei solcher Rohre.

1 Klicke in der Figurenliste auf das Symbol ***Malen*** bei ***Figur wählen*** und gib dieser Figur den Namen ***Rohre***.

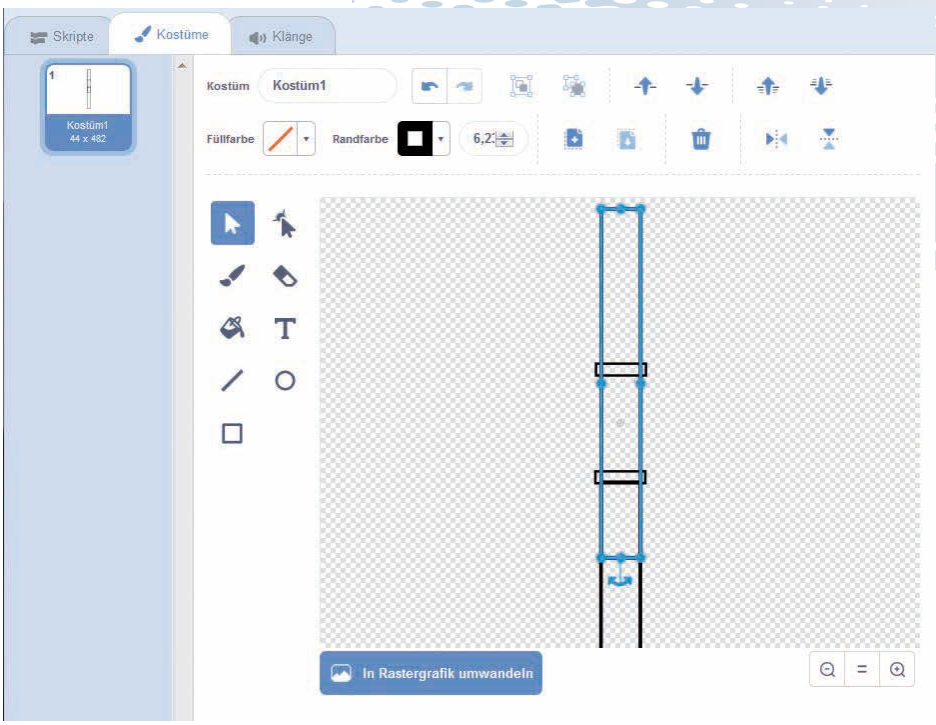
2 Zeichne jetzt im Vektormodus zwei Rechtecke, ein schmales hohes und ein kleines flaches, das die Rohrmuffe am Ende darstellt. Auf der Bühne siehst du die Größe der Figur. Das Rohrende soll etwa so breit wie der Vogel sein.

3 Markiere die beiden Rechtecke und dupliziere sie mit dem ***Kopieren***-Symbol. Spiegele die kopierten Teile mit einem Klick auf ***Vertikal spiegeln***. Schiebe die Rohre so, dass der Objektmittelpunkt, das kleine graue Kreuz, in der Mitte zwischen den beiden Rohrenden liegt.

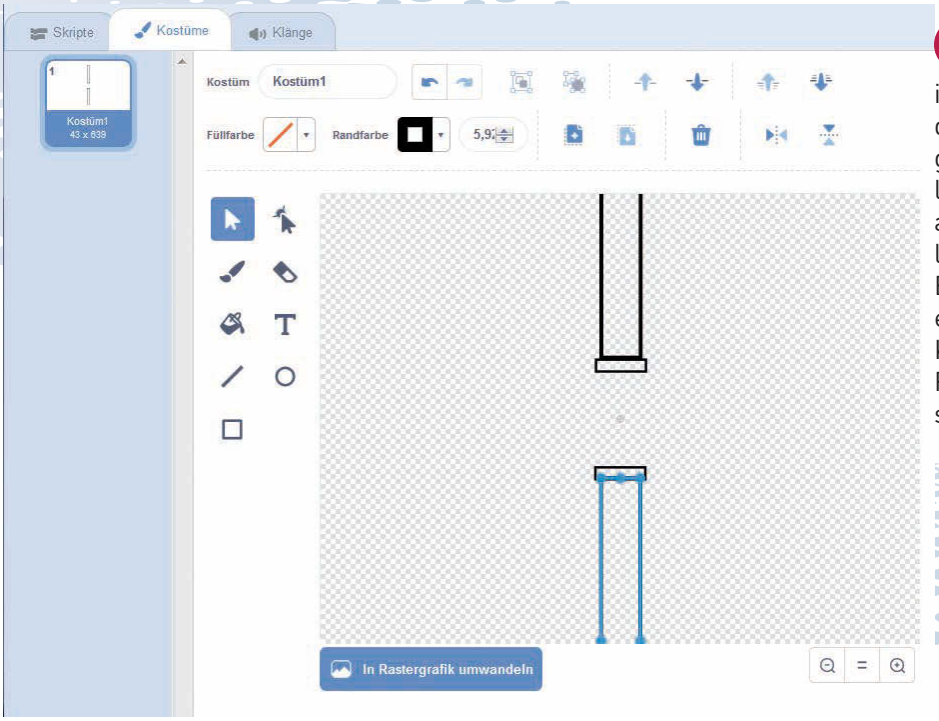




4 Zoome unten rechts im Malprogramm auf 100 % zurück. Klicke mit dem Pfeilsymbol den Griff in der Mitte des unteren Endes des oberen Rohrs an und ziehe dieses damit noch weiter nach unten in die Länge. Schiebe das neue untere Ende mit den Pfeiltasten wieder in die richtige Position. Das Rohr ragt jetzt oben aus dem Bild heraus, ist dort aber nicht mehr zu sehen. Mache das Gleiche mit dem unteren Rohr. Beide Rohre zusammen müssen deutlich länger sein, als die Bühne hoch ist, damit der Zwischenraum, durch den der Vogel fliegt, auf unterschiedlichen Höhen liegen kann.

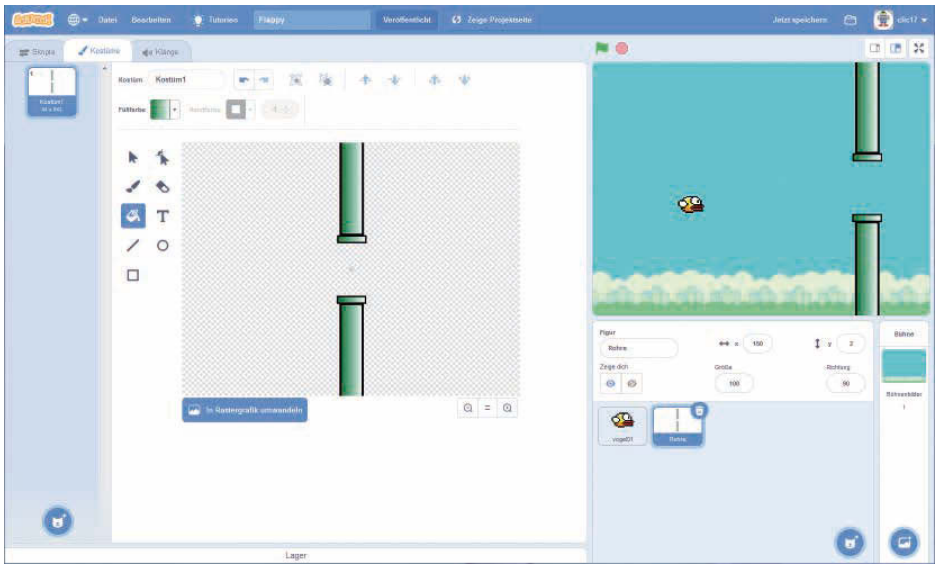


9 Flappy Bird



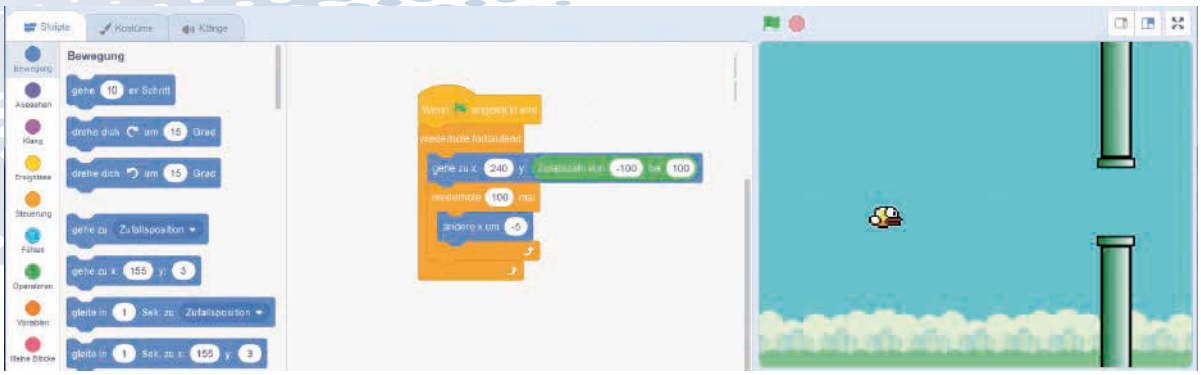
6 Im Spiel bewegt sich der Vogel nur in senkrechter Richtung, die Rohre dagegen waagerecht von rechts nach links, sodass es aussieht, als würde der Vogel von links nach rechts fliegen. Baue für die Figur ***Rohre*** ein Programm, das bei Klick auf das grüne Fähnchen eine Endlosschleife startet.

5 Im Originalspiel haben die Rohre einen grünen Farbverlauf, den unsere Rohre auch bekommen sollen. Klicke auf das Symbol ***Füllfarbe*** und wähle bei den Füllmustern den senkrechten Farbverlauf. Stelle für die linke Farbe mit den Farbreglern ein dunkles Grün und für die rechte Farbe Weiß ein. Klicke dann mit dem Farbeimer nacheinander in die vier Rechtecke, die die Rohre bilden, um sie mit dem Farbverlauf zu füllen.



7 Am Anfang starten die Rohre ganz rechts außen am Bühnenrand bei der x-Koordinate ***240***. Der Zwischenraum, durch den der Vogel fliegt, kann auf unterschiedlichen Höhen liegen. Deshalb wird die y-Koordinate des Startpunkts durch eine Zufallszahl zwischen ***-100*** und ***100*** festgelegt. Die Koordinaten einer Figur werden immer am Objektmittelpunkt gezählt. Deshalb ist es wichtig, dass dieser genau in der Mitte zwischen den beiden Rohren liegt.

8 Dann startet eine weitere Schleife, die die Rohre in ***100*** einzelnen Schritten je ***5*** Einheiten nach links bewegt. Am Ende kommen die Rohre am linken Bildschirmrand an. Danach startet die Bewegung wieder rechts außen, was wie ein neues Rohr aussieht.



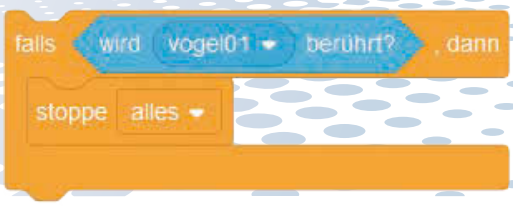
9 Jetzt kannst du mit einem Klick auf das grüne Fähnchen das Spiel wieder starten und mit der Leertaste versuchen, den Vogel durch die Rohre fliegen zu lassen. Allerdings passiert noch nichts, wenn er anstößt. Er fliegt dann einfach weiter. Das Spiel endet nur, wenn der Vogel auf der Wiese landet.

KOLLISIONSERKENNUNG UND PUNKTE

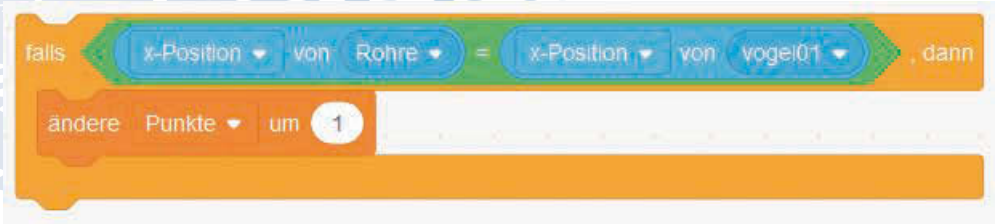
Um das Spiel perfekt zu machen, brauchen wir noch eine Kollisionserkennung, die prüft, ob Vogel und Rohre sich gegenseitig berühren, sowie einen Punktezähler, der jeden erfolgreichen Durchflug durch die Rohre zählt. Beide Abfragen werden in die Bewegung der Rohre mit eingebaut.

0 Lege auf der Blockpalette ***Variablen*** eine neue Variable ***Punkte*** an. Diese wird am Anfang des Spiels auf ***0*** gesetzt. Die Variable soll natürlich auf der Bühne sichtbar sein, damit du deine Punkte jederzeit sehen kannst.

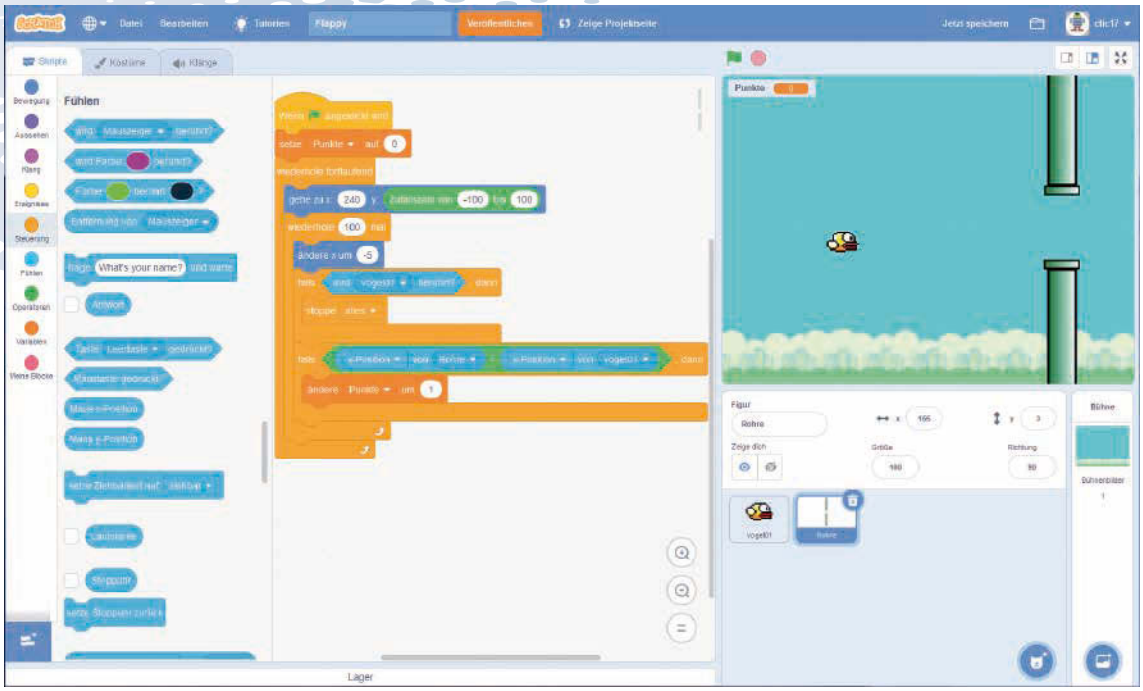
2 Nach jedem Bewegungsschritt der Rohre muss das Programm prüfen, ob sich Vogel und Rohre berühren. Wenn ja, soll das Programm sofort enden.



3 Ob der Vogel durch die Rohre fliegt oder eigentlich die Rohre am Vogel vorbeifliegen, lässt sich über die x-Koordinate ermitteln. Wenn die x-Koordinate des Objektmittelpunkts der Rohre, der genau zwischen den Rohrenden liegt, und die x-Koordinate des Objektmittelpunkts des Vogels gleich sind, fliegt er gerade hindurch. Das wird nach jedem Bewegungsschritt der Rohre ebenfalls überprüft. Beim Durchflug gibt es einen Punkt.



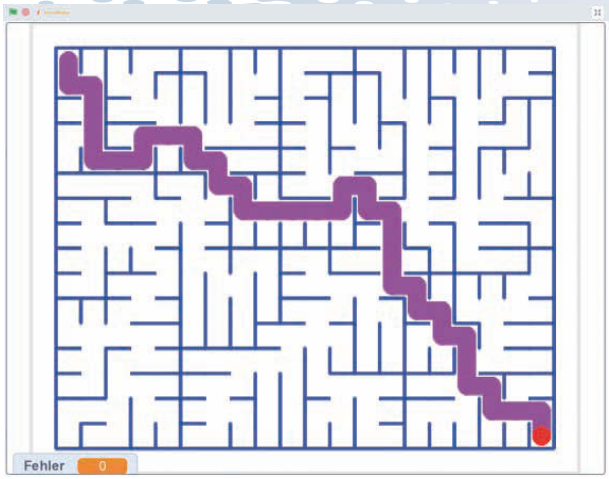
Damit ist das Spiel auch schon fertig. Starte es mit einem Klick auf  das grüne Fähnchen und versuche, den Vogel möglichst oft durch die Rohre fliegen zu lassen und so möglichst viele Punkte zu sammeln.



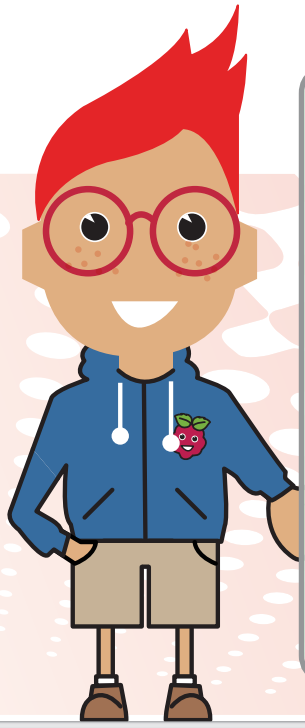
10 Labyrinth

Seit Jahrtausenden faszinieren Labyrinthe und Irrgärten unterschiedlichster Formen die Menschen. Besonderes Interesse löst diese Form der Grafik bei Künstlern, aber auch bei Mathematikern und Logikern aus. Im Computerzeitalter stellen sowohl das Gestalten wie auch das Lösen solcher Labyrinthe immer wieder interessante Herausforderungen an Programmierer.

Dieses Scratch-Projekt liefert einen ersten Einblick in die Geometrie und die Logik von Labyrinthen.



Das Scratch-Programm generiert im ersten Schritt ein zufälliges Labyrinth. Anschließend kannst du mit den Pfeiltasten den roten Punkt



ES GIBT IMMER GENAU EINEN WEG DURCH DAS LABYRINTH

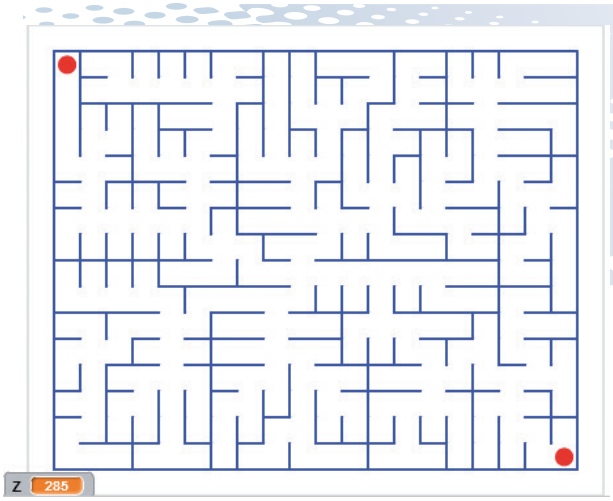
Jedes Labyrinth sieht etwas anders aus, und durch jedes gibt es genau einen Weg von links oben nach rechts unten – probiere es aus.

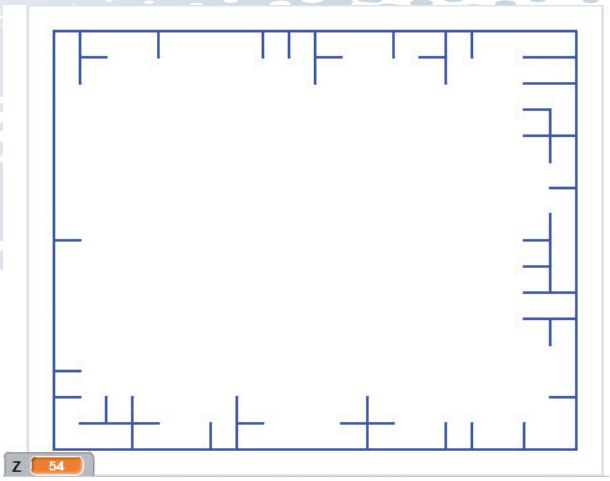
Dieser garantierte Weg basiert auf einem einfachen Prinzip. Beim Generieren des Labyrinths wird an den Rändern angefangen. Von einem zufälligen Punkt auf einer Wand aus wird ein neues Wandsegment in den freien Raum gezogen. Es darf nicht an einer bestehenden Wand enden. So wird nacheinander immer zufällig ein Wandpunkt (im Raster des Labyrinths) ausgewählt, und von dort werden Wände auf angrenzende freie Rasterpunkte gezogen – so lange, bis alle Rasterpunkte belegt sind.

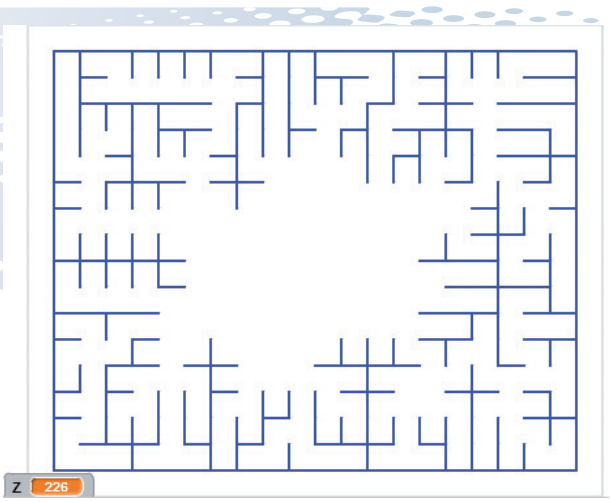
Dadurch, dass nie eine Wand zwischen zwei Wänden gezogen wird, kann es keine durchgehende Sperre geben, die einen Lösungsweg verhindert. Umgekehrt fängt keine Wand an einem freien Punkt im Raum an, sodass es auch keine Inseln geben kann, woraus sich zwei unterschiedliche Wege ergeben würden.

durch das Labyrinth bewegen, ohne an den Wänden anzustoßen.

Dieses Prinzip erkennst du am besten, wenn du das Generieren eines Labyrinths genau mitverfolgst.

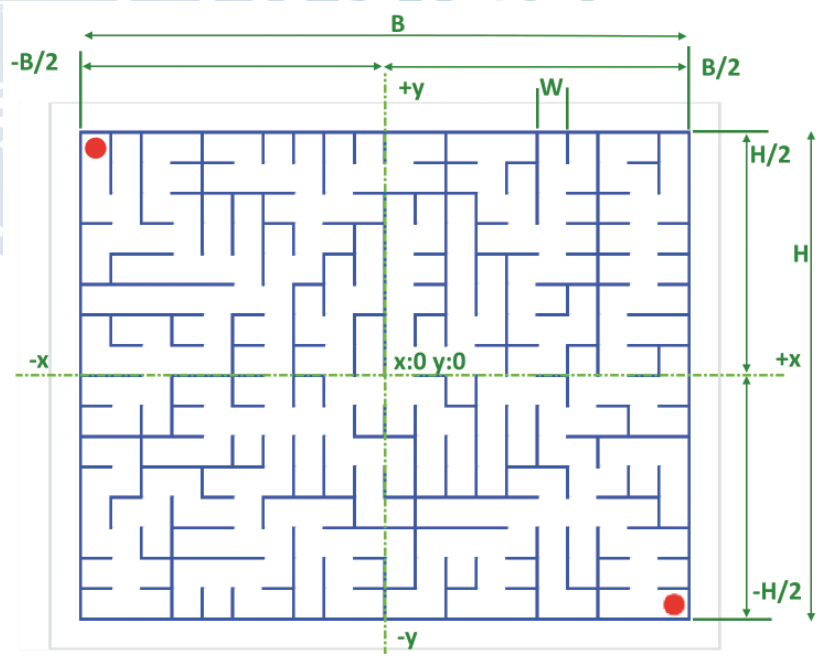






DAS KOORDINATENSYSTEM DES LABYRINTHS

Die Scratch-Bühne hat zwar eine genau definierte Größe, trotzdem ist dieses Programm dank der Verwendung von Variablen so allgemein gehalten, dass es theoretisch auf unterschiedlich großen Bühnen laufen könnte. Außerdem verliert man bei der Berechnung mit Variablen nicht so schnell die Übersicht wie bei absoluten Zahlen, bei denen man irgendwann nicht mehr weiß, wie sie sich zusammensetzen.



Zur Beschreibung bestimmter Punkte im Labyrinth werden die Variablen ***B*** (= Breite), ***H*** (= Höhe) und ***W*** (= Wegbreite) verwendet. Die Wegbreite legt das Raster des Labyrinths fest.

Der Mittelpunkt des Labyrinths liegt bei den Koordinaten ***x:0*** und ***y:0***.

Das Labyrinth hat eine Breite von ***B***, gezählt in Wegbreiten, nicht in Koordinateneinheiten. Die Breite in Koordinateneinheiten ergibt sich durch Multiplikation mit der Wegbreite ***W***. ***B = 20*** und ***W = 20***, daraus ergibt sich eine Gesamtbreite von ***400*** Koordinateneinheiten.

Das Labyrinth hat eine Höhe von ***H***, ebenfalls gezählt in Wegbreiten. Die Höhe in Koordinateneinheiten ergibt sich wieder durch Multiplikation mit der Wegbreite ***W***. ***H = 16*** und ***W = 20***, daraus ergibt sich eine Gesamthöhe von ***320*** Koordinateneinheiten.

ZEICHNE DAS LABYRINTH

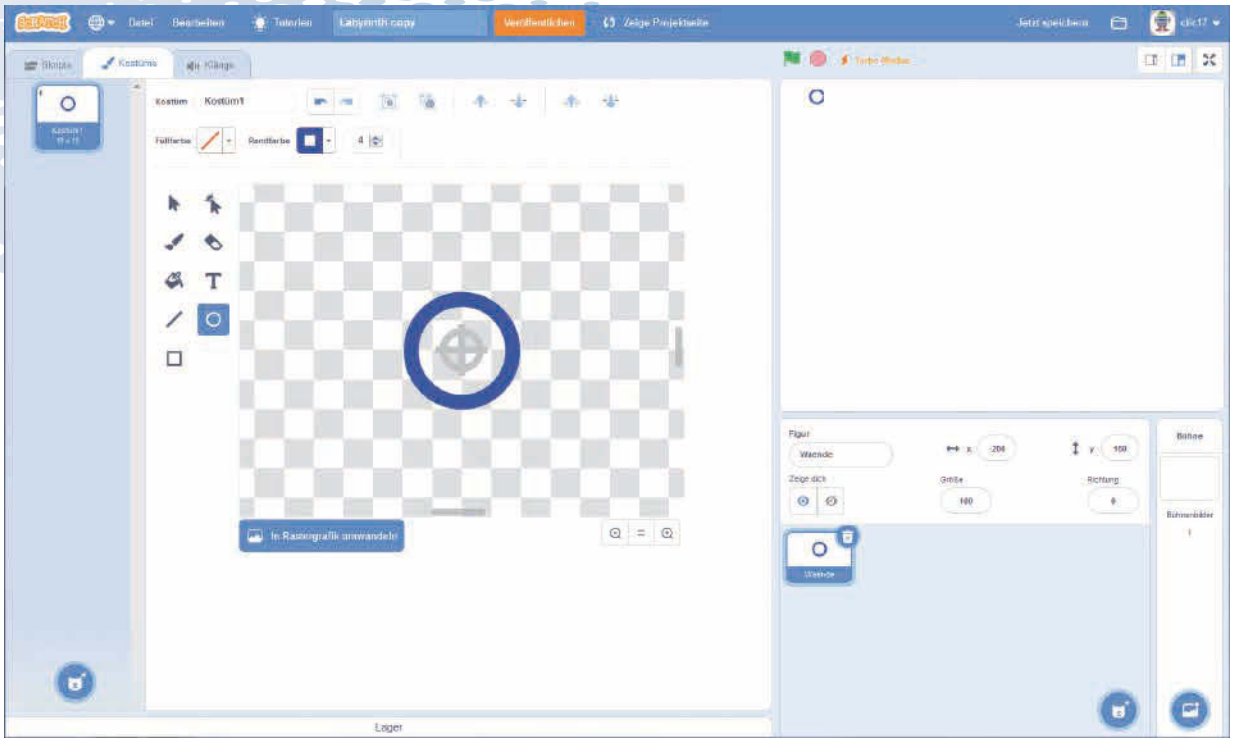
Im Programm verwenden wir zwei einfache Figuren, eine zeichnet die Wände und ist selbst nicht sichtbar, mit der anderen bewegst du dich später durch das Labyrinth und zeichnest dabei den Weg.

0 Lege in Scratch ein neues Projekt an und lösche die Katze. Installiere die Erweiterung ***Malstift***, mit der die Linien des Labyrinths gezeichnet werden.

2 Klicke in der Figurenliste auf das Symbol ***Neue Figur zeichnen***. Zeichne im Vektormodus einen blauen Kreis, der ***16 x 16*** Einheiten groß ist. Zoome dazu am besten weit hinein. Dann

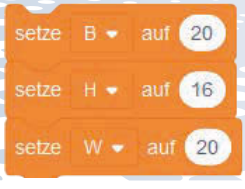
kannst du den Kreis gut in der exakten Größe zeichnen. Der Objektmittelpunkt muss genau in der Mitte des Kreises liegen.

9 Erstelle jetzt die drei Variablen ***B***, ***H*** und ***W***. Sie sollen auf der Bühne nicht sichtbar sein. Achte aber beim Anlegen der Variablen darauf,

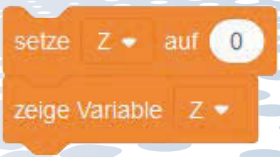


8 Baue für diese Figur ein neues Programm, das startet, wenn man auf das grüne Fähnchen klickt. Dieses Programm soll als Erstes die Figur verstecken, da sie beim Zeichnen nur stört.

dass sie für alle Figuren gelten. Ziehe anschließend drei Blöcke ***setze ... auf ...*** ins Programm, die diese Variablen auf ihre vorgegebenen Werte setzen.

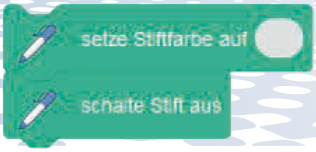


5 Lege noch eine Variable ***Z*** für den Zähler an, der mitzählt, wie viele Rasterpunkte bereits an das Labyrinth angeschlossen sind, und so feststellt, wann es fertig gezeichnet ist. Diese Variable wird mit dem Block ***zeige Variable ...*** auf der Bühne gezeigt. Diese Methode hat gegenüber dem einfachen Setzen des Häkchens den Vorteil, dass die Variable jederzeit angezeigt und wieder versteckt werden kann. Wenn das Labyrinth fertig gezeichnet ist und wir sie nicht mehr brauchen, wird sie versteckt.



6 Bereite jetzt die Bühne für das neue Labyrinth vor. Lösche dazu alle Malspuren und setze die Stiftdicke auf ***2***. Diese beiden Blöcke aus der Erweiterung ***Malstift*** sind besonders wichtig, wenn das Programm schon einmal gelaufen ist, damit das nächste Labyrinth wieder mit den Grundeinstellungen gezeichnet werden kann.

7 Als Erstes zeichnen wir ein hellgraues Rechteck mit einem Abstand von einer Wegbreite um das eigentliche Labyrinth herum. Das wird später verhindern, dass beim Erzeugen des Labyrinths Mauersegmente außen am Labyrinth angebaut werden. Setze die Stiftfarbe auf Hellgrau und schalte den Stift zunächst aus, um die Figur an die erste Ecke dieses Rechtecks zu bringen.



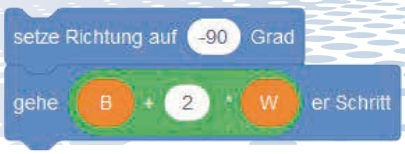
8 Die linke obere Ecke des Labyrinths liegt bei ***x:-B/2 y:H/2*** im Labyrinthraster oder entsprechend ***x:-B/2*W y:H/2*W*** in Koordinateneinheiten. Die linke obere Ecke des grauen Rechtecks soll um jeweils eine Wegbreite weiter links und oben liegen. Daraus ergibt sich diese Formel:

9 Da man in Scratch nicht einfach vor eine Variable ein negatives Vorzeichen setzen kann, teilen wir die Breite durch ***-2*** und erreichen so einen negativen Wert. Schalte an diesem Punkt den Stift ein, um mit dem Zeichnen des hellgrauen Rechtecks zu beginnen.

10 Setze die Richtung auf ***90*** Grad nach rechts und bewege die Figur dann einen Schritt, der um zwei Labyrinthraster länger ist als die Breite des Labyrinths. Damit zeichnest du die obere Kante des Rechtecks.

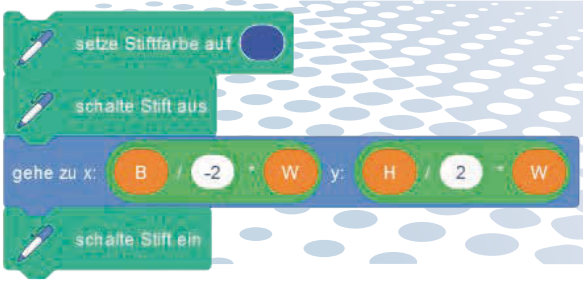
11 Schalte die Richtung auf ***180*** Grad nach unten und zeichne die rechte senkrechte Kante. Diese ist um zwei Labyrinthraster länger als die Höhe des Labyrinths.

12 Schalte die Richtung auf ***-90*** Grad nach links und zeichne die untere waagerechte Kante. Diese ist wieder um zwei Labyrinthraster länger als die Breite des Labyrinths.

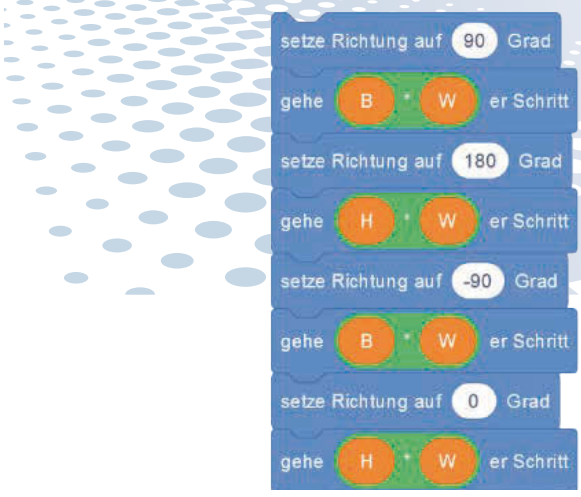


13 Schalte zum Schluss die Richtung auf ***0*** Grad nach oben und zeichne die linke senkrechte Kante des hellgrauen Rechtecks.

14 Als Nächstes zeichnen wir den Umriss des Labyrinths. Das funktioniert ganz ähnlich, da es ebenfalls ein Rechteck ist, dessen Mittelpunkt im Nullpunkt des Koordinatensystems liegt. Setze die Stiftfarbe auf Blau, schalte den Stift aus und bewege die Figur in die linke obere Ecke des Labyrinths bei ***x:-B/2 y:H/2*** im Labyrinthraster oder entsprechend ***x:-B/2*W y:H/2*W*** in Koordinateneinheiten. Schalte dort den Stift wieder ein.

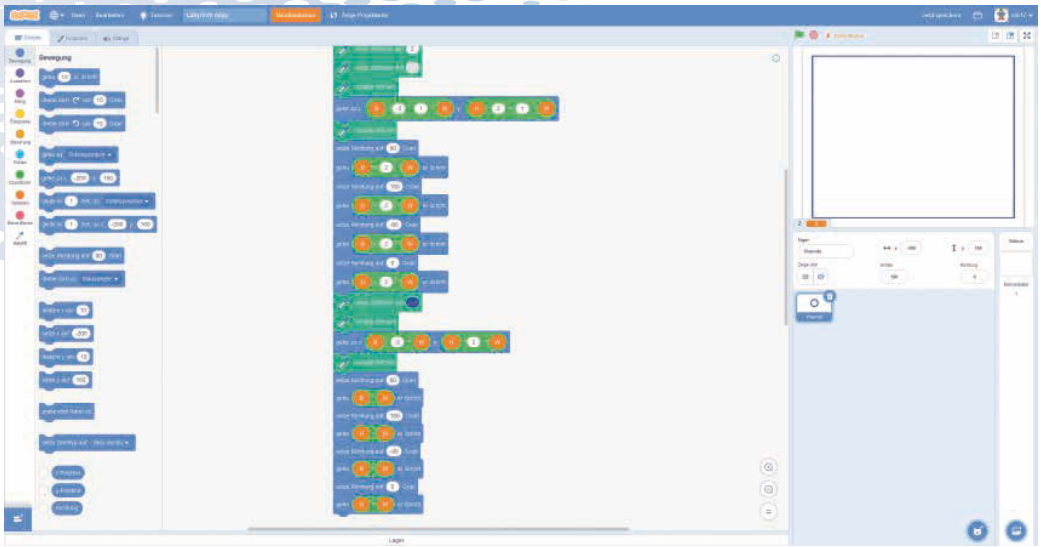


15 Das blaue Rechteck wird genau so gezeichnet wie das hellgraue. Breite und Höhe entsprechen direkt der Breite und Höhe des Labyrinths.



10 Labyrinth

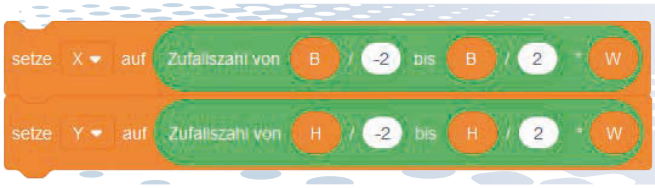
16 Das folgende Bild zeigt, wie das Programm bis jetzt aussieht. Starte es mit einem Klick auf das grüne Fähnchen. Es zeichnet wie erwartet ein hellgraues und ein blaues Rechteck und zeigt die Variable ***Z*** auf der Bühne an.



17 Jetzt kommt der eigentlich interessantere Programmteil, der das Labyrinth aufbaut. Da wir, bedingt durch den Zufallsgenerator, vorher nicht wissen, wie lange es dauert, das Labyrinth aufzubauen, verwenden wir eine ***wiederhole bis ...***-Schleife, die so lange läuft, bis alle inneren Rasterpunkte des Rechtecks an das Netz der Labyrinthmauern angeschlossen sind. Es gibt ***B-1 * H-1*** Rasterpunkte innerhalb des Rechtecks, die die Variable ***Z*** zählt. Bei den vorgegebenen Maßen sind das ***285*** Punkte.

18 In jedem Schleifendurchlauf wird ein Punkt zufällig ermittelt, von dem aus neue Labyrinthmauern gebaut werden. Voraussetzung ist natürlich, dass der Punkt bereits an einer Mauer liegt, da Mauern nach den Baurегeln nicht frei im Raum beginnen können. Die Koordinaten dieses zufällig ermittelten Punkts werden in den Variablen ***X*** und ***Y*** gespeichert. Lege diese beiden Variablen neu an. Sie sollen auf der Bühne nicht sichtbar sein.

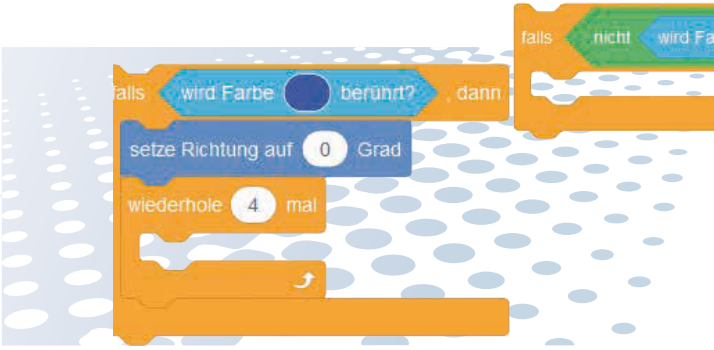
19 Die x-Koordinate kann irgendwo zufällig zwischen der linken und der rechten Kante des Labyrinths liegen, die y-Koordinate irgendwo zwischen der unteren und der oberen Kante.



20 Schalte den Stift aus und bewege die Figur an die zufällig ermittelten Koordinaten.

21 Liegt der zufällig ermittelte Punkt auf einer Mauer, können von dort aus neue Mauern gezeichnet werden. Um das zu überprüfen, verwenden wir den Block ***wird Farbe ... berührt***. Wenn die unsichtbare Figur, die die Mauern zeichnet, die blaue Farbe berührt, mit der die Mauern gezeichnet werden, steht sie auf einem Rasterpunkt mit einer Mauer. Die Figur ist klein genug, um keine Nachbarmauer auf dem nächsten Rasterpunkt zu berühren. Sie kann auch bei der verwendeten Programmlogik nie zwischen zwei Rasterpunkten stehen. Die Farbe kannst du mit der Pipette im Farbauswahlfeld von der bereits gezeichneten Umrandung des Labyrinths abgreifen.

22 Trifft die Bedingung zu, versucht das Programm, innerhalb der Schleife möglichst viele Mauern von dem ermittelten Punkt aus in die vier Richtungen zu zeichnen. Es können aber höchstens drei Mauern neu gezeichnet werden, da ein Punkt nur als Anfangspunkt verwendet wird, wenn er bereits an einer Mauer liegt. Die vier möglichen Richtungen werden mit einer weiteren Schleife abgearbeitet. Setze vor Beginn dieser Schleife die Richtung auf ***0***, was als Winkelangabe nach oben dient.



23 Schalte den Stift aus und bewege die Figur in der vorgegebenen Richtung um ein Rasterfeld.

24 Diese Bewegung dient nur dazu, festzustellen, ob in der angegebenen Richtung auf dem nächsten Rasterfeld eine Mauer ist. Ist dort nämlich keine Mauer, kann vom letzten Punkt, dessen Koordinaten immer noch in den Variablen ***X*** und ***Y*** gespeichert sind, eine neue Mauer zum aktuellen Rasterpunkt gezeichnet werden. Diese Abfrage überprüft, dass weder die Farbe Blau noch die Farbe Hellgrau berührt wird. In Hellgrau ist das äußere Rechteck im Abstand von einem Rastermaß um das Labyrinth herum gezeichnet. Auf diese Weise stellst du sicher, dass keine Mauern entstehen, die vom blauen Umriss des Labyrinths nach außen laufen.

25 Trifft die Abfrage zu, soll eine Mauer gezeichnet werden. Schalte dazu den Stift ein und erhöhe die Variable ***Z*** um ***1***. Sie zählt die angeschlossenen Rasterpunkte. Bei ***285*** ist das Labyrinth fertig.

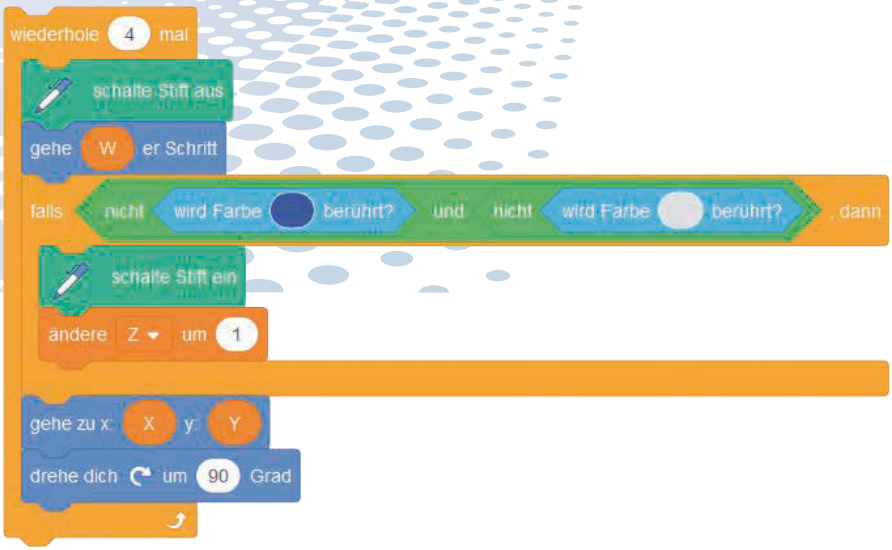
26 Unabhängig vom Ergebnis der Abfrage wird die Figur danach wieder an ihren Ausgangspunkt bewegt, den zufällig ermittelten Punkt, dessen Koordinaten in den Variablen ***X*** und ***Y*** gespeichert sind. Ist der Stift eingeschaltet, wird dabei eine Mauer gezeichnet, andernfalls kommt die Figur trotzdem an diesen Punkt zurück, um, möglicherweise um 90 Grad weitergedreht, eine Mauer zu zeichnen.

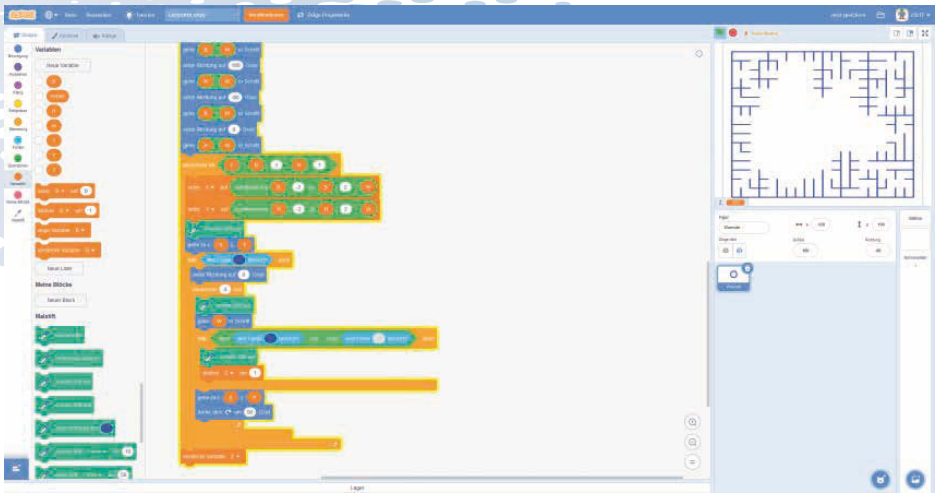
27 Es folgt noch ein Block ***drehe dich um 90 Grad***. Danach wiederholt sich die Schleife und versucht, eine weitere Mauer vom gleichen Ausgangspunkt aus zu zeichnen.

28 Hänge ganz am Ende noch einen ***verstecke Variable ...***-Block an, der die Variable ***Z*** versteckt. Sie wird, nachdem das Labyrinth fertig ist, nicht mehr benötigt.

29 Schalte den Turbo-Modus ein. Auch dann kann es einige Sekunden dauern, bis das Labyrinth fertig gezeichnet ist.

30 Starte das Programm mit einem Klick auf das grüne Fähnchen.





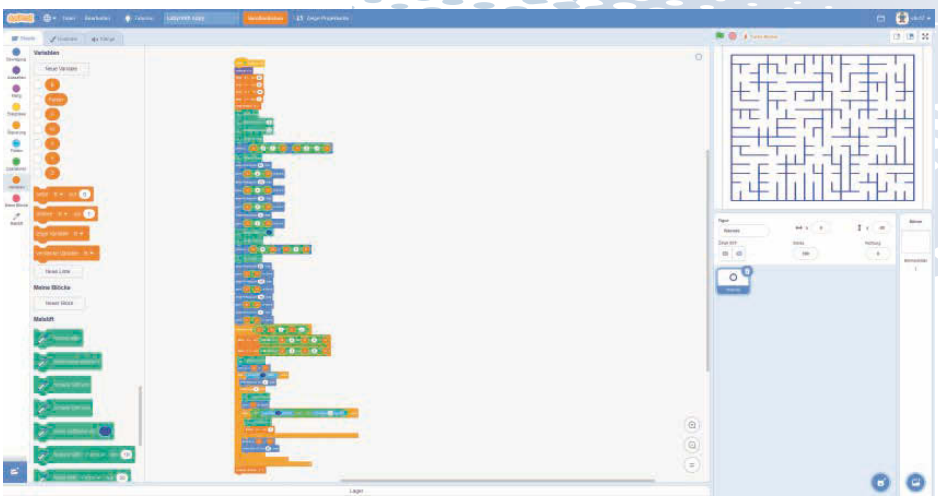
31 Jetzt kannst du beobachten, wie sich das Labyrinth aufbaut. Die Variable ***Z*** zählt die angeschlossenen Rasterpunkte hoch bis ***285*** und wird am Ende wieder versteckt.

Das Programm ist inzwischen so groß, dass es je nach Bildschirmauflösung auch nicht mehr auf große Bildschirme passt. Du kannst im Programmfenster scrollen und unten rechts mit den Lupensymbolen auch zoomen. Das Symbol mit dem Gleichheitszeichen zoomt die Programmblöcke wieder auf Originalgröße.

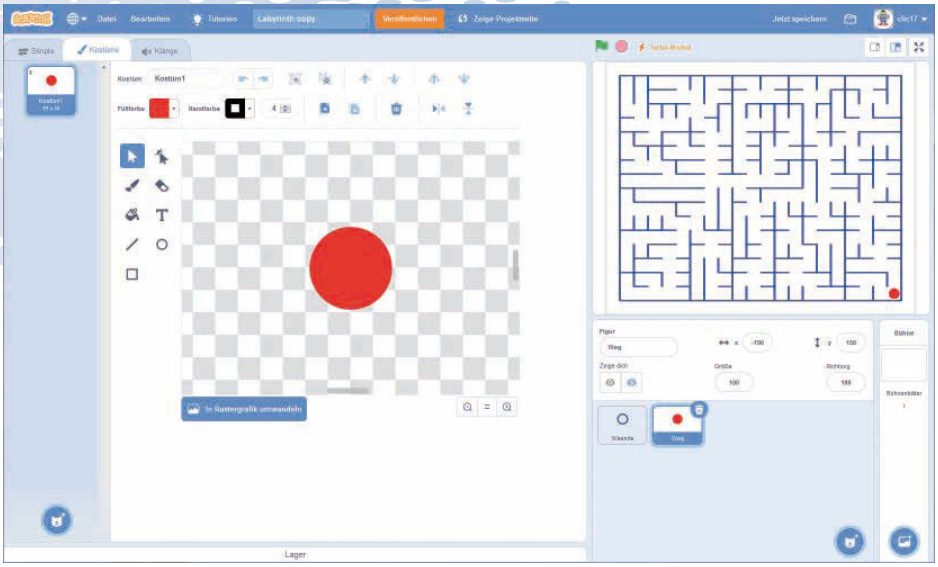
FINDE DEN WEG DURCH DAS LABYRINTH

Um durch das Labyrinth zu laufen, verwenden wir eine zweite Figur, einen roten Kreis, der mit den Pfeiltasten gesteuert wird.

0 Zeichne eine neue Figur, einen rot gefüllten Kreis mit einer Größe von ***15 x 15*** Einheiten. Du kannst diese Figur natürlich auch über den



10 Labyrinth



Menüpunkt ***Duplizieren*** aus der ersten Figur erstellen. Die neue Figur heißt ***Weg***, weil sie später den Weg durch das Labyrinth zeichnet.

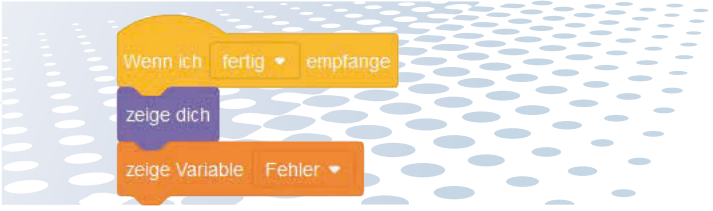
2 Lege eine neue Variable ***Fehler*** an. Sie soll mitzählen, wie oft du auf dem Weg durch das Labyrinth gegen eine Mauer rennst.

3 Baue im Skriptbereich dieser Figur ein neues Programm, das bei Klick auf das grüne Fähnchen startet. Dieses einfache Programm soll am Anfang die rote Figur noch verstecken, bis das Labyrinth fertig ist. Die Variable ***Fehler*** wird zu Beginn ebenfalls versteckt und auf ***0*** gesetzt.

4 Die Figur soll erst auftauchen, wenn das Labyrinth fertig gezeichnet ist. Dazu brauchen wir neue Blöcke, die es einer Figur möglich machen, an eine andere eine Nachricht zu senden. Bisher galten alle Programmblöcke immer nur für eine Figur. Wähle nochmal die erste Figur ***Waende*** aus, die das Labyrinth zeichnet, und hänge an das Ende des Programms einen Block ***sende ... an alle*** aus der Blockpalette ***Ereignisse*** an. Klicke in diesem Block auf das Listenfeld, wähle im Menü ***Neue Nachricht ...*** und gib den Text ***fertig*** ein. An dieser Stelle kannst du einen beliebigen Text eingeben. Bereits verwendete Texte können direkt aus der Liste ausgewählt werden.

5 Schalte wieder auf die rote Figur ***Weg***. Ziehe aus der Blockpalette ***Ereignisse*** einen neuen Programmblock ***Wenn ich ... empfange*** in den Skriptbereich, klicke auf das Listenfeld und wähle den zuvor eingegebenen Text ***fertig*** aus.

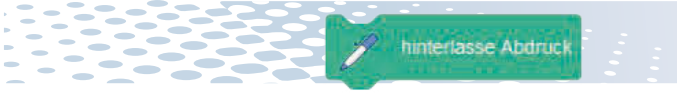
6 Dieser Programmblock wird gestartet, wenn die rote Figur ***Weg*** von der blauen Figur ***Waende*** die Nachricht ***fertig*** empfängt. Jetzt sollen ein paar Grundeinstellungen vorgenommen werden. Die Figur zeigt sich auf der Bühne. Am Anfang war sie noch versteckt. Auch die Variable ***Fehler*** wird jetzt angezeigt, damit du jederzeit sehen kannst, wie oft du schon gegen eine Mauer gelaufen bist.



7 Während du durch das Labyrinth läufst, wirst du eine kräftige Spur hinterlassen. Setze die Stiftfarbe auf Violett und die Stiftdicke auf ***14***. Schalte dann den Stift erst einmal aus, um die Figur in die Startposition zu bringen, ohne quer über das Labyrinth zu malen.

8 Bewege die Figur in das untere rechte Rasterfeld des Labyrinths. An dieser Stelle soll das Ziel sein. Die rote Figur bewegt sich immer um ein halbes Rastermaß neben den Mauern genau in der Mitte der Wege des Labyrinths. Der Zielpunkt liegt von der rechten unteren Ecke des Labyrinths um jeweils ein halbes Raster nach links und oben verschoben.

9 An dieser Stelle soll die Figur einen Abdruck hinterlassen, bevor sie sich an den Startpunkt oben links bewegt. Ein solcher Abdruck funktioniert wie ein Stempel- oder ein Fußabdruck. Er ist nur ein Bild, aber keine wirkliche Figur, kann also auch keine weiteren Programmblöcke nutzen. Hänge dazu einen Block ***hinterlasse Abdruck*** aus der ***Malstift***-Erweiterung an das Programm.



10 Da der Stift immer noch ausgeschaltet ist, kann sich die Figur jetzt einfach an den Startpunkt bewegen. Dieser liegt von der linken

oberen Ecke des Labyrinths um jeweils ein halbes Raster nach rechts und unten verschoben.

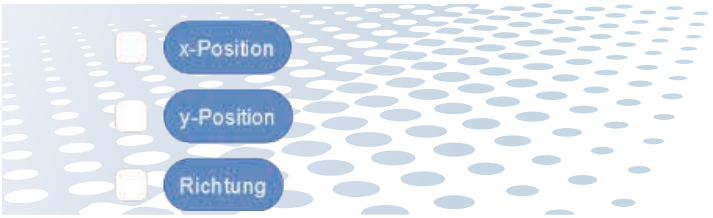
11 Bei komplizierteren Aktivitäten oder wenn es etwas länger dauert, ist es hilfreich, wenn der Benutzer erfährt, was er tun soll. Mit dem Block ***sage ... für ... Sek*** von der Blockpalette ***Aussehen*** kann eine Figur für eine bestimmte Zeit einen beliebigen Text in einer Sprechblase anzeigen. Hänge diesen Block an das Programm und trage den abgebildeten Text ein.

12 Ab jetzt braucht das Hauptprogramm nur noch abzuwarten, bis die rote Figur am Ziel angekommen ist. Die Tastatursteuerung und die Bewegung der Figur werden über eigenständige Programmblöcke geregelt. Hänge dazu einen ***warte bis ...***-Block aus der Blockpalette ***Steuerung*** an das Programm und ziehe in das Feld für die Bedingung einen ***... und ...***-Block aus der Blockpalette ***Operatoren***, da zwei Bedingungen gleichzeitig erfüllt sein müssen.

13 Die beiden Bedingungen lauten:

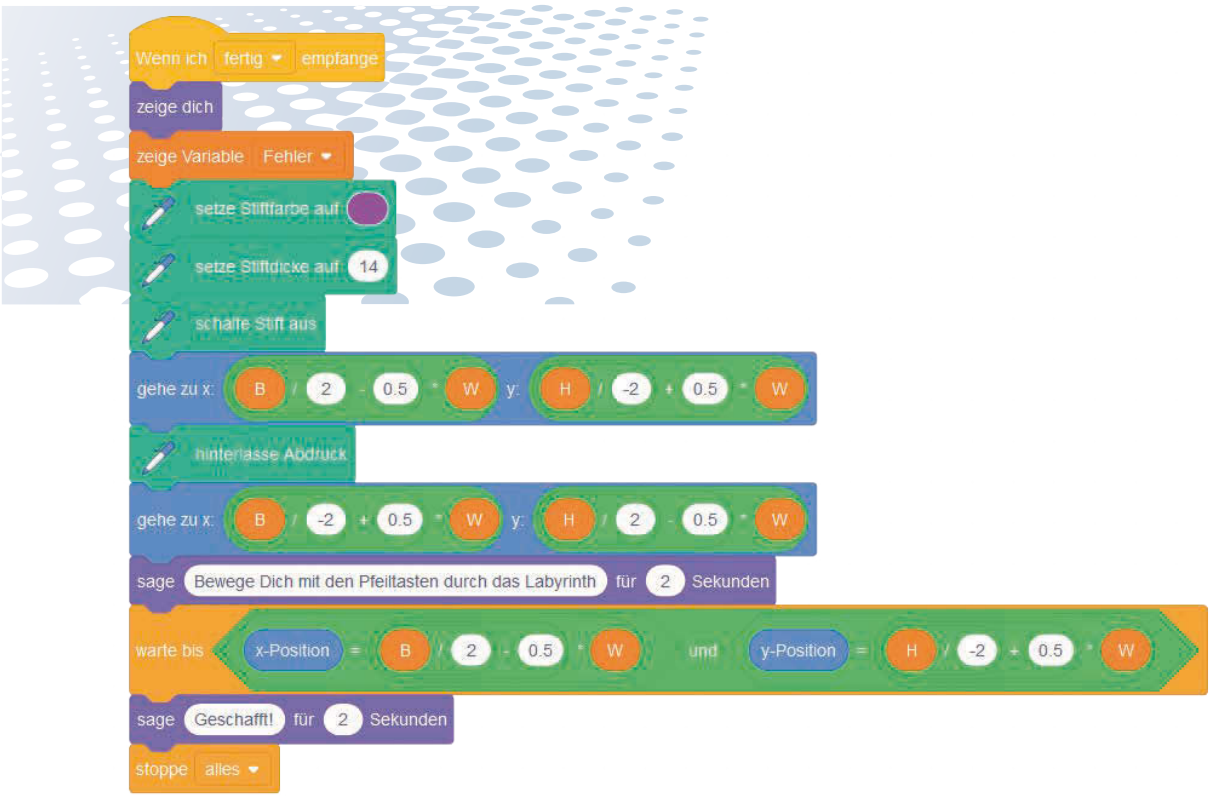
- Die ***x-Position*** der Figur muss in der letzten Spalte des Labyrinths ganz rechts außen sein.
- Die ***y-Position*** der Figur muss in der untersten Zeile des Labyrinths sein.

14 Wenn beide Bedingungen gleichzeitig erfüllt sind, befindet sich die Figur in der rechten unteren Ecke des Labyrinths. ***x-Position***, ***y-Position*** und auch ***Richtung*** können wie Variablen zum Rechnen verwendet werden. Die aktuellen Werte lassen sich auch auf der Bühne anzeigen, du findest die Blöcke auf der Blockpalette ***Bewegung***.

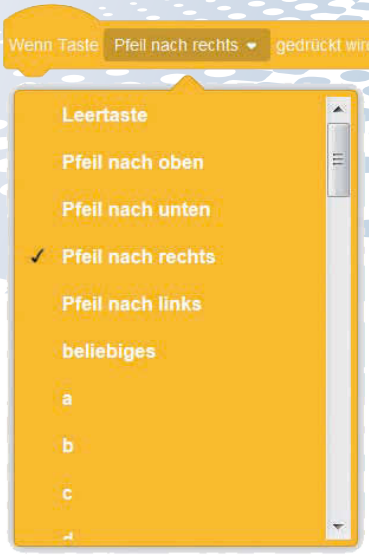


15 Wenn die rote Figur am Ziel angekommen ist, liefert ein ***sage ... für ... Sek***-Block eine Erfolgsmeldung, danach werden alle Programmaktivitäten beendet.

16 Das Programm ist aber noch nicht fertig. Jetzt müssen die Pfeiltasten abgefragt und danach muss die Figur entsprechend bewegt werden. Hierfür verwenden wir ***Wenn Taste ...***



gedrückt-Blöcke von der Blockpalette ***Ereignisse***. Dieser Block bietet eine lange Auswahlliste an Tasten, die Aktionen auslösen können.



17 Wähle hier den ***Pfeil nach rechts***. In diesem Fall soll die Bewegungsrichtung ebenfalls nach rechts auf 90 Grad gesetzt werden.

18 Die Aktionen, die durch die Pfeiltasten ausgelöst werden, unterscheiden sich nur in der Richtung, sind sonst aber gleich. Deshalb bauen wir dazu einen Programmblock, der von allen vier Pfeiltasten aufgerufen wird. Hänge dazu ans Ende wieder einen ***sende ... an alle***-Block an, der diesmal den Befehl ***gehe*** sendet. Natürlich kannst du auch einen anderen Text verwenden.

19 Dupliziere den gesamten Programmblock noch dreimal und ändere jeweils die auslösende Taste und die entsprechende Richtung. Die Programmblöcke sind hier der Übersichtlichkeit

halber im Skriptfenster so wie die Tasten angeordnet. Dies ist aber nicht notwendig. Das Programm funktioniert immer. Die Lage der einzelnen Blöcke im Fenster spielt keine Rolle.

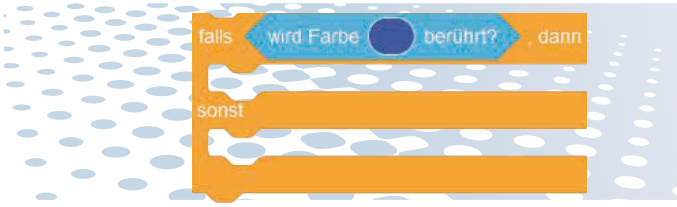
20 Ziehe noch einen Block ***Wenn ich ... empfange*** aus der Blockpalette ***Ereignisse*** in das Skriptfenster und wähle dort den zuvor verwendeten Text ***gehe*** aus.

21 Als Erstes wird die rote Figur einen halben Schritt weit in die durch den Tastendruck festgelegte Richtung gehen.

22 Ginge die Figur einen ganzen Schritt im Labyrinthraster, würde sie auf jeden Fall auf dem benachbarten Weg landen. Durch den Trick mit dem halben Schritt gibt es zwei Möglichkeiten:

- Die Figur kommt mitten in einer Wand zum Stehen.
- Die Figur kommt auf einem Weg zwischen zwei Rasterfeldern zum Stehen.

23 Diese beiden möglichen Ergebnisse haben natürlich völlig unterschiedliche Folgen. Eine ***falls ... dann ... sonst***-Abfrage prüft, ob die Figur nach dem halben Schritt die blaue Farbe einer Mauer berührt – also in diesem Fall genau in einer Mauer steht.



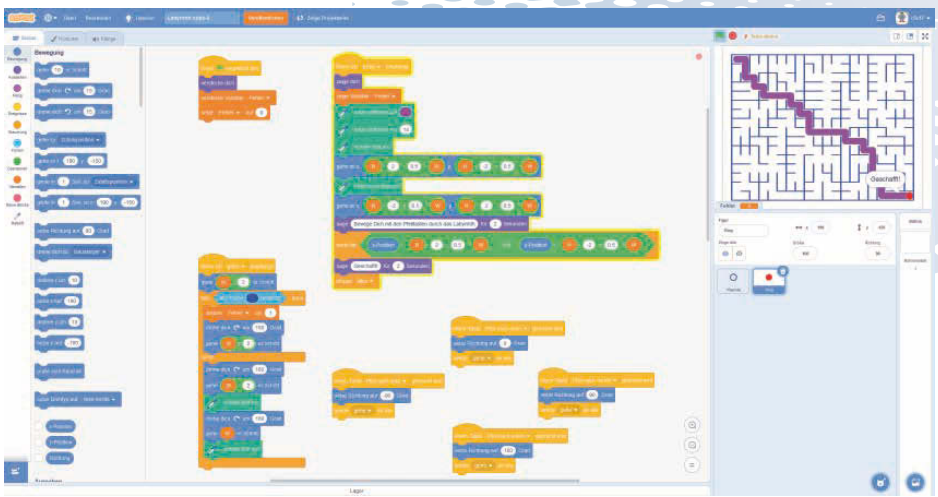
24 Ist das der Fall, ist die Figur auf dem falschen Weg. Die Variable ***Fehler*** wird um 1 hochgezählt, die Figur dreht sich um 180 Grad und geht den halben Schritt wieder zurück an ihren ursprünglichen Platz.

25 Im anderen Fall, wenn die Figur keine blaue Farbe berührt, also auf dem Weg zwischen zwei Rasterfeldern steht, soll sie auf das nächste Rasterfeld laufen und dabei eine Spur hinterlassen. Da sie auf dem ersten halben Schritt keine Spur hinterlassen hat, lasse die Figur den halben Schritt wieder zurückgehen, wie das auch beim Berühren der Mauer passiert, und schalte dann, am ursprünglichen Platz angekommen, den Stift ein.

26 Drehe die Richtung erneut um 180 Grad und bewege die Figur mit eingeschaltetem Stift einen ganzen Schritt auf das nächste Rasterfeld. Schalte dort den Stift wieder aus.

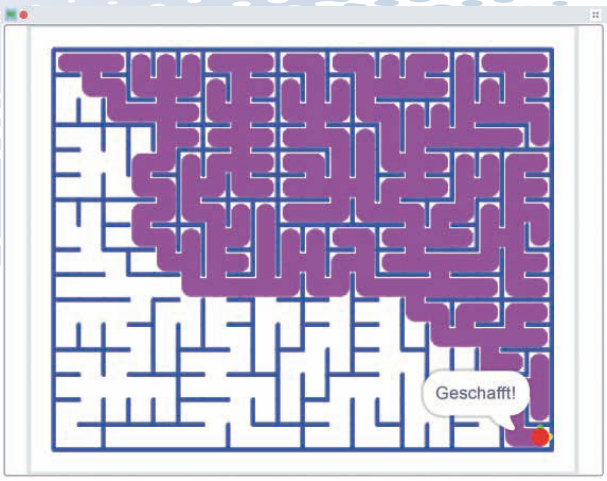
27 Auf dem nächsten Feld angekommen, bleibt die Figur stehen, bis du wieder eine Taste drückst, die eine neue Bewegungsrichtung vorgibt und den Programmblock wieder ausführt.

Starte das Programm mit einem Klick auf das grüne Fähnchen. Warte, bis das Labyrinth fertig gezeichnet ist und die Aufforderung zum Loslaufen erscheint. Bewege dich mit den Pfeiltasten, aber nicht zu hektisch. Zurückzugehen ist erlaubt, gegen die Wand zu laufen, bringt böse Strafpunkte.



AUTOMATISCH DEN WEG DURCH DAS LABYRINTH FINDEN

Den Weg durch ein Labyrinth zu finden macht Spaß. Für Programmierer ist es natürlich deutlich interessanter, das Programm den Weg finden zu lassen.



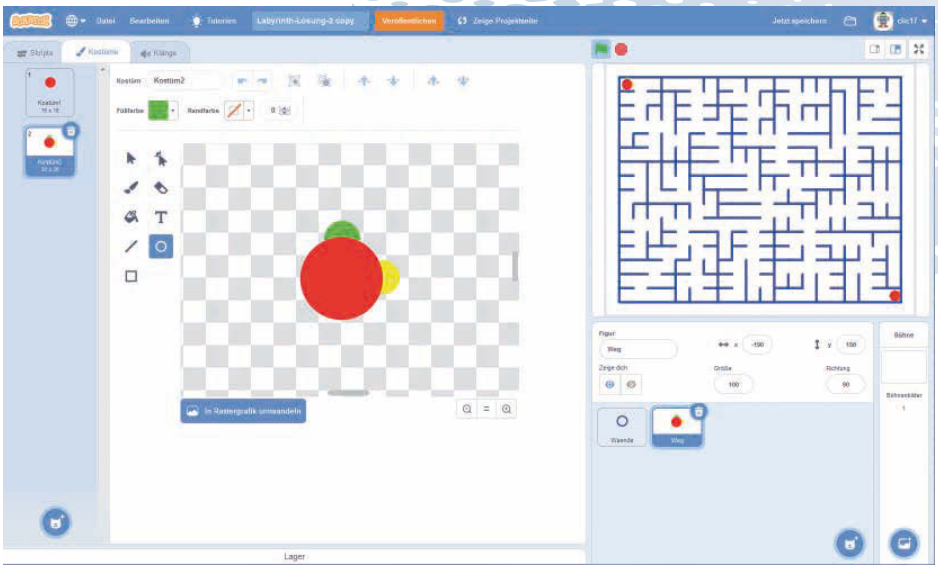
Für jedes Labyrinth, das keine Inseln, sondern einen eindeutigen Weg hat, gibt es eine sichere Lösung: Gehe immer mit der linken Hand an der Wand entlang, dann bleibt die rechte frei für eine Taschenlampe.

Nach diesem Prinzip wird die nächste Version des Scratch-Programms einen Weg durch das Labyrinth finden.

1 Die rote Figur bekommt zur Suche nach dem Weg zwei Sensoren, ein gelbes „Auge“, das noch vorne blickt, und eine grüne „Hand“, die sich an der Wand entlangtastet. Dupliziere dazu für die Figur ein zweites Kostüm und füge diesem, wie auf der Abbildung zu sehen, zwei farbige Kreise hinzu. Diese kannst du in der Reihenfolge der Ebenen nach hinten schieben. Es ist nur wichtig, dass sie weit genug aus dem roten Kreis herausragen, um benachbarte Wände zu berühren.

2 Das Skript zum Zeichnen des Labyrinths bleibt unverändert, das Skript für die rote Figur bekommt die neuen Funktionen, um den Weg zu finden. Lösche die Programmblöcke, die mit ***Wenn Taste ... gedrückt*** und ***Wenn ich gehe empfange*** beginnen.

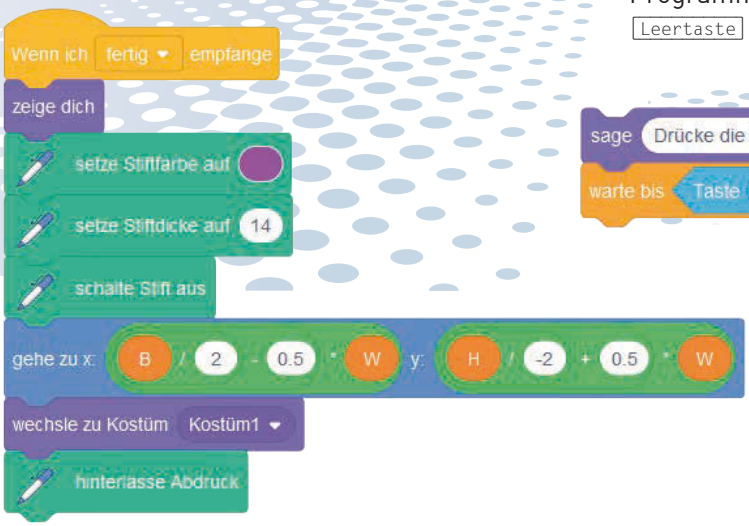
3 Lösche auch die Variable ***Fehler***. Du wirst sie nicht mehr brauchen, das Programm macht keine Fehler. Lösche auch unter dem Block ***Wenn grünes Fähnchen angeklickt*** die Blöcke, die diese



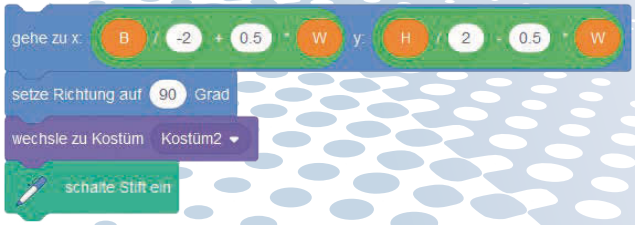
10 Labyrinth

Variable betreffen. An dieser Stelle wird jetzt nur noch die rote Figur versteckt.

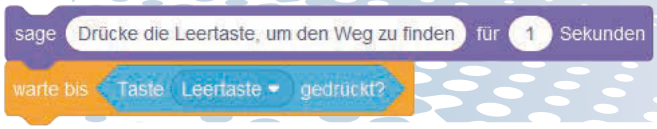
4 Wenn das Labyrinth fertig gezeichnet ist und die rote Figur erscheint, braucht das Programm ein paar Änderungen. Entferne hier den Block mit der Variablen ***Fehler***. Füge dafür, bevor die Figur unten rechts am Ziel den Abdruck hinterlässt, einen Block ***wechsle zu Kostüm Kostüm1*** ein. Der Abdruck soll einfach nur ein roter Kreis ohne die beiden Sensoren sein.



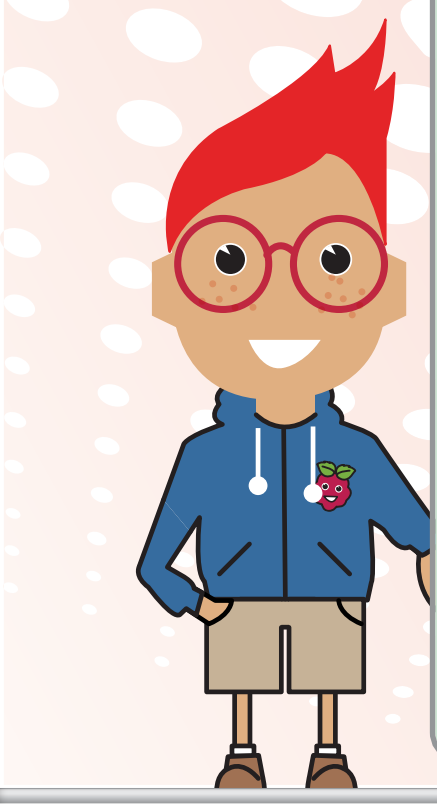
5 Jetzt wird die Figur an den Start oben links bewegt. Danach wird die Richtung auf 90 Grad (nach rechts) gesetzt, und ***Kostüm2*** wird ausgewählt, bei dem die beiden Sensoren sichtbar sind. Außerdem wird der Stift eingeschaltet, damit die Figur auf ihrem Weg eine Spur hinterlässt.



6 Jetzt ist die rote Figur bereit, auf den Weg geschickt zu werden. Ein Block ***sage ...*** teilt dem Benutzer mit, dass er die Leertaste drücken muss, um zu starten. Anschließend wartet das Programm darauf, dass der Benutzer die Leertaste auch wirklich drückt.



7 Danach startet eine neue ***wiederhole bis ...***-Schleife, die die Bewegungsschritte der roten Figur so lange wiederholt, bis diese in der rechten unteren Ecke des Labyrinths angekommen ist.

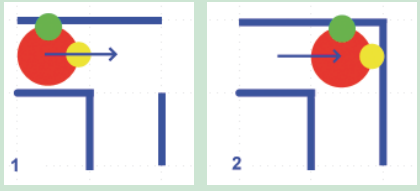


SO SUCHT DAS PROGRAMM SEINEN WEG

Bevor wir das Programm bauen, das den Weg findet, hier die Logik, nach der die Suche im Labyrinth funktioniert.

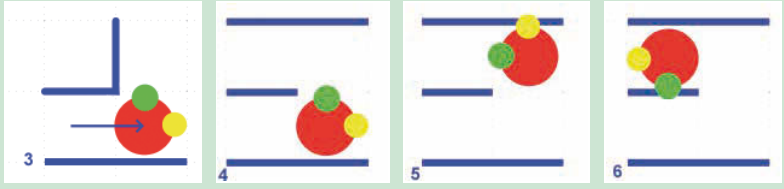
Wenn sich die Figur durch das Labyrinth bewegt, kann es zwei Fälle geben:

1. Die Figur steht auf einem Feld, auf dem der grüne Sensor links eine Wand berührt.



Im ersten Fall muss das Programm noch unterscheiden, ob die Figur vor einem freien Feld steht (1) und geradeaus weitergehen kann oder ob sie vor einer Wand steht (2) und nach rechts abbiegen muss – links ist ja die Wand.

2. Die Figur ist einen Schritt auf ein Feld gegangen, auf dem der grüne Sensor links keine Wand mehr berührt.



Im zweiten Fall, wenn links keine Wand ist, muss sie immer links abbiegen (3).

Dies gilt auch für den Fall, dass die Figur nicht an einer Ecke, sondern neben einem Wandende (4) steht. Auch in diesem Fall kann sie links abbiegen und einen Schritt weitergehen. Danach steht sie vor der gleichen Situation: links ist keine Wand (5). Also soll die Figur sich wieder nach links drehen und dann noch mal einen Schritt weitergehen. Jetzt steht sie mit dem grünen Sensor wieder an einer Wand (6).

8 Diese Bewegungsregeln werden in zwei ineinandergeschachtelten ***falls ... dann ... sonst***-Abfragen verarbeitet.

9 Falls der grüne Sensor eine blaue Wand berührt, wird geprüft, ob auch der gelbe Sensor eine blaue Wand berührt.

10 Ist das der Fall, steht die Figur in einer Ecke, in der es nur nach rechts weitergeht, sie muss sich also um 90 Grad nach rechts drehen.

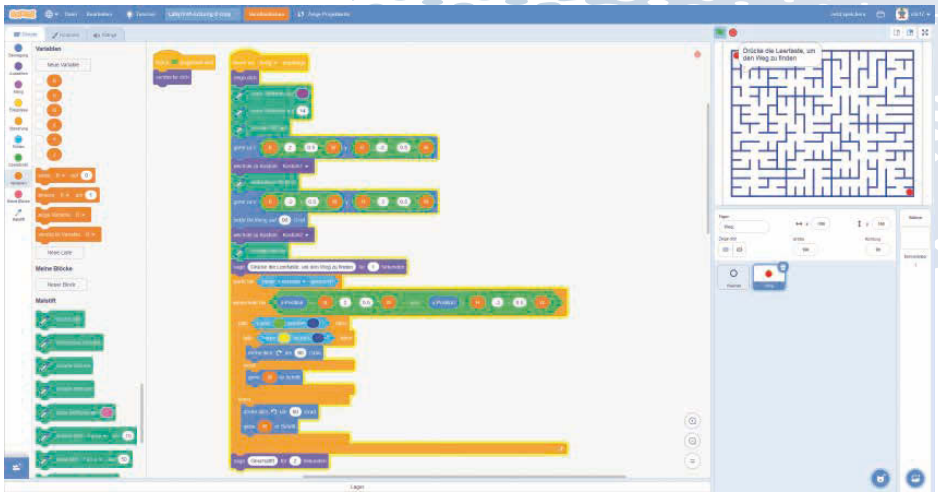
11 Berührt nur der grüne Sensor eine Wand, der gelbe aber nicht, geht es geradeaus weiter. Die Figur macht einen eine Rastereinheit ***W*** großen Schritt.

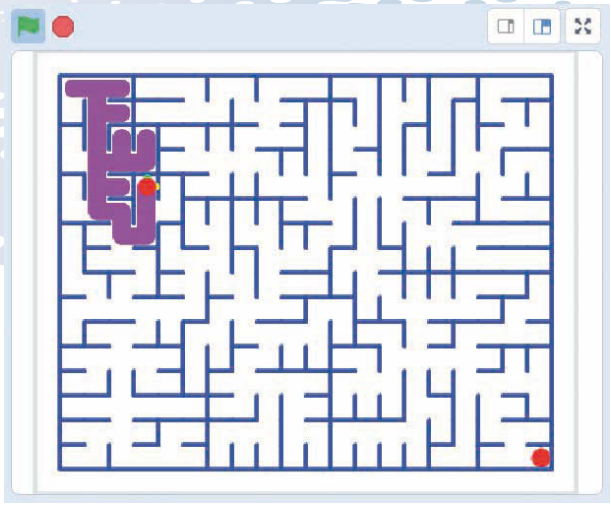
12 Berührt der grüne Sensor keine Wand, muss die Figur nach links abbiegen und einen Schritt gehen, um wieder eine Wand zu finden.

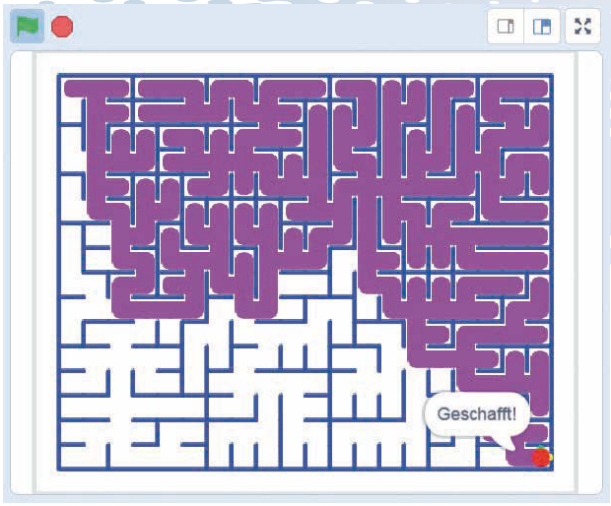
13 Diese Bewegungsmuster werden so lange wiederholt, bis die Figur den Zielpunkt in der rechten unteren Ecke erreicht hat. Dann wird noch kurz eine Meldung angezeigt, und das Programm ist zu Ende.

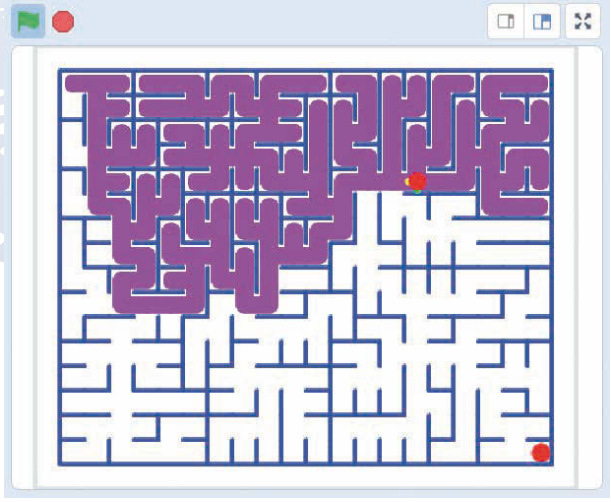
Starte das Programm mit einem Klick auf das grüne Fähnchen. Warte, bis das Labyrinth fertig gezeichnet ist und die Aufforderung zum Loslaufen erscheint. Drücke dann einmal auf die Leertaste, und die rote Figur wird loslaufen.

Jetzt kannst du mitverfolgen, wie sich die Figur immer an der linken Wand entlanghangelt und so etwa die Hälfte des Labyrinths durchwandert, bis sie am Ziel ankommt.



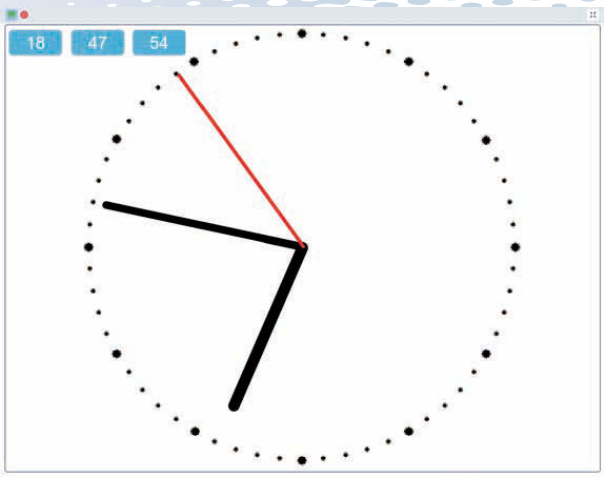



Geschafft!



11 Analoguhr

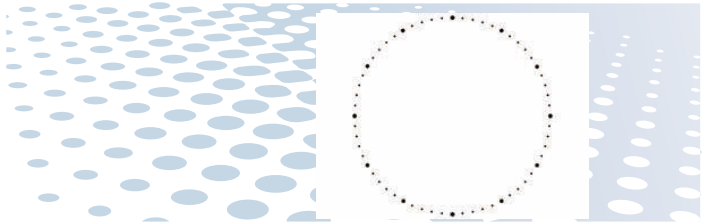
Die digitale Zeitanzeige, wie wir sie heute von Computern gewöhnt sind, kam erst in den 1970er-Jahren in Mode. Davor hatte man jahrhundertelang die Uhrzeit analog mit Zeigern auf einem Zifferblatt angezeigt. Der Digitaluhrboom ist in den letzten Jahren wieder etwas zurückgegangen, da man erkannt hat, dass Analoguhren schneller und bei schlechten Wetterbedingungen oder auf große Entfernungen, wie zum Beispiel auf Bahnhöfen, auch klarer abzulesen sind. Das menschliche Auge erfasst eine Grafik schneller als Ziffern oder Buchstaben. Das Bild einer Analoguhr prägt sich ins Kurzzeitgedächtnis ein, sodass man es, auch wenn man es nur unvollständig oder verschwommen gesehen hat, dennoch richtig umsetzen kann. Sieht man dagegen eine Digitaluhr nur ungenau, kann man daraus keine zuverlässigen Rückschlüsse auf die angezeigte Zeit ziehen.



Das Programm stellt eine Analoguhr mit Sekundenzeiger auf dem Bildschirm dar, die die aktuelle Uhrzeit anzeigt. Zusätzlich wird die Zeit auch noch digital angezeigt. Scratch bietet Blöcke, die das Rechnen mit Uhrzeiten sehr einfach machen.

Das Programm stellt die Uhr nur durch Bewegung der drei Zeiger dar. Die ***Malstift***-Erweiterung wird nicht benötigt.

1 Lege in Scratch ein neues Projekt an, klicke auf der Figurenpalette rechts außen auf die Bühne, fahre mit der Maus auf ***Bühnenbild wählen*** und klicke dann auf das Symbol ***Bühnenbild hochladen***. Lade das Hintergrundbild ***uhr.png*** aus dem Download. Es zeigt ein einfaches Zifferblatt einer Analoguhr ohne Zahlen im Stil einer Bahnhofsuhr.

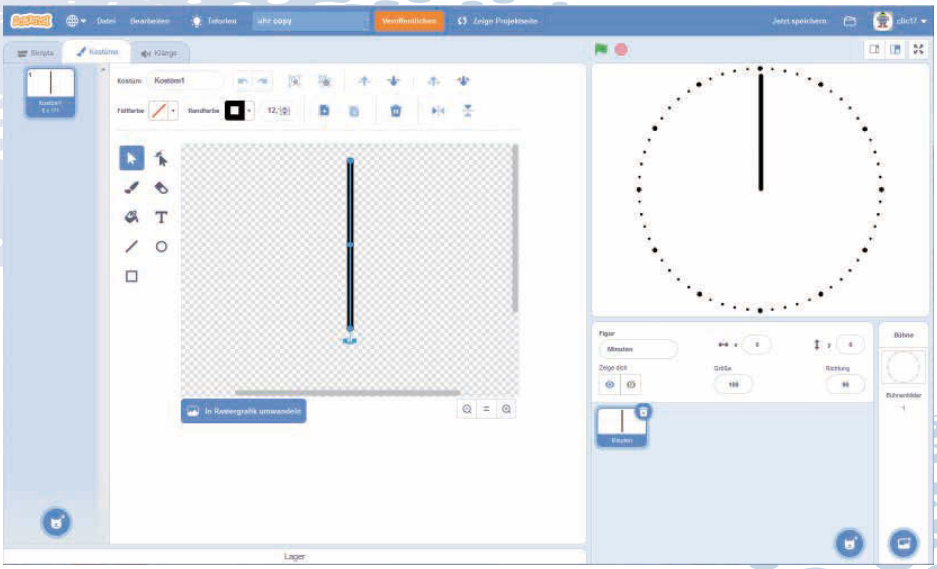


2 Lösche die Katze, fahre mit der Maus in der Figurenliste auf das Symbol ***Figur wählen*** und klicke dann auf das Symbol ***Figur malen***. Nenne diese Figur ***Minuten***, es wird der Minutenzeiger der Uhr.

3 Zeichne eine senkrechte Linie in der gewünschten Länge des Minutenzeigers. Das untere Ende dieser Linie muss genau auf dem Objektdrehpunkt liegen. Wenn du während des Zeichnens die Umschalt-Taste gedrückt hältst, wird die Linie exakt senkrecht.

4 Schiebe den Minutenzeiger genau auf den Mittelpunkt des Zifferblatts. Gib dazu auf der Figurenpalette in die beiden Felder ***x*** und ***y*** jeweils eine ***0*** ein.

5 Dupliziere den Minutenzeiger mit einem Rechtsklick auf die Figur in der Figurenliste. Nenne die neue Figur ***Stunden***. Mache im Bereich ***Kostüme*** die Strichstärke etwas dicker und verkürze den Zeiger dafür ein Stück. Das untere Ende



muss immer noch genau auf dem Objektdrehpunkt liegen.

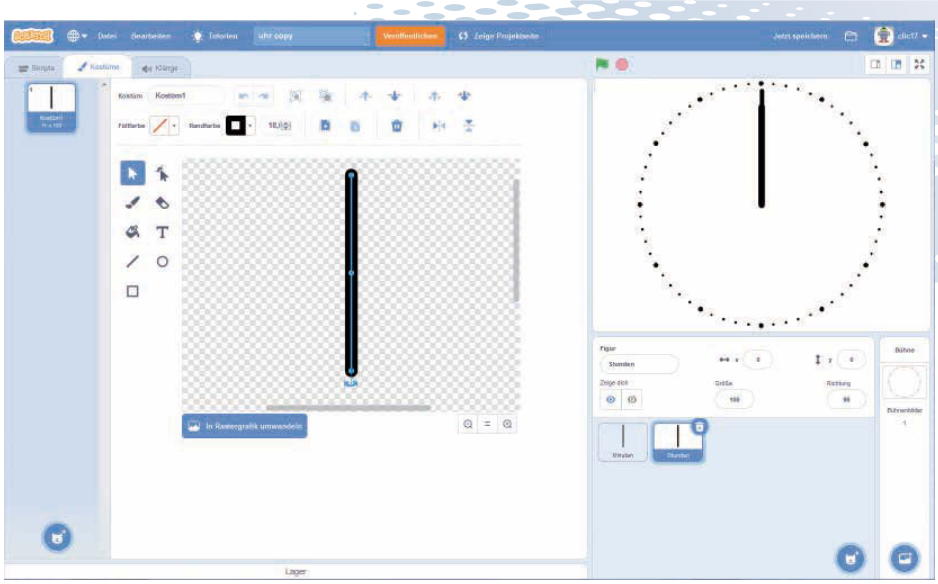
6 Schiebe diese Figur ebenfalls genau auf den Mittelpunkt des Zifferblatts.

7 Bevor du auch noch den Sekundenzeiger aus dem Minutenzeiger duplizierst, bauen wir das Programm zusammen. Das hat den Vorteil, dass das Skript für den Sekundenzeiger aus dem sehr ähnlichen Skript für den Minutenzeiger mit dupliziert werden kann.

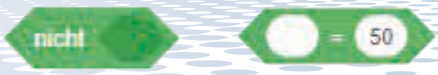
8 Als Erstes erstellen wir das Skript für den Minutenzeiger, der sich in jeder Minute um 6 Grad drehen soll. Das ergibt in 60 Minuten eine volle Umdrehung mit 360 Grad. Lege für das Programm auf der Blockpalette ***Variablen*** eine Variable ***M*** an.

9 Wenn der Benutzer auf das grüne Fähnchen klickt, soll eine Endlosschleife starten.

10 Einmal in der Minute soll der Minutenzeiger gedreht werden. Die Dauer eines Schleifendurchlaufs im Programm ist nicht bekannt. Für die Analoguhr bedeutet das, dass die Grafik nicht bei jedem Schleifendurchlauf aktualisiert werden



muss, sondern nur, wenn die aktuelle Minute eine andere ist als die zuletzt gezeichnete. Da Scratch keinen Block für ***ungleich*** kennt, verwenden wir eine Kombination der Blöcke ***nicht*** und ***gleich***.

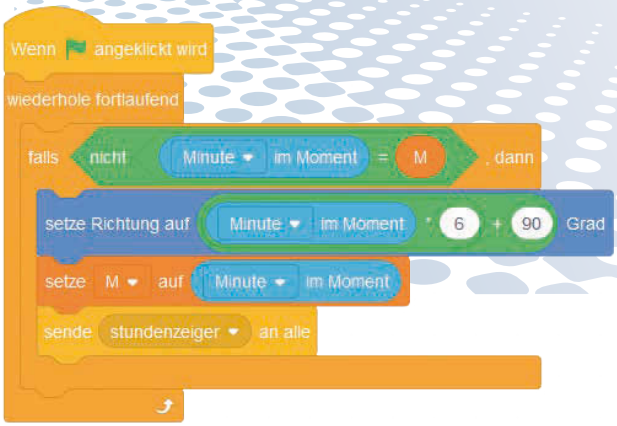


11 Der Block ***Minute im Moment*** liefert die aktuelle Minute aus der Systemuhr des Computers.

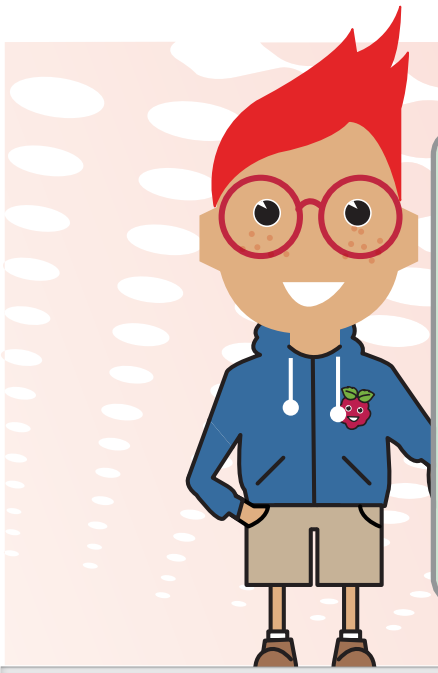
12 Falls die aktuelle Minute nicht der zuletzt dargestellten und in der Variablen ***M*** gespeicherten entspricht, muss der Minutenzeiger in die neue Richtung gedreht werden. Da er sich in jeder Minute um 6 Grad dreht, wird die aktuelle Minute mit 6 multipliziert. Dazu werden noch 90 Grad addiert, da die Standardrichtung jeder Figur bei Programmstart immer 90 Grad ist.

13 Anschließend wird die Variable ***M*** auf die aktuelle Minute gesetzt.

14 Damit sich auch der Stundenzeiger einmal in der Minute um einen kleinen Winkel bewegt, sendet das Skript noch eine Nachricht ***Stundenzeiger***, die von der Figur des Stundenzeigers ausgewertet wird.



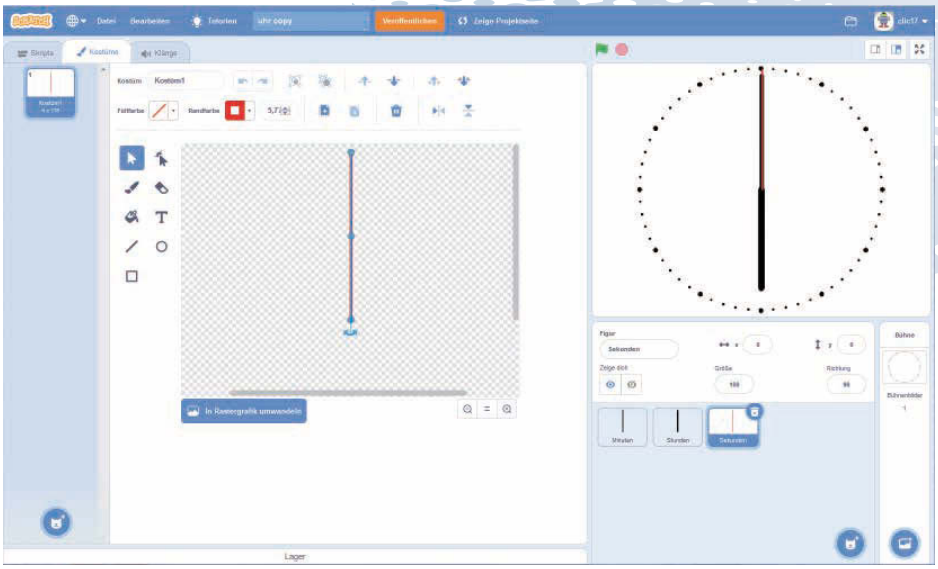
15 Wähle jetzt den Stundenzeiger. Er bekommt ein Skript, das startet, wenn die Nachricht ***Stundenzeiger*** empfangen wird. Der Stundenzeiger dreht sich in jeder ganzen Stunde um 30 Grad. Hinzu kommen noch 0,5 Grad pro Minute in der laufenden Stunde.



ZEITDARSTELLUNG BEI ANALOGUHREN

Je nach verwendeter Mechanik gibt es zwei unterschiedliche Anzeigemethoden bei Analoguhren. Bei echten analog laufenden Uhren führt der Minutenzeiger eine gleichförmige Kreisbewegung aus, bei digital gesteuerten Uhren, wie zum Beispiel Bahnhofsuhren, springt er zur vollen Minute um eine ganze Minute weiter. Letzteres Verfahren hat den Vorteil, dass die Uhrzeit besser auf einen Blick minutengenau abgelesen werden kann. Bruchteile von Minuten sind im Alltag normalerweise nicht wichtig. Wir verwenden für unser Uhrenprogramm ebenfalls dieses Verfahren. Der Stundenzeiger muss aber trotzdem eine gleichförmige Kreisbewegung ausführen. Es wäre sehr befremdlich und unübersichtlich, würde er jede volle Stunde um eine ganze Stunde weiterspringen.

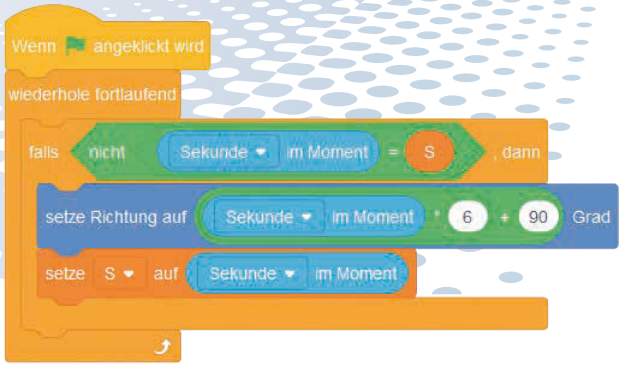
16 Dupliziere jetzt aus dem Minutenzeiger einen Sekundenzeiger. Ändere dessen Farbe auf Rot, mache die Strichstärke noch dünner und ziehe ihn etwas in die Länge. Schiebe auch diese Figur genau auf den Mittelpunkt des Zifferblatts.



17 Mit der Figur wird automatisch auch das Skript des Minutenzeigers für den Sekundenzeiger dupliziert. Die Skripte sind sich sehr ähnlich. Für den Sekundenzeiger sind nur kleine Veränderungen nötig. Lege zunächst auf der Blockpalette ***Variablen*** eine neue Variable ***S*** an, in der die Sekunde

eingetragen wird, in der sich der Sekundenzeiger zuletzt bewegt hat.

18 Wenn das grüne Fähnchen angeklickt wird, beginnt der Sekundenzeiger zu laufen. Das Skript zur Bewegung unterscheidet sich von dem des Minutenzeigers nur dadurch, dass die Bewegung auf der Sekunde im Moment basiert und nicht auf der Minute.



19 Setze auf der Blockpalette ***Fühlen*** das Häkchen links neben dem Block ***Minute im Moment***, um diesen Wert auf der Bühne anzuzeigen.

20 Wähle dort im Listenfeld nacheinander ***Stunde*** und ***Sekunde*** und schalte auch dann das Häkchen ein. Die Werte werden auf der Bühne angezeigt.

21 Klicke mit der rechten Maustaste nacheinander auf diese Anzeigefelder und schalte auf ***Großanzeige*** um. Ordne die drei Felder danach wie bei einer Digitaluhr nebeneinander an.

Die Digitaluhr zeigt nur den aktuellen Status von Systemvariablen an, benötigt also kein Skript. Deshalb läuft sie schon, bevor du auf das grüne Fähnchen klickst.

Klicke auf das grüne Fähnchen, um auch die Analoguhr zu starten. 

12 Simon – Senso – Einstein

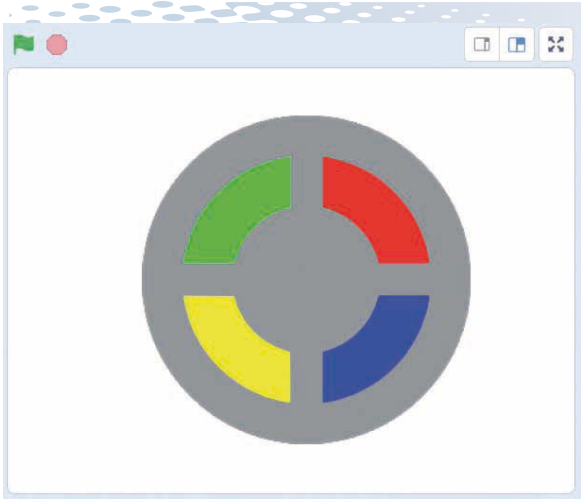
In den späten 1970er-Jahren, noch vor der Zeit echter Computerspiele, gab es ein elektronisches Spiel mit vier farbigen Lampen, das es im Jahr 1979 sogar auf die allererste Auswahlliste zum Spiel des Jahres schaffte. Das Spiel war in Deutschland unter dem Namen ***Senso*** auf dem Markt. Atari brachte einen Nachbau unter dem Namen ***Touch Me*** in der Größe eines damaligen Taschenrechners heraus. Ein weiterer Nachbau erschien als ***Einstein***, im englischen Sprachraum wurde das Spiel als ***Simon*** vermarktet.

Das Spielprinzip ist einfach. Farbige Flächen blinken in einer zufälligen Reihenfolge. Der Benutzer muss die gleiche Reihenfolge anschließend anklicken. Mit jeder Runde leuchtet eine weitere Farbfläche, sodass es immer schwieriger wird, sich die Folge zu merken. Sobald man einen Fehler macht, ist das Spiel zu Ende. Wer zehn Runden schafft, hat gewonnen.

DIE GRAFIK

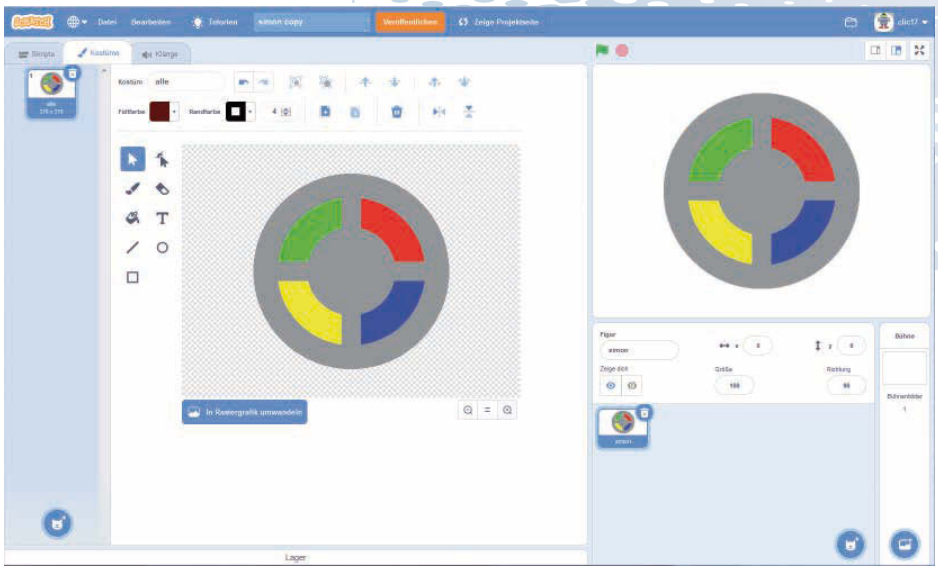
Die Grafik des Spiels besteht aus einer einzigen Figur, die verschiedene Kostüme verwendet, je nachdem, welche Farbe gerade leuchtet.

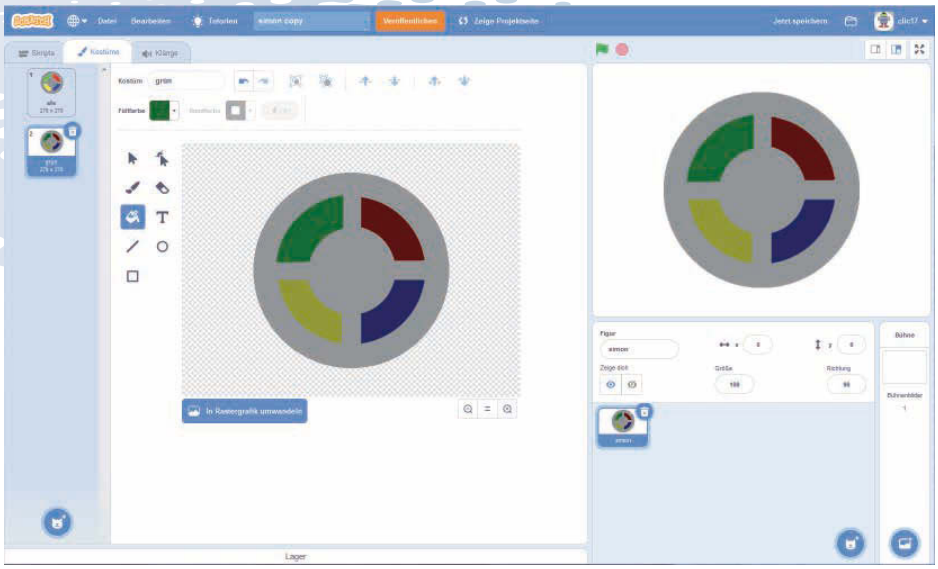
0 Starte ein neues Projekt in Scratch und lösche als Erstes die Katze. Wir verwenden als Figur eine stilisierte Darstellung des alten Elektronikspiels. Fahre mit der Maus auf das Symbol ***Figur wählen*** auf der Figurenpalette, klicke auf ***Figur hochladen*** und lade die Grafik ***simon.svg*** in das Projekt.



2 Schiebe die Figur genau auf den Mittelpunkt der Bühne. Gib dazu auf der Figurenpalette in die beiden Felder ***x*** und ***y*** jeweils eine ***0*** ein.

3 Schalte danach auf die Registerkarte ***Kostüme*** um und schiebe dort die Figur genau auf das kleine Kreuz in der Mitte. Der Drehpunkt, auch als Objektmittelpunkt bezeichnet, muss genau in der Mitte der Figur liegen.





Farbfläche wieder mit einem kräftigen Grün, Rot, Gelb oder Blau, wie in der Abbildung gezeigt.

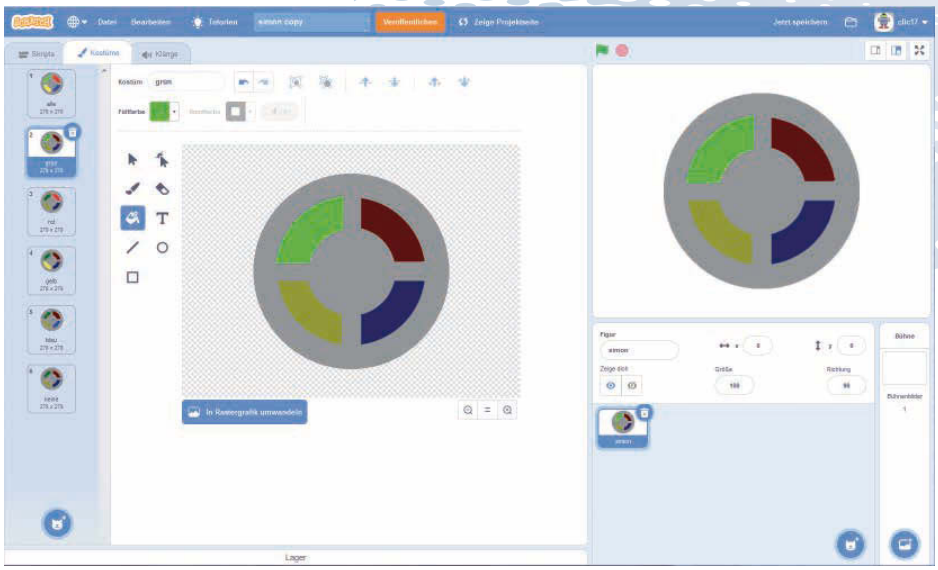
7 Damit klarer wird, welches Kostüm welche Farben leuchten lässt, benenne die Kostüme über das Namensfeld oben links wie in der Abbildung um: ***alle***, ***grün***, ***rot***, ***gelb***, ***blau***, ***keine***.

4 Klicke mit der rechten Maustaste auf das Kostüm ***simon*** in der Spalte der Kostüme und dupliziere es. Wähle in der Farbpalette den dunkelsten Grünton und fülle mit dem Farbeimersymbol damit die grüne Fläche im neuen Kostüm ***simon2***. Auf diese Weise wird im Spiel dargestellt, dass das Licht nicht leuchtet.

8 Installiere außerdem die Erweiterung ***Musik***. Wir brauchen sie, um die typischen Töne des Spiels abzuspielen.

5 Wähle nacheinander noch den jeweils dunkelsten Rot-, Gelb- und Blauton und fülle die entsprechenden Flächen.

6 Dupliziere das so veränderte Kostüm noch viermal. Du hast jetzt die Kostüme ***simon*** bis ***simon6***. Bei den Kostümen ***simon2*** bis ***simon5*** soll jeweils eine Farbe leuchten. Fülle dazu in jedem dieser Kostüme eine



12 Simon – Senso – Einstein

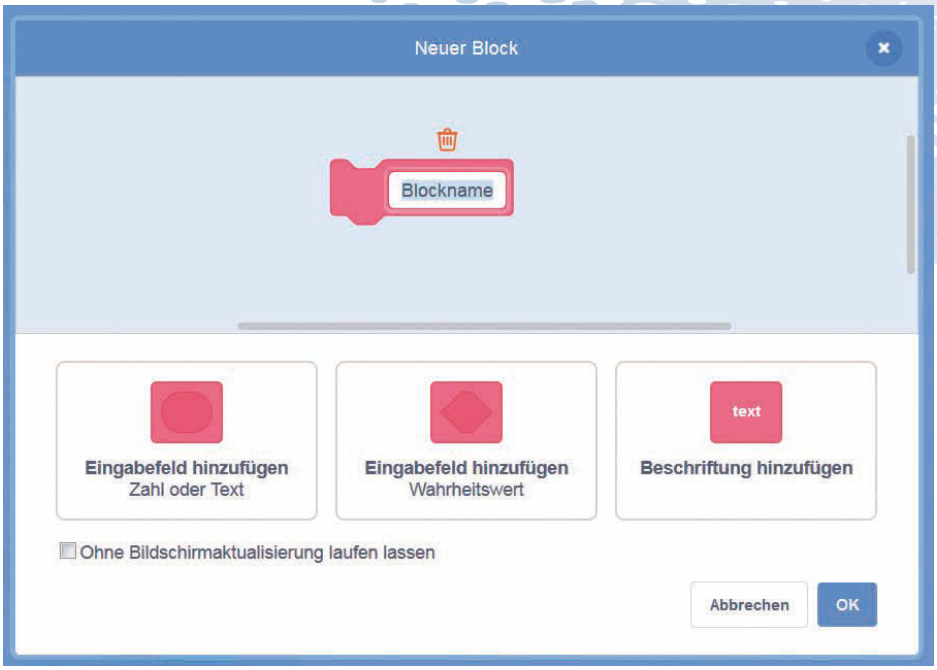
EIGENEN BLOCK DEFINIEREN

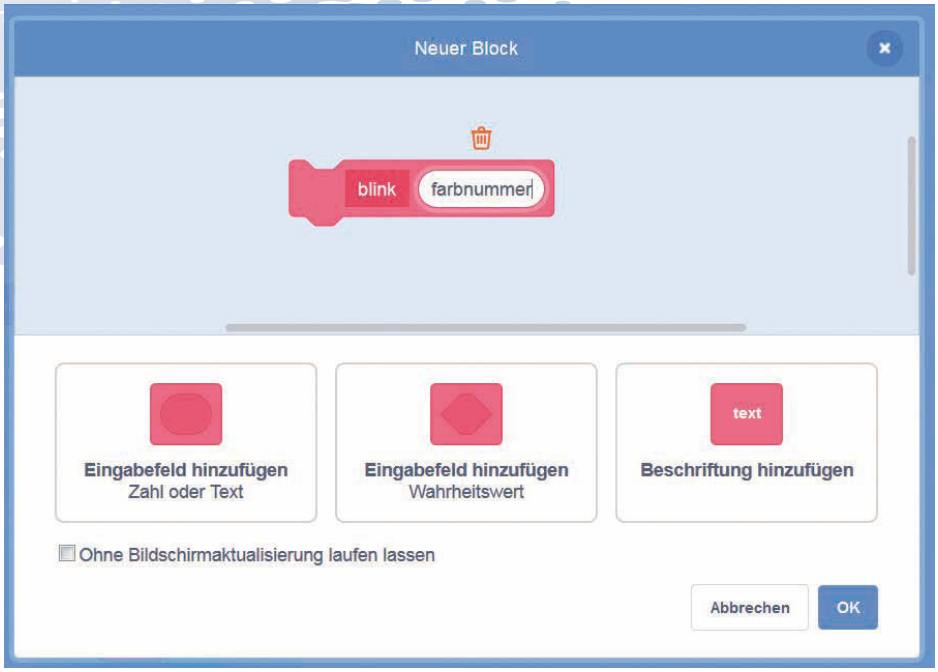
Eine wichtige Neuerung in Scratch 2.0 gegenüber Scratch 1.x ist die Möglichkeit, selbst eigene Blöcke zu definieren, die Aktionen, die im Programm immer wieder vorkommen, abarbeiten. In Scratch 3.0 funktioniert das natürlich auch. Durch Verwendung solcher Blöcke braucht man nicht ein und denselben Programmteil mehrfach an verschiedene Stellen zu duplizieren. Andere Programmiersprachen nennen dieses Verfahren ***Funktionen*** oder ***Prozeduren***.

1 In unserem Spiel wird es immer wieder vorkommen, dass eine der vier Lampen kurz aufleuchtet und dazu jeweils ein charakteristischer Ton abgespielt wird. Dies soll ein eigener Block erledigen. Klicke dazu auf der Blockpalette ***Meine Blöcke*** auf die Schaltfläche ***Neuer Block***.

2 Es erscheint ein Fenster mit einem roten Blocksymbol. Klicke in das Feld ***Blockname*** und gib dem Block den Namen ***blink***. Klicke dann auf ***Eingabefeld hinzufügen Zahl oder Text*** und füge ein Zahlenfeld hinzu. Das Programm soll später dem Block eine Zahl von ***1*** bis ***4*** übergeben, die angibt, welche der vier farbigen Flächen leuchten soll. Ändere den vorgegebenen Feldnamen ***number or text*** in ***farbnummer***, was dem zukünftigen Inhalt dieses Felds eher entspricht.

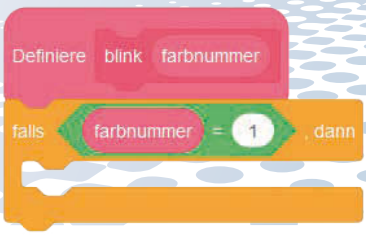
3 Klicke auf ***OK***, damit der Block angelegt wird. Im Skriptfenster erscheint die Blockdefinition mit einer Variablen ***farbnummer***, die innerhalb des neuen Blocks verwendet werden kann. Auf der Blockpalette ***Meine Blöcke*** erscheint





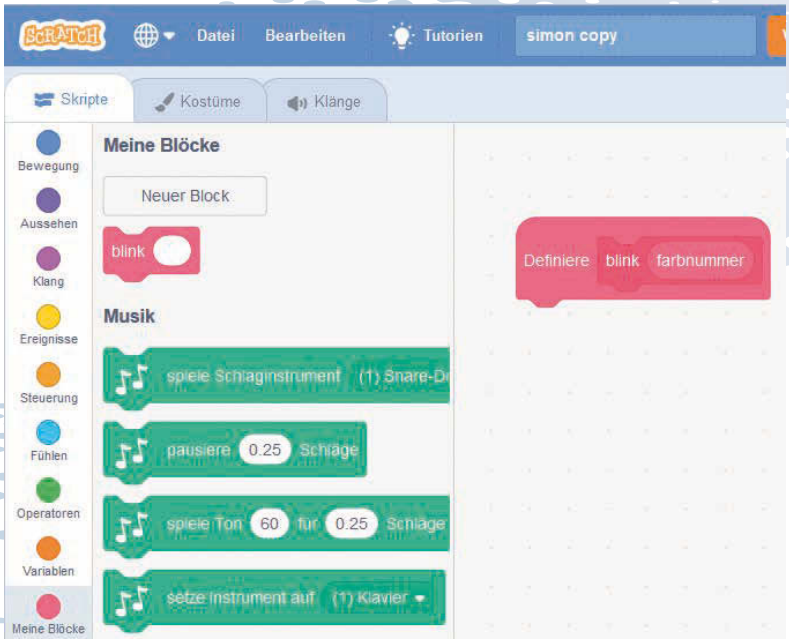
der Block, damit du ihn später in das Skript an die gewünschte Stelle ziehen kannst.

4 Hänge unten an die Blockdefinition die Programmblöcke, die der selbst definierte Block verarbeiten soll. In unserem Fall soll eine Farbfläche leuchten, was durch die Auswahl eines anderen Kostüms erledigt wird, und ein Ton soll abgespielt werden. Welche Farbe und welcher Ton, das hängt von der Zahl ab, die in der Variablen ***farbnummer*** übergeben wird. Hänge dazu als Erstes einen ***falls ... dann***-Block an, der prüft, ob ***farbnummer = 1*** ist.

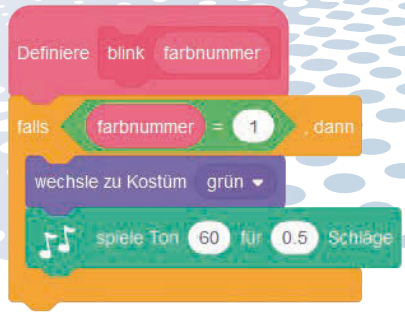


5 Um die Variable ***farbnummer*** in den ***gleich***-Operator einzubauen, ziehe sie einfach aus der Blockdefinition in das erste Feld des grünen Operatorblocks.

6 Im Fall von Farbe ***1*** soll die grüne Farbfläche links oben leuchten, und Ton ***60*** soll abgespielt werden.



12 Simon – Senso – Einstein



Ziehe dazu einen Block ***wechsle zu Kostüm ...*** in die ***falls ... dann***-Abfrage und wähle dort das Kostüm ***grün***. Ziehe zusätzlich noch einen Block ***spiele Ton ...*** in die Abfrage.

7 Dupliziere den ganzen ***falls ... dann***-Block noch dreimal und ändere in der Abfrage den Wert jeweils auf ***2***, ***3*** und ***4***, um die anderen möglichen Zahlen auszuwerten.

8 Wähle in den ***wechsle zu Kostüm ...***-Blöcken die passenden Kostüme ***rot***, ***gelb*** und ***blau***.

9 Wähle in den ***spiele Ton ...***-Blöcken die Töne ***64***, ***67*** und ***71***.

10 Nach den vier Abfragen leuchtet in jedem Fall genau eine Farbfläche, und es war ein Ton zu hören. Jetzt soll das Programm eine halbe Sekunde warten und anschließend das Kostüm ***keine*** wählen, auf dem alle Farbflächen wieder dunkel sind.

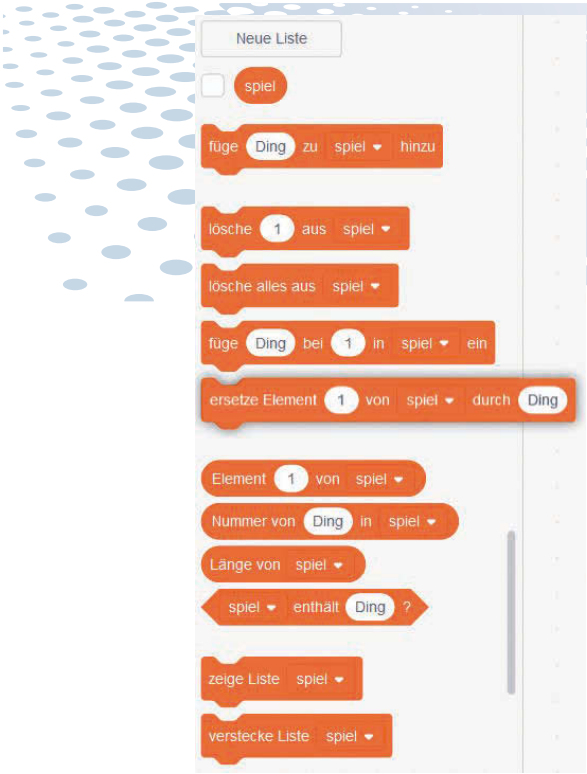
Dieser neu definierte Programmblock wird jedes Mal ablaufen, wenn eine Farbfläche leuchten soll. Dies passiert mehrfach hintereinander, wenn das Spiel dem Spieler eine Farbfolge vorspielt, die dieser nachspielen soll. Später wird der gleiche Programmblock verwendet, um die Farbflächen, die der Spieler anklickt, aufleuchten zu lassen. Auch dabei sollen die Töne zu hören sein.

DAS SPIEL

Das Spiel soll Farbfolgen von bis zu zehn Farben liefern, wobei mit einer Farbe begonnen wird und bei jedem Durchlauf die Folge um eine Farbe länger wird. Nachdem die Farbfolge abgespielt wurde, muss der Benutzer sie nachspielen. Bei einem Fehler ist das Spiel sofort zu Ende.

1 Lege zuerst auf der Blockpalette ***Variablen*** zwei Variablen ***farbe*** und ***i*** an. Beide sollen auf der Bühne nicht zu sehen sein. ***farbe*** enthält die Farbe, die der Benutzer angeklickt hat: Grün = ***1***, Rot = ***2***, Gelb = ***3***, Blau = ***4***. ***i*** wird, wie unter Programmierern üblich, für Schleifenzähler verwendet.

2 Zum Speichern der Farbfolge im Spiel verwenden wir eine Liste. Lege auf der Blockpalette ***Variablen*** mit der Schaltfläche ***Neue Liste*** eine neue Liste mit Namen ***spiel*** an. Sie soll auf der Bühne nicht zu sehen sein, da der Spieler sonst jederzeit die korrekte Farbfolge zur Lösung des Spiels ablesen könnte. Nachdem du die Liste angelegt hast, erscheinen einige neue Blöcke auf der Blockpalette, mit denen die Werte der Liste bearbeitet werden können.



LISTEN IN SCRATCH

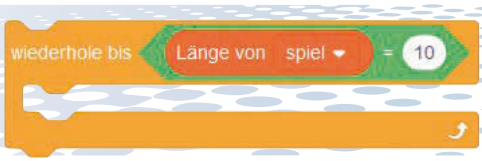
Listen sind eine Sonderform von Variablen, die mehrere Werte in einer festen Reihenfolge enthalten können. Listen können beliebig lang sein.

3 Wenn du auf das grüne Fähnchen klickst, soll ein neues Spiel beginnen. Dabei müssen zuerst alle Werte der alten Farbfolge aus der Liste ***spiel*** gelöscht werden. Mit dem Block ***lösche alles aus ...*** kannst du alle Werte aus einer Liste auf einmal löschen.

4 Danach startet die Hauptschleife des Spiels. In jedem Durchlauf wird die Farbfolge um eine Farbe verlängert. Das Spiel soll so oft laufen, bis diese Farbfolge und damit die Liste zehn Ele-

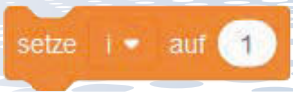
12 Simon – Senso – Einstein

mente enthalten. Schaffst du es, diese Farbfolge richtig nachzuspielen, hast du das Spiel gewonnen. Der Block ***Länge von ...*** liefert die aktuelle Anzahl von Elementen in einer Liste.



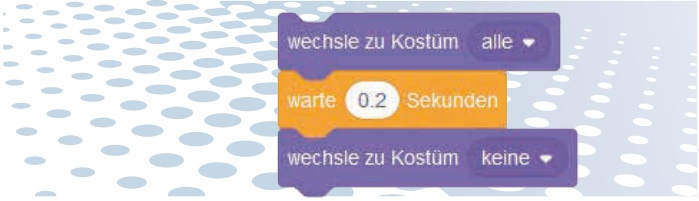
5 Beim Start jeder Runde wird der Liste eine Zufallszahl von 1 bis 4 hinzugefügt. Diese Zahl steht für die jeweilige Farbe, die später über den selbst definierten Block ***blink*** angezeigt wird.

6 Der Schleifenzähler ***i*** wird auf ***1*** gesetzt. Er wird an verschiedenen Stellen im Programm immer wieder gebraucht.



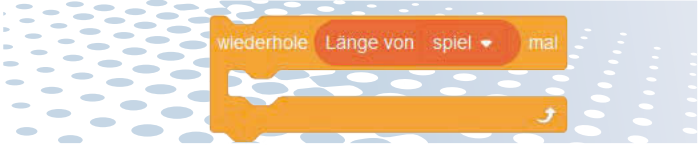
7 Jetzt folgt eine Schleife, die so oft durchläuft, wie die Liste gerade lang ist, wie viele Farben also abgespielt werden. Diese Schleife spielt die Farbfolge ab, die sich der Spieler merken muss. Beim ersten Mal zeigt sie nur eine einzige Farbe, beim zehnten Mal zehn Farben nacheinander. In jedem Durchlauf der Schleife wird der selbst definierte Block ***blink*** aufgerufen, und ihm wird ein Element der Liste übergeben. Dies ist immer eine Zahl von 1 bis 4. Danach wird der Zähler ***i*** um 1 erhöht. Den Block ***blink*** findest du auf der Blockpalette ***Meine Blöcke***, von dort kannst du ihn wie jeden anderen Block in das Programm ziehen.

8 Nachdem die Farbfolge abgespielt wurde, sollen einmal kurz alle Farben aufleuchten, um dem Spieler anzuzeigen, dass er jetzt an der Reihe ist, die Farbfolge nachzuspielen. Dazu wird das Kostüm mit allen leuchtenden Farben und nach einer kurzen Pause das Kostüm mit allen dunklen Farben angezeigt.



9 Danach wird der Schleifenzähler ***i*** erneut auf ***1*** gesetzt, da er für die folgende Schleife, die die Mausklicks des Spielers auswertet, wieder gebraucht wird.

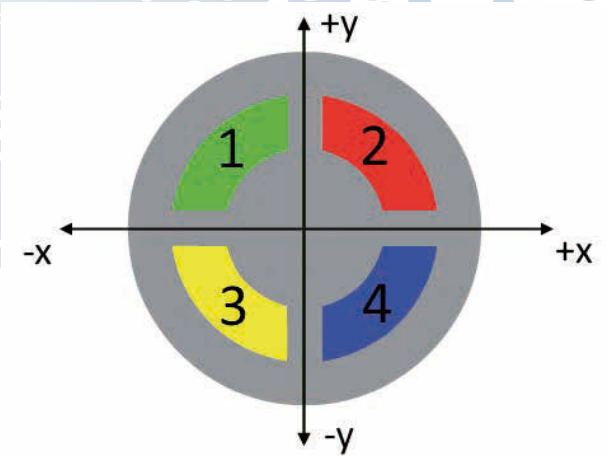
10 Auch diese Schleife wird so oft wiederholt, wie die Liste derzeit lang ist. Der Spieler muss genauso viele Farben nachspielen, wie er kurz vorher gesehen hat.



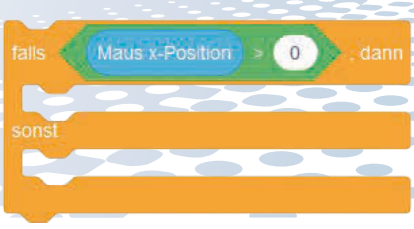
11 Da das Spiel nur eine einzige Figur enthält, kann nicht abgefragt werden, ob eine bestimmte Figur angeklickt wurde. Stattdessen

wertet das Spiel die Position des Mauszeigers aus. Dazu wartet das Skript zunächst, bis der Spieler die Maustaste gedrückt hat. Auf der Blockpalette ***Fühlen*** gibt es dafür den Block ***Maustaste gedrückt?***.

12 Da sich die Grafik genau in der Mitte der Bühne befindet, kann man einfach abfragen, ob x-Position und y-Position des Mauszeigers größer oder kleiner als 0 sind. Damit werden die vier Quadranten der Bühne und gleichzeitig die vier möglichen Farben eindeutig festgelegt.



13 Eine ***falls ..., dann ... sonst***-Abfrage unterscheidet anhand der ***Maus y-Position***, ob der Spieler in die obere oder die untere Hälfte geklickt hat.



14 Ist die ***Maus y-Position*** größer als 0, hat der Spieler in die obere Hälfte der Bühne geklickt. Eine weitere ***falls ..., dann ... sonst***-Abfrage entscheidet anhand der ***Maus x-Position***, ob die Farbe Grün = ***1*** oder Rot = ***2*** gewählt wurde.

15 Ist die ***Maus y-Position*** dagegen nicht größer als 0, hat der Spieler in die untere Hälfte der Bühne geklickt. Hier entscheidet eine ähnliche ***falls ..., dann ... sonst***-Abfrage anhand der ***Maus x-Position***, ob die Farbe Gelb = ***3*** oder Blau = ***4*** gewählt wurde. In allen Fällen wird die Variable ***farbe*** auf die vom Spieler ausgewählte Farbnummer gesetzt.

16 Nachdem der Spieler eine Farbe – oder genauer gesagt, einen Quadranten der Bühne – angeklickt hat, muss das Spiel auswerten, ob dies die richtige Farbe war, die dem jeweiligen Listenelement entspricht. Wenn ja, darf der Spieler weiterspielen, wenn nicht, ist das Spiel verloren. Auch hier wird eine ***falls ..., dann ... sonst***-Abfrage verwendet, die prüft, ob die Variable

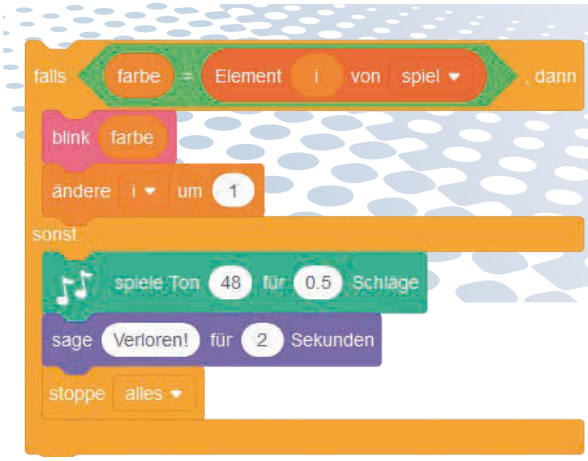
12 Simon – Senso – Einstein

farbe dem Listenelement ***i*** entspricht. Der Schleifenzähler ***i*** zählt auch hier wieder, wie zuvor beim Abspielen der Farbfolge, durch die ganze Liste.

17 Hat der Spieler die richtige Farbe angeklickt, wird der selbst definierte Block ***blink*** aufgerufen. Diesem wird der Wert der Variablen ***farbe*** übergeben, damit die angeklickte Farbe blinkt und der passende Ton ertönt.

18 Anschließend wird der Schleifenzähler ***i*** um 1 hochgezählt. Jetzt kann der Spieler die nächste Farbe anklicken.

19 Hat der Spieler aber eine falsche Farbe angeklickt – entspricht die Variable ***farbe*** also nicht dem Listenelement ***i*** –, ertönt ein tiefer Ton, es wird kurz die Meldung ***Verloren!*** angezeigt, und das Programm wird gestoppt.



20 Hast du es zehnmal erfolgreich geschafft, die komplette und immer länger werdende Farbfolge richtig nachzuspielen, ist die Hauptschleife beendet. In diesem Fall ertönt ein hoher Ton, und es wird die Meldung ***Gewonnen!*** angezeigt.

21 So sieht das Programm fertig aus. Je nach Bildschirmauflösung passt es möglicherweise, nicht mehr auf eine Bildschirmseite, ohne dass man scrollen muss.

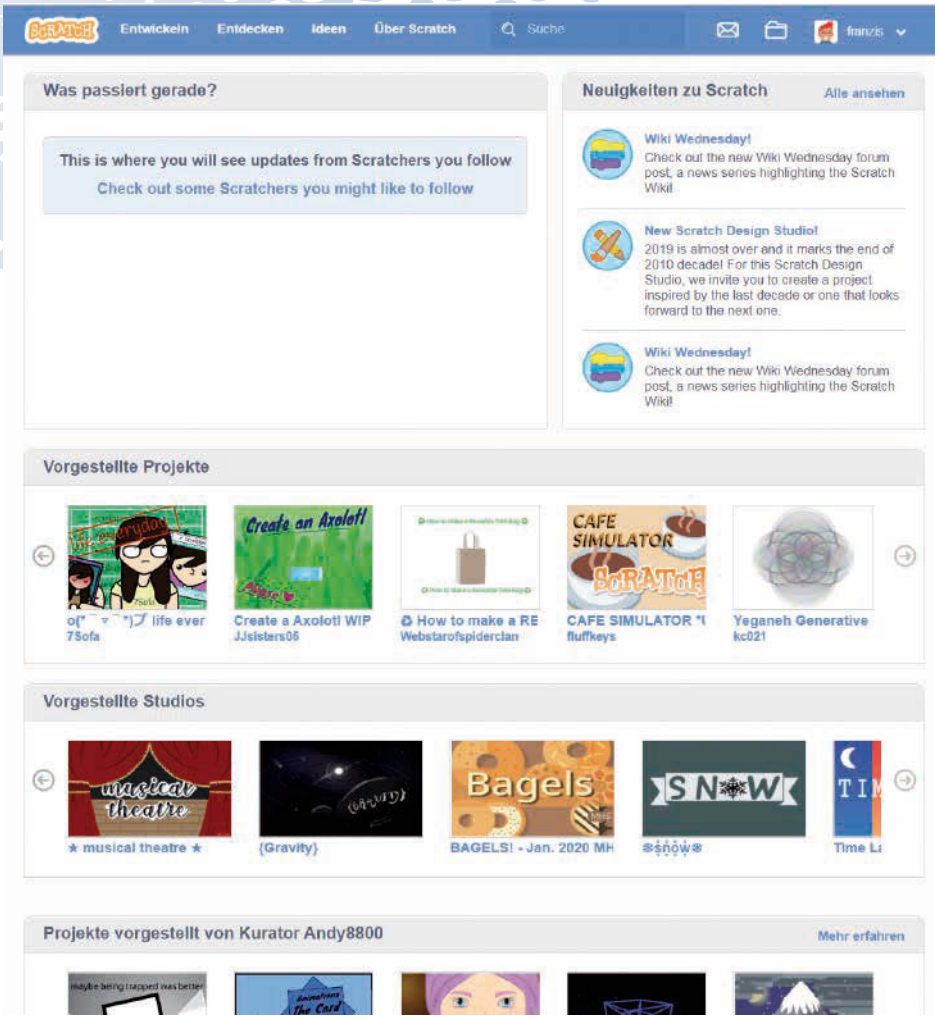
Klicke auf das grüne Fähnchen, um das Spiel zu starten. Es zeigt dir zunächst eine Farbe. Anschließend blinken einmal kurz alle Farben auf. Klicke auf diese Farbe. Danach erscheinen zwei Farben hintereinander, wobei die erste die zuvor angezeigte ist, die zweite eine zufällige. Wiederhole auch diese Farbfolge durch Klicken. Die Farbfolge wird jedes Mal länger, wobei die ersten Farben immer die gleichen sind wie beim letzten Mal. Nur die letzte Farbe ist eine zufällige neue. Versuche, die zehn Runden des Spiels zu überstehen, also die Farbfolgen fehlerfrei zu wiederholen.

13 Die Scratch-Gemeinschaft

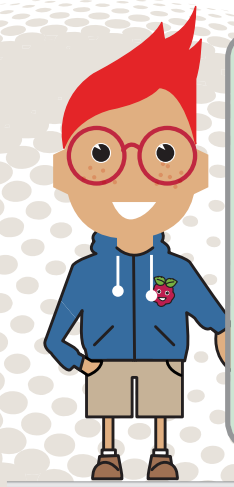
Programmieren ist nicht unbedingt nur etwas für diejenigen, die einsam in ihren Computerkellern sitzen. Scratch bietet eine riesige Onlinegemeinschaft, in der Menschen aus über 150 Ländern in mehr als 50 Sprachen Programme veröffentlichen und diese auch gegenseitig weiterentwickeln.

Wer ein Projekt entdeckt und dazu neue Ideen hat, kann es für sich kopieren und remixen. Das bedeutet verändern, verbessern, vereinfachen, übersetzen oder was auch immer. Auf diese Weise entstehen durch die kreativen Fähigkeiten vieler ganz neue Ideen. Jeden Tag werden weltweit über 225.000 neue Programme, Geschichten und Spiele mit Scratch entwickelt.

Melde dich über den Button ***Scratcher werden*** oben rechts auf der Scratch-Webseite ***scratch.mit.edu*** als neuer Scratcher an. Dazu musst du dir nur einen Benutzernamen und ein Passwort ausdenken. Gib im nächsten Schritt dein Geburtsdatum und dein Heimatland an. Diese Daten werden nur für statistische Zwecke verwendet und nicht veröffentlicht. Du erhältst eine E-Mail, die du mit einem Klick bestätigen musst. Damit wird verhindert, dass jemand automatisch massenhaft Benutzerkonten anlegt und irgendwelchen Spam verbreitet.



Jeden Monat treten der Scratch-Gemeinschaft ungefähr eine Million neue Menschen bei. Sie funktioniert deshalb so gut, weil sich alle an ein paar einfache Regeln halten. Lies dir diese Regeln durch und respektiere sie, damit du und alle anderen Spaß an Scratch haben.



DER SCRATCH-DAY

Der Scratch Day ist ein weltweites Netzwerk von Veranstaltungen zum Thema Scratch, das einmal im Jahr an einem bestimmten Tag stattfindet. Auf der Seite *day.scratch.mit.edu* findest du das nächste Datum und kannst nach Veranstaltungen in deiner Region suchen.

Klicke anschließend oben auf die Schaltfläche ***Entdecken***, um beliebte oder neu veröffentlichte Scratch-Projekte aus verschiedenen Kategorien zu finden. Du kannst jedes Projekt direkt ausprobieren und mit dem Button ***Schau hinein*** auch den Scratch-Code sehen und daran herumbasteln.

Regeln für Scratcher

Wir brauchen die Hilfe aller Scratcher, damit Scratch eine freundliche und kreative Gemeinschaft bleibt, in der sich Menschen mit unterschiedlichen Hintergründen und Interessen willkommen fühlen.

Sei respektvoll.
Wenn du Projekte oder Kommentare veröffentlichst denke daran dass das von Leuten verschiedenen Alters und verschiedenen kulturellem Hintergrund gesehen wird.

Sei konstruktiv.
Wenn du die Projekte andere kommentierst, schreibe was du daran magst und mache Verbesserungsvorschläge.

Veröffentlichen.
Es steht dir frei Projekte, Ideen, Bilder oder irgendetwas anderes was du auf Scratch findest zu remixen – und jeder kann alles verwenden was du veröffentlichst. Stelle sicher dass du die Vorarbeiten benennst wenn du etwas remixt.

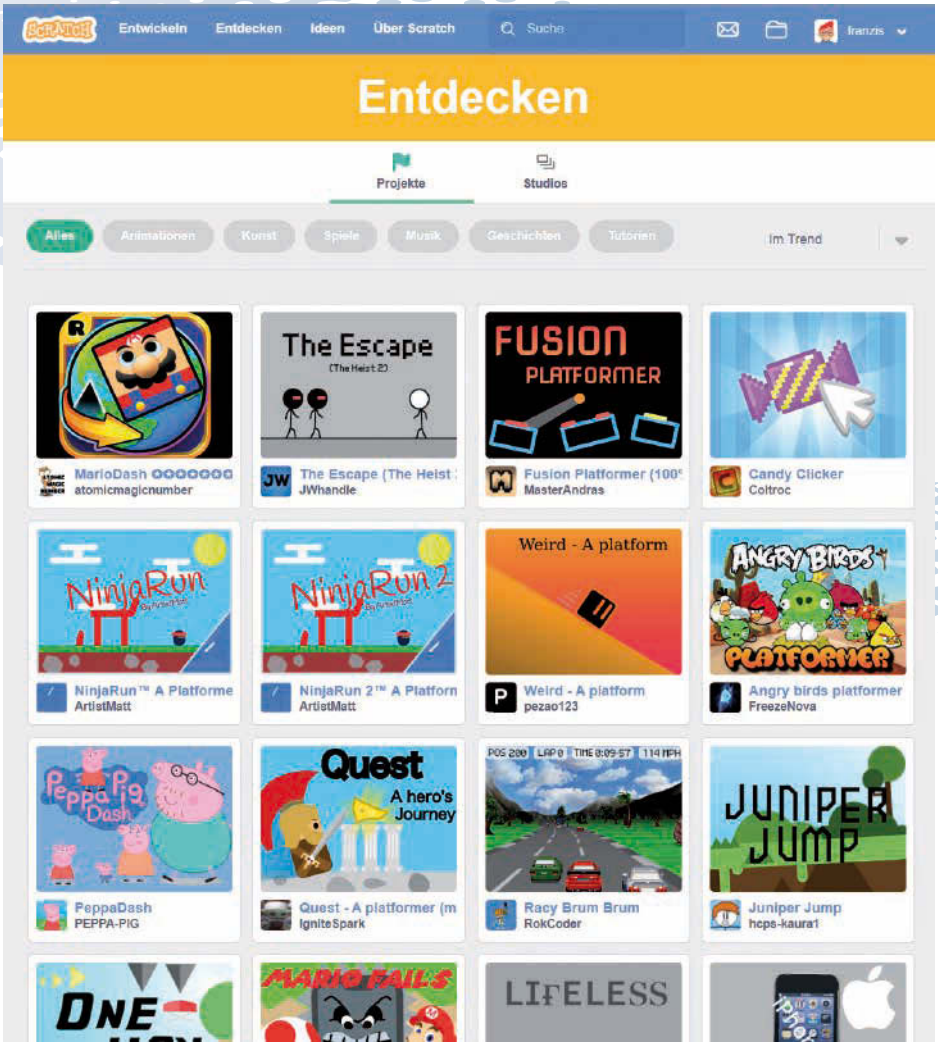
Halte persönliche Informationen geheim.
Gib aus Sicherheitsgründen niemals irgendwelche Informationen preis,die für private Kommunikation genutzt werden können - wie etwa echte Nachnamen, Telefonnummern, Adressen, E-Mail-Adressen, Verweise zu sozialen Netzwerken oder zu Webseiten mit unmoderiertem Chat.

Sei ehrlich.
Versuche nicht dich für andere Scratcher auszugeben, Gerüchte zu verbreiten oder die Community sonstwie auszutricksen.

Hilf mit die Seite freundlich zu halten.
Wenn du glaubst dass ein Projekt oder Kommentar gemein, beleidigend, zu gewalttätig oder sonstwie unpassend ist, klicke auf "Reporten" um uns darüber zu informieren.

Scratch heißt Menschen jeden Alters, jeder Herkunft, Ethnie, Religion, sexueller Orientierung und Geschlechteridentität willkommen.

13 Die Scratch-Gemeinschaft



kommentieren und zu remixen.

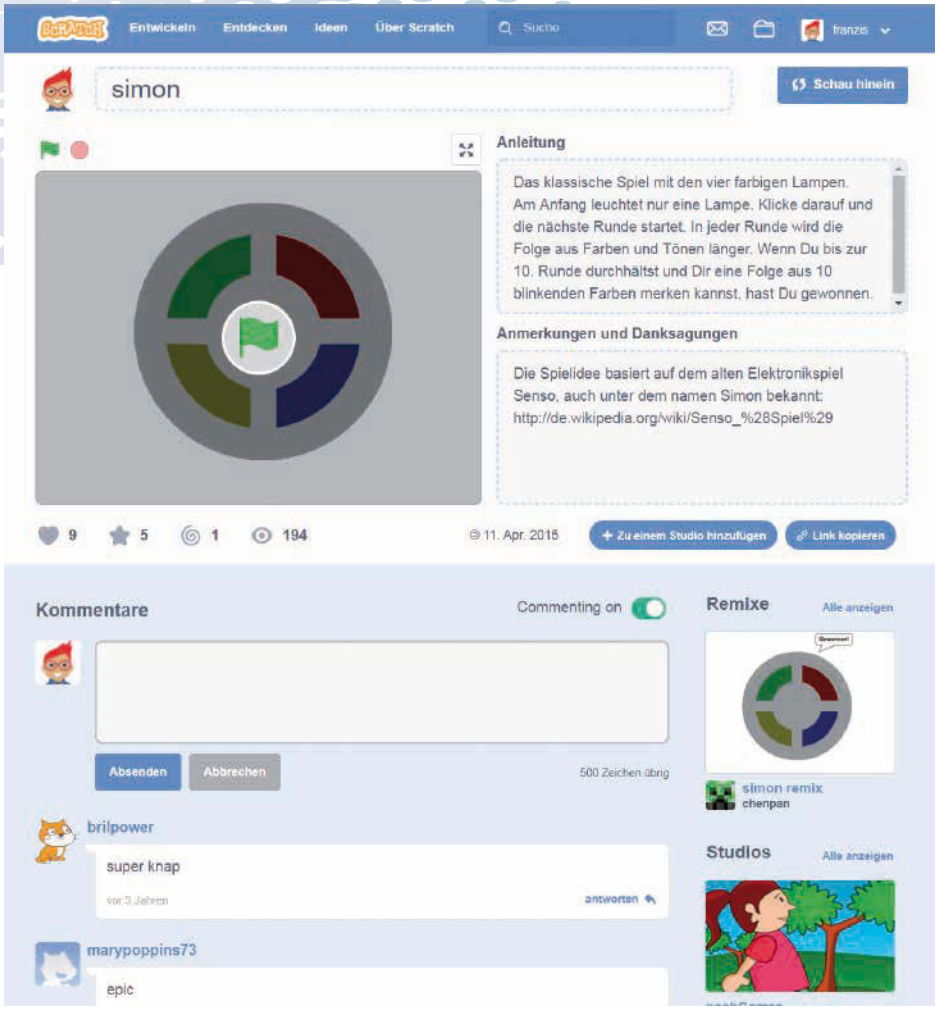
Wenn du an einem Projekt bastelst, wird es automatisch unter ***Meine Sachen*** gespeichert. Du brauchst dich um das Speichern nicht zu kümmern. Zu deiner Liste ***Meine Sachen*** kommst du jederzeit mit diesem Ordnersymbol oben rechts.

Das Projekt bleibt so lange nur für dich sichtbar, bis du es mit dem Button ***Veröffentlichen*** oben veröffentlichst.

Beim Veröffentlichen solltest du eine Kurzbeschreibung zu deinem Projekt hinterlegen, wenn nötig Bedienungshinweise für Benutzer geben, die mit dem Projekt nicht vertraut sind, sowie deine Quellen nennen. Damit sind andere Projekte gemeint, aus denen du Sachen oder Ideen übernommen hast.

EIGENE PROJEKTE VERÖFFENTLICHEN

Nachdem du angemeldet bist, kannst du eigene Projekte auf dem Scratch-Server ablegen und auch veröffentlichen. Außerdem hast du als angemeldeter Scratcher die Möglichkeit, die Projekte von anderen zu

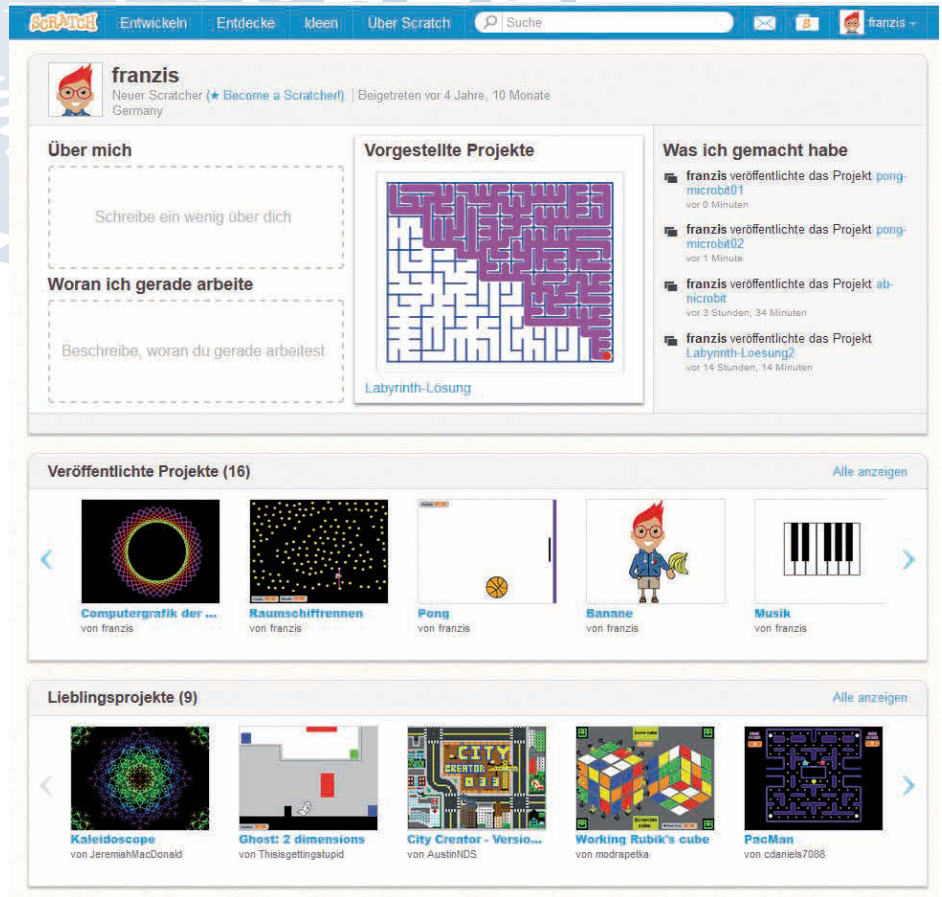


Möchtest du ein Projekt von jemand anderem remixen, öffne es einfach und klicke oben rechts auf ***Remixen***. Dann wird es bei deinen Projekten gespeichert. Wenn du ein remixtes Projekt veröffentlichst, denke daran, den ursprünglichen Autor zu erwähnen und in deine Beschreibung einzutragen, was du am Projekt bearbeitet hast. Das ist auch für den Autor interessant.

Auf deiner Profilseite sind deine veröffentlichten Projekte sowie deine letzten Kommentare zu anderen Projekten und die Projekte anderer Scratcher, die du mit dem Sternchen als Lieblingsprojekte markiert hast, zu sehen.

Unter dem Scratch-Benutzernamen ***franzis*** (***scratch.mit.edu/users/franzis***) haben wir die in diesem Buch verwendeten Programme veröffentlicht. Hier kannst du sie alle direkt ausprobieren und gern auch kommentieren, liken oder remixen.

13 Die Scratch-Gemeinschaft



PARK18

Park18 ist ein Schiebepuzzle, das auf der Idee des Spieleklassikers *Rush Hour* (*de.wikipedia.org/wiki/Rush_Hour_(Spiel)*), manchmal auch als *Traffic Jam* bezeichnet, basiert. Ziel ist es, ein Auto aus einem zugeparkten Parkplatz herauszuschieben. Dabei können alle Autos nur vorwärts und rückwärts bewegt werden, nicht seitlich. Klicke dazu einfach vorne oder hinten auf die Fahrzeuge.

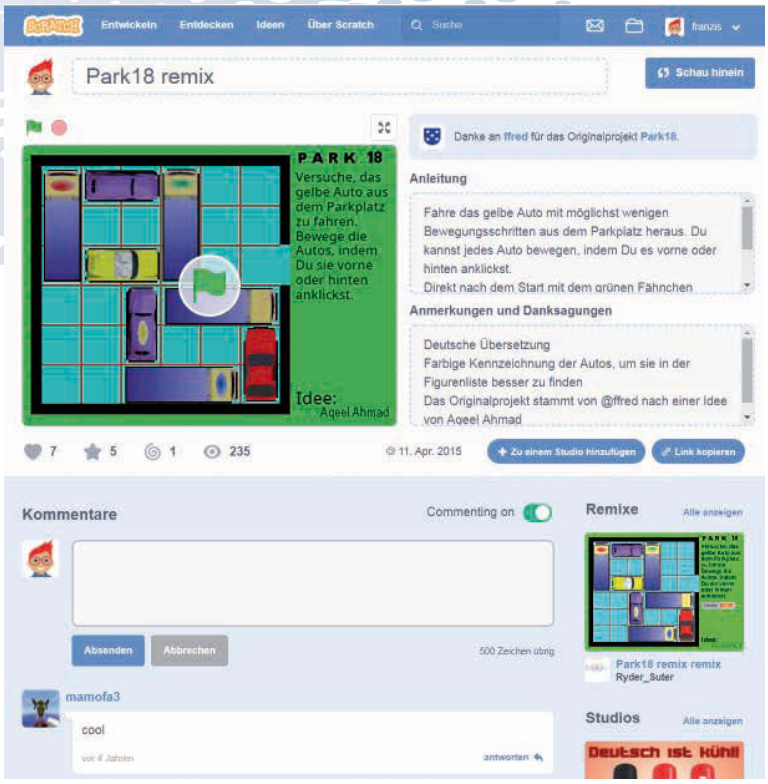
Der Scratcher *franzis* hat dieses Projekt remixt und ins Deutsche übersetzt (*scratch.mit.edu/projects/56573184*). Außerdem wurden die Autos, die im Original alle gleich aussehen, farbig gekennzeichnet, damit man sie im Programm leichter erkennt.

Das Programm sieht auf den ersten Blick kompliziert aus, ist aber eigentlich recht einfach. Bei jedem Fahrzeug gibt es ein Skript, das das Fahrzeug auf die Ausgangsposition setzt, wenn das grüne Fähnchen angeklickt wird.

Ein weiteres Skript steuert die Bewegung, wenn das Fahrzeug angeklickt wird. Dabei wird die Mausposition im Verhältnis zum Drehpunkt des

DIE COOLSTEN SCRATCH-PROJEKTE UND WIE SIE FUNKTIONIEREN

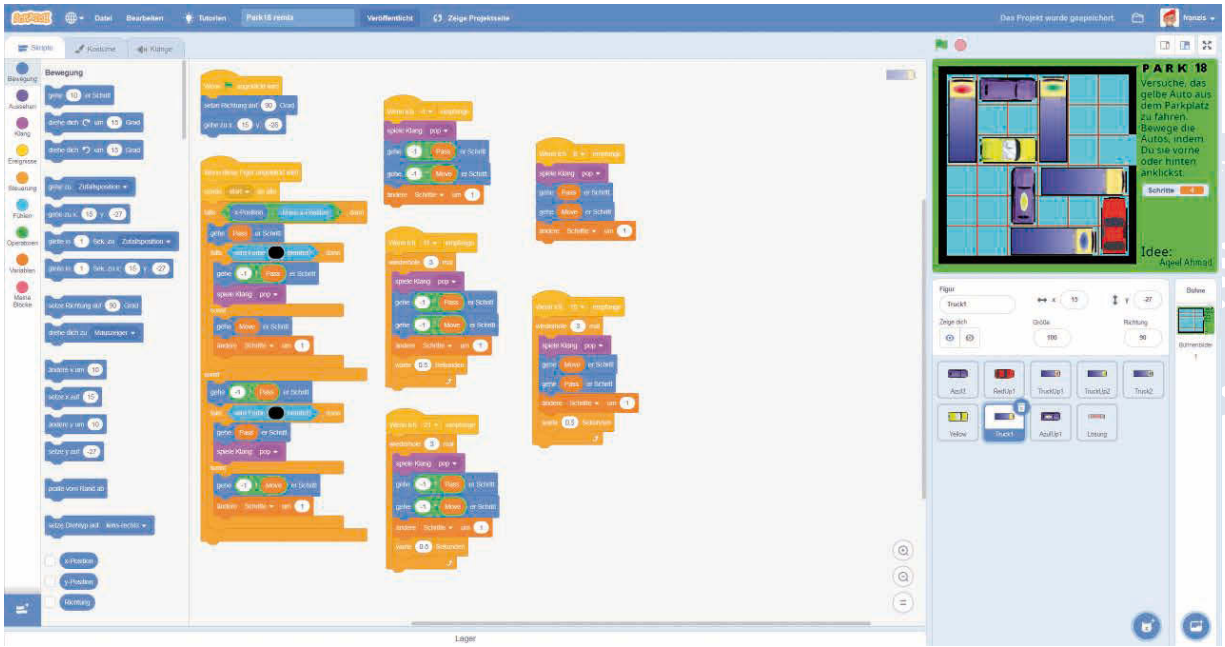
Beim Herumsurfen auf der Scratch-Webseite wirst du sicher jede Menge coole Projekte entdecken. Dort stellen wir jetzt ein paar besonders sehenswerte vor. Folge einfach dem Scratcher *franzis*, und du wirst bei den Lieblingsprojekten immer wieder coole Sachen entdecken.



Fahrzeugs ausgewertet, um die Bewegungsrichtung festzulegen. Das Fahrzeug wird erst einen kleinen Schritt, der in der Variablen ***Pass*** gespeichert ist, bewegt, um zu prüfen, ob es an die Mauer oder an den schwarzen Rand eines anderen Fahrzeugs stößt. Ist dies nicht der Fall, wird es um ein Rasterfeld, das in der Variablen ***Move*** gespeichert ist, bewegt.

Beim gelben Auto wird zusätzlich ständig geprüft, ob sich seine x-Position schon außerhalb des Parkplatzes befindet. Wenn ja, wird eine entsprechende Meldung ausgegeben und das Spiel beendet.

Sobald der erste Bewegungsschritt gemacht ist, wird die Nachricht ***Start*** gesendet und damit der Button ***Lösung*** versteckt. Die Lösung kann



nur am Anfang aufgerufen werden, da sie davon ausgeht, dass sich alle Autos in der Ausgangsposition befinden.

Jeder Klick auf den Button ***Lösung*** sendet eine um 1 höhere Zahl als der vorherige Klick als Nachricht an alle Objekte. Die für die einzelnen Autos erforderlichen Bewegungsschritte sind dort als Skriptblöcke hinterlegt, die aufgerufen werden, wenn die jeweilige Zahl als Nachricht empfangen wird.

PACMAN

PacMan ist ein weiterer Spieleklassiker, der sich geradezu anbietet, mit Scratch nachgebaut zu werden (***scratch.mit.edu/projects/13701368***).

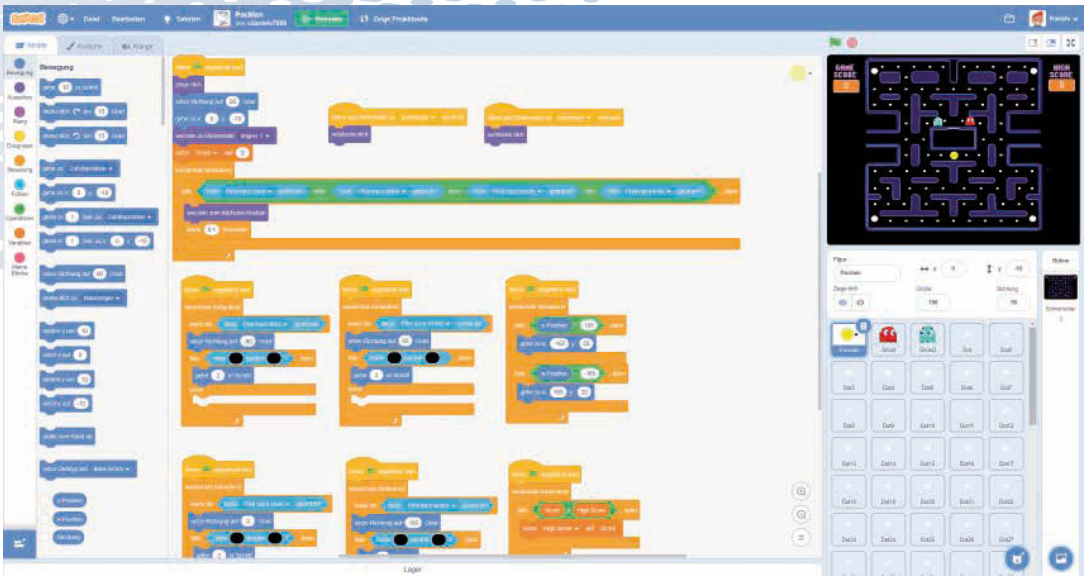
Bewege die gelbe PacMan-Figur mit den Pfeiltasten und versuche, die weißen Punkte zu fressen. Die großen weißen Objekte bringen Bonuspunkte, aber lass dich nicht von den beiden Geistern erwischen.

Bei jedem Drücken einer Pfeiltaste ändert die PacMan-Figur ihr Kostüm und schnappt so mit dem Mund auf und zu. Solange der schwarze Punkt vor der Figur den schwarzen Hintergrund berührt, kann die Figur einen Schritt gehen.

Jeder weiße Punkt (Dot) ist eine eigene Figur, die sich automatisch versteckt, wenn sie die PacMan-Figur berührt. In diesem Moment werden der in der Variablen ***Score*** gespeicherte Punktestand und die in ***Dots*** gespeicherte Zahl der gefressenen Dots erhöht. Die Bonusobjekte funktionieren ähnlich, geben aber über die Nachricht ***BONUS*** den Geistern für acht Sekunden ein anderes Kostüm. In diesem Zustand können sie vom PacMan berührt werden, was 100 Punkte bringt.

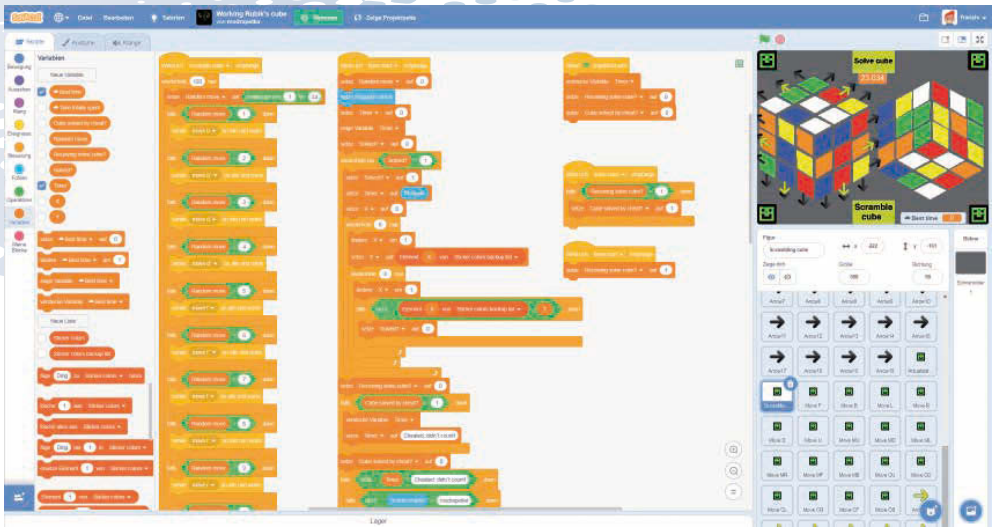
Die Geister haben einen zufällig gesteuerten Bewegungsmechanismus. Werden sie in ihrem Normalzustand vom PacMan berührt, ist das Spiel vorbei.

Wenn das Spiel zu Ende ist, wechselt das Bühnenbild auf ***Game over*** oder ***You Won!***. Beim Wechsel des Bühnenbilds verstecken sich automatisch alle Objekte.



RUBIK'S CUBE

Das Scratch-Projekt ***Working Rubik's cube*** (***scratch.mit.edu/projects/41883578***) simuliert den Klassiker ***Rubik's Cube***, in Deutschland auch als ***Zauberwürfel*** bezeichnet. Der Würfel wird in allen sechs Ansichten gezeigt. Mit den goldenen Pfeilen kannst du den Würfel im Ganzen drehen, um ihn von einer anderen Seite zu betrachten. Mit den schwarzen Pfeilen verdrehst du einzelne Ebenen gegeneinander.

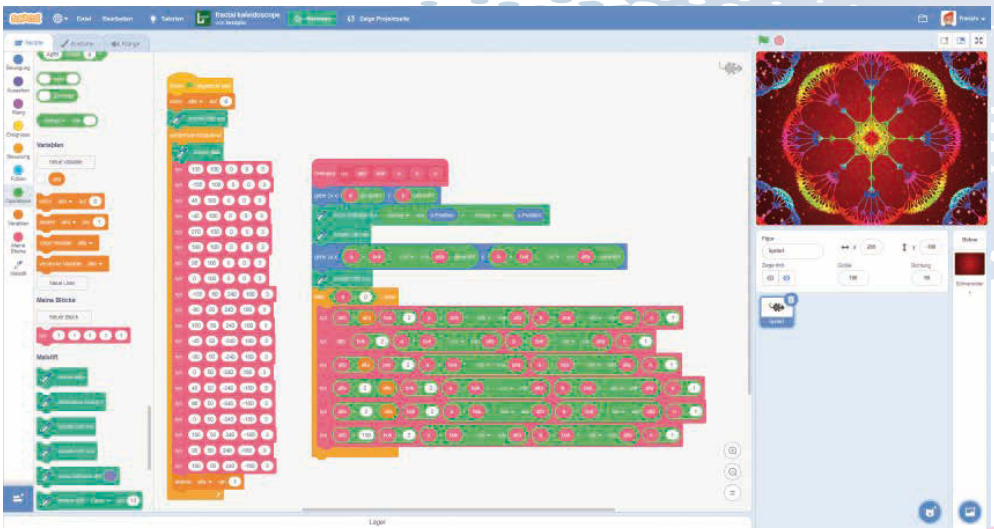


Jede Fläche ist eine eigene Figur, die sechs verschiedene Kostüme für die sechs Farben annehmen kann. Die Farben werden in der Liste ***Sticker colors*** gespeichert und bei einer Drehung entsprechend geändert. Für jede mögliche Bewegung ist eine eigene Figur definiert, die die Farben aller betroffenen Flächen entsprechend dem Bewegungsmuster verändert.

Ein Klick auf ***Scramble cube*** verdreht den Würfel zufällig 100 Mal. Ein Klick auf ***Solve cube*** setzt den Würfel wieder in den Ausgangszustand zurück.

FRACTAL KALEIDOSCOPE

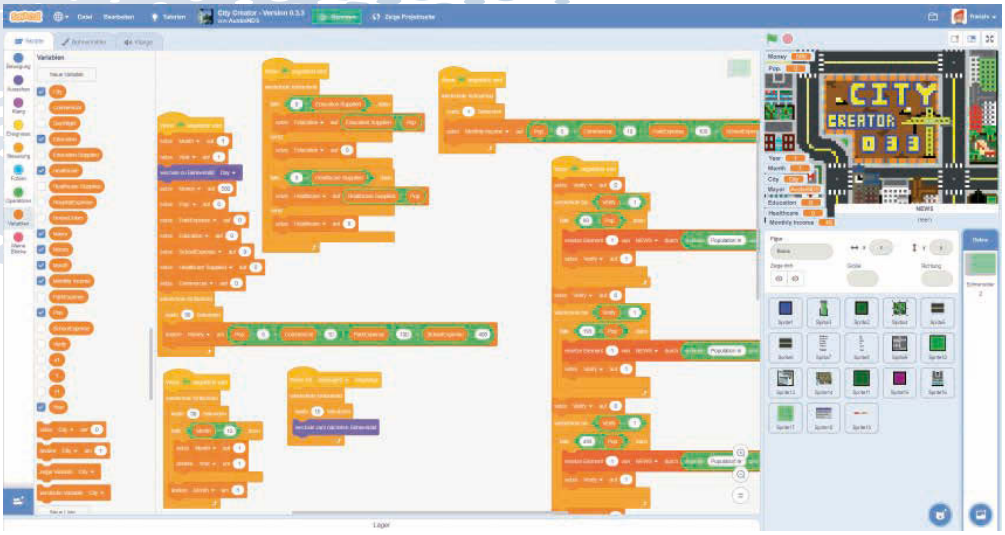
Fractal Kaleidoscope (***scratch.mit.edu/projects/18810758***) zeigt, dass auch mit Scratch komplexe mathematische Berechnungen möglich sind. Das Programm berechnet fraktale Grafiken, die mit dem Malstift eine unsichtbare Figur malt.



Die Grafik entsteht durch einen selbst definierten Block, der sich selbst mehrfach wieder aufruft. So etwas bezeichnet man als rekursive Funktionen.

CITY CREATOR

City Creator (***scratch.mit.edu/projects/44966072***) ist eine einfache Stadtbausimulation, in der du versuchst, eine möglichst wohlhabende und gesunde Stadt zu erschaffen. Spiele das Spiel am besten im Vollbildmodus, da das Spielfeld sehr kleinteilig ist. Am Anfang zeigt ein Tutorial, wie das Bauen funktioniert.



Es gibt verschiedene Arten von Gebäuden, die unterschiedliche Preise haben. Teure Gebäude kannst du erst bauen, wenn du genug Geld hast, das du über die Steuern der Einwohner und Geschäfte einnimmst.

- Wohnhäuser müssen an Straßen gebaut werden. Jede Person zahlt 5 Dollar Steuern. Am Anfang wohnt eine Person im Haus. Wenn die Bildung in der Stadt (***education***) mindestens 1 beträgt und das Haus in der Nähe eines Parks steht, wohnen zwei Personen im Haus. Die dritte Person zieht ein, wenn das Haus in einer Schulzone und neben einem Park steht. Die vierte Stufe wird erreicht, wenn die Stadt ein Krankenhaus hat und die Bedingungen der dritten Stufe erfüllt sind.

- Geschäfte zahlen 10 Dollar Steuern und werden automatisch erweitert, wenn die Bevölkerung zunimmt.

- Parks werden benötigt, um Wohnhäuser zu erweitern. Sie kosten 100 Dollar Betriebskosten in jeder Runde.

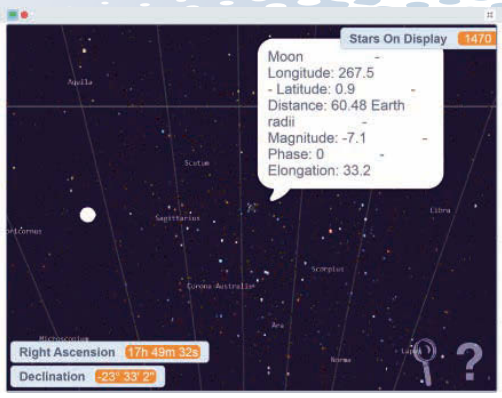
- Straßen sind billig, werden aber gebraucht, um die Bewohner zu den Häusern zu bringen.

- Schulen erhöhen die Bildung. Jede Schule versorgt 1.000 Bewohner und erzeugt eine Schulzone von 6 × 6 Feldern, in denen die Wohnhäuser wertvoller werden. Eine Schule kostet 500 Dollar Betriebskosten in jeder Runde.

- Krankenhäuser erhöhen die Gesundheit (***healthcare***). Jedes Krankenhaus versorgt 1.000 Bewohner und erzeugt eine Zone von 6 × 6 Feldern, in denen die Wohnhäuser wertvoller werden.

VIRTUAL TELESCOPE

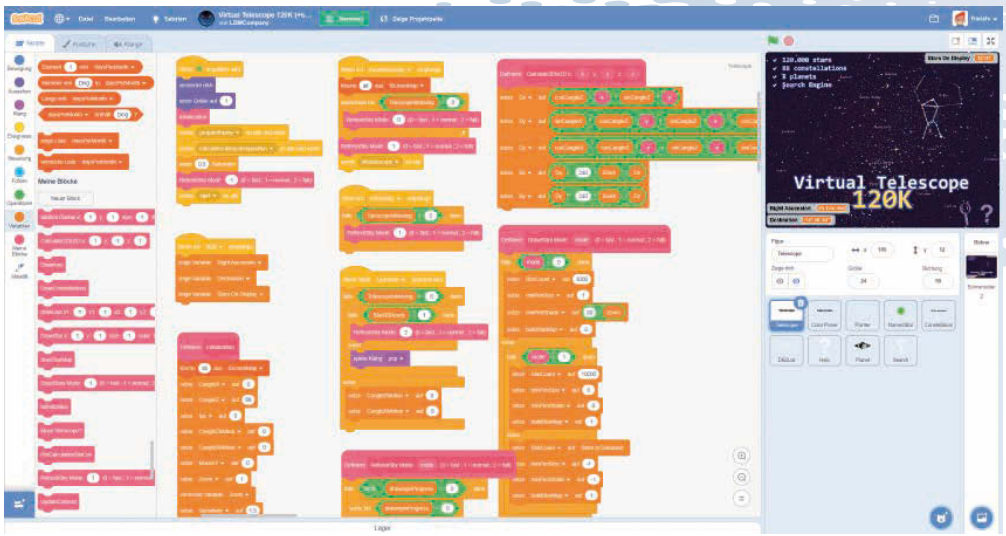
Virtual Telescope (***scratch.mit.edu/projects/52484548***) ist eines der komplexesten Projekte, die mit Scratch entwickelt wurden. Es stellt einen interaktiven Sternenhimmel dar, in dem man sich frei umsehen kann und die Daten zu 120.000 Sternen, 8 Planeten, 1 Mond und 88 Sternbildern sehen kann. Nutze das Programm am besten im Vollbildmodus.



Klickst du auf das Fragezeichen, wird eine kurze Erklärung eingeblendet.

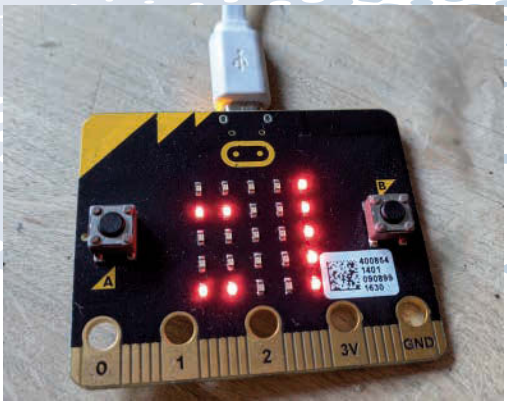
- Pfeiltasten bewegen das Teleskop.
- Z und Pfeil nach oben/unten zoomt die Ansicht.
- S und Pfeil nach oben/unten verändert die Empfindlichkeit und damit die Anzahl der angezeigten Sterne.
- T und Pfeil nach oben/unten verändert das Datum und damit die Position der Planeten.
- Die Leertaste erzeugt aus der Liste der 120.000 Sterne ein Bild mit allen sichtbaren Sternen. Standardmäßig werden aus Performancegründen nur die hellsten 10.000 Sterne berechnet.
- A blendet die Achsen ein oder aus.
- Ein Klick auf den Namen blendet ein Sternbild als Linien ein.
- Fahre mit der Maus über einen Stern, um seine Daten anzuzeigen.

Wenn die Vollversion mit 120.000 Sternen zu langsam läuft: Unter ***scratch.mit.edu/projects/57684230/***, gibt es auch eine Light-Version mit „nur“ 10.000 Sternen .



14 micro:bit mit Scratch steuern

Der micro:bit ist eine kleine Experimentierplatine, die sich sehr einfach programmieren lässt. Sie wurde ursprünglich für den Schulunterricht entwickelt, lässt sich aber auch für viele andere Experimente und Spiele einsetzen.



Natürlich ist der micro:bit nicht die erste programmierbare Platine. Bis vor einigen Jahren war die Programmierung von Mikrocontrollern und Einplatinencomputern nur etwas für Ingenieure. Erst die bekannten Platinen Raspberry Pi und Arduino machten diese Technik für jeden verständlich.

Im Gegensatz zum Raspberry Pi ist der micro:bit kein wirklicher Computer, sondern eine Mikrocontrollerplatine, die einfach ein einziges Programm abarbeitet, das vorher auf dem PC erstellt und dann per USB-Kabel übertragen wurde. Ein großer Vorteil des micro:bit gegenüber ähnlichen Mikrocontrollern ist die Vielzahl fest auf der Platine verbauter Sensoren.

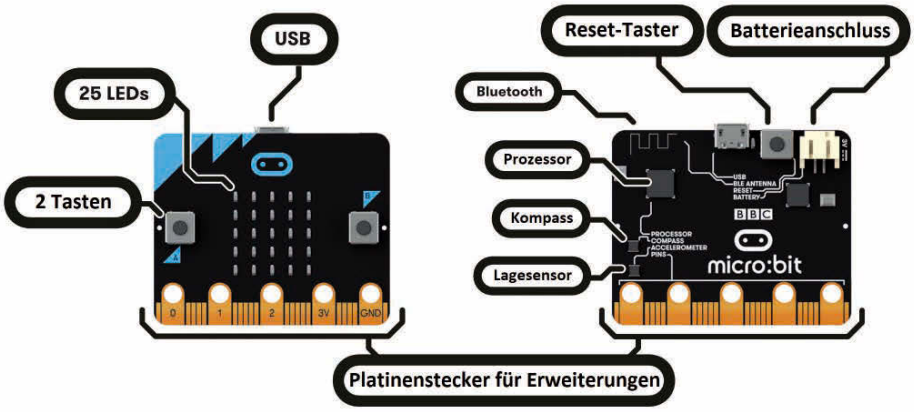
Unter anderem gibt es zwei Taster, drei berührungsempfindliche Kontakte, Sensoren für Beschleunigung, Helligkeit und Temperatur sowie eine Bluetooth-Schnittstelle zur Verbindung mit anderen Geräten. Eine Matrix aus 5 × 5 LEDs ermöglicht die Darstellung von Text oder einfacher Grafik direkt auf der Platine.

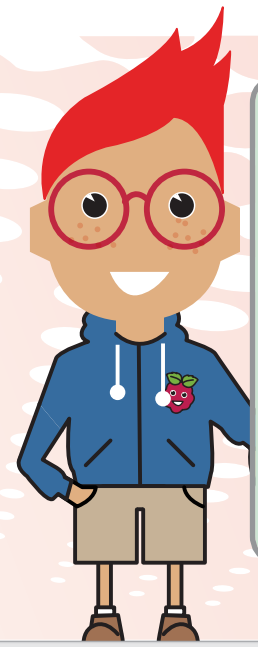
Die Scratch-Erweiterung ***micro:bit*** programmiert den Mikrocontroller nicht selbst, sondern nutzt ihn für die Ein- und Ausgabe in Programmen, die auf dem PC laufen. Die Verbindung zwischen PC und micro:bit erfolgt per Bluetooth. Wenn du den micro:bit über eine Batterie mit Strom versorgst, kannst du dich so frei bewegen. Es ist nach der Treiberinstallation keine Kabelverbindung mit dem PC mehr nötig.

MICRO:BIT ZUR VERWENDUNG MIT SCRATCH EINRICHTEN

Um den micro:bit mit Scratch nutzen zu können, musst du ein paar Vorbereitungen treffen:

0 Schließe den micro:bit über ein USB-Kabel am PC an. Dafür kannst du ein Datenkabel





STROMVERSORGUNG ÜBER BATTERIEN

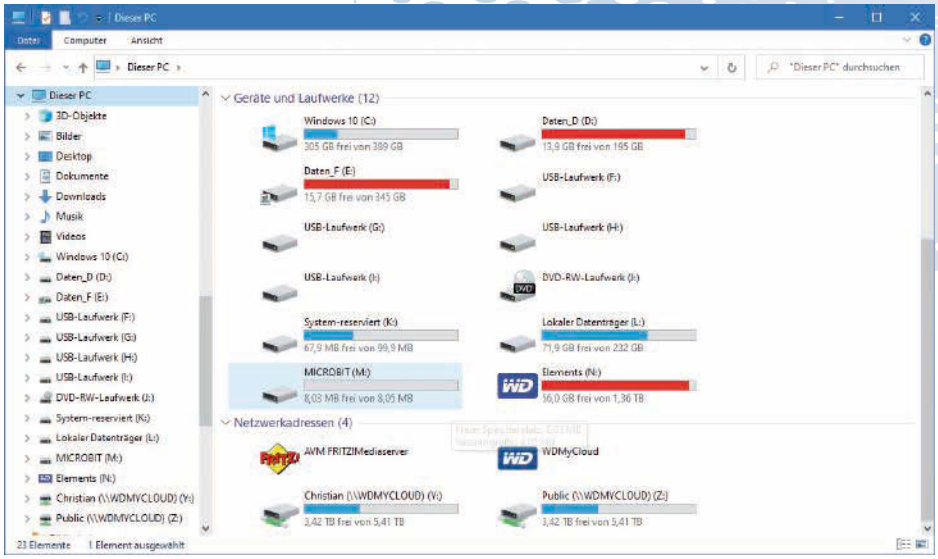
Der micro:bit hat einen Anschluss für einen Batteriekasten. Das ist der weiße Stecker in der Ecke auf der Rückseite. Solche Steckverbindungen sind im Modellbau üblich. Du kannst den micro:bit über den USB-Anschluss auch mit einem Handyladegerät oder einer Powerbank mit Strom versorgen. Wenn du eine Powerbank verwendest, nimm möglichst eine ältere.

Moderne Hochleistungs-Powerbanks schalten sich automatisch in einen Sparmodus, wenn kein Strom entnommen wird, um eine Selbstentladung der Akkus über das Kabel oder ein angeschlossenes Gerät zu verhindern. Der micro:bit braucht so wenig Leistung, dass es die Powerbanks oft nicht merken und sich dann abschalten. Aber ganz ohne Strom funktioniert auch der micro:bit nicht.

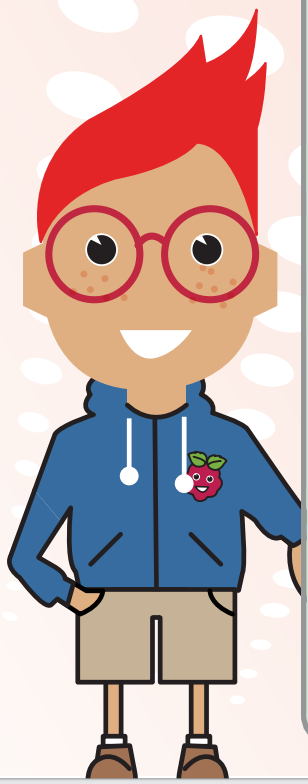
eines Smartphones verwenden. Der micro:bit hat den typischen Micro-USB-Anschluss.

2 Beim ersten Anschließen wird automatisch ein Treiber installiert, was nur einige Sekunden dauert. Danach findest du den micro:bit im Windows-Explorer bei den Laufwerken unter dem Namen ***MICROBIT*** mit einem bisher nicht verwendeten Laufwerkbuchstaben. Der micro:bit ist aber trotzdem kein normales Laufwerk. Es kann immer nur eine Programmdatei darauf abgelegt werden. Sie ist im Windows-Explorer auf dem Laufwerk nicht zu sehen. Auch das Speichern anderer Dateitypen ist nicht möglich.

3 Lade dir bei ***downloads.scratch.mit.edu/microbit/scratch-microbit-1.1.0.hex.zip*** ein



14 micro:bit mit Scratch steuern

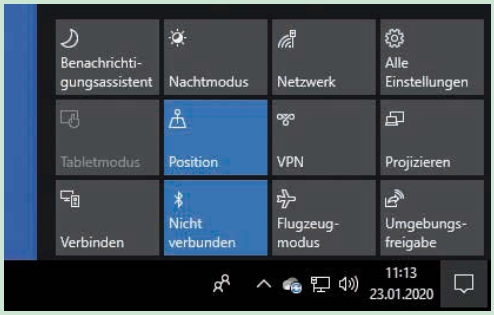


SYSTEMVORAUSSETZUNGEN

Die Verbindung funktioniert ab der Windows-10-Version 1709. Ältere Windows-Versionen wie Windows 7 oder Windows 8.1 werden nicht unterstützt. Wenn du in Windows 10 die automatischen Updates aktiviert hast, kannst du sicher sein, eine neuere Windows-10-Version zu haben. Die Version 1709 wurde im Herbst 2017 verteilt.

Zur Verbindung mit dem micro:bit muss dein PC Bluetooth haben, was bei den meisten Laptops und modernen PCs der Fall ist. Sollte dein PC kein Bluetooth eingebaut haben, kannst du einen Bluetooth-USB-Stick verwenden. Klicke auf das Benachrichtigungssymbol ganz rechts in der Taskleiste. Wenn Bluetooth verfügbar ist, erscheint bei den Schnelleinstellungen ein Bluetooth-Symbol.

Falls es ausgeschaltet (grau) ist, klicke darauf, um Bluetooth einzuschalten. Das Bluetooth-Symbol sollte ***Nicht verbunden*** anzeigen, da Scratch die Verbindung selbst steuert. Du brauchst in den Windows-Einstellungen kein Bluetooth-Gerät zu koppeln.



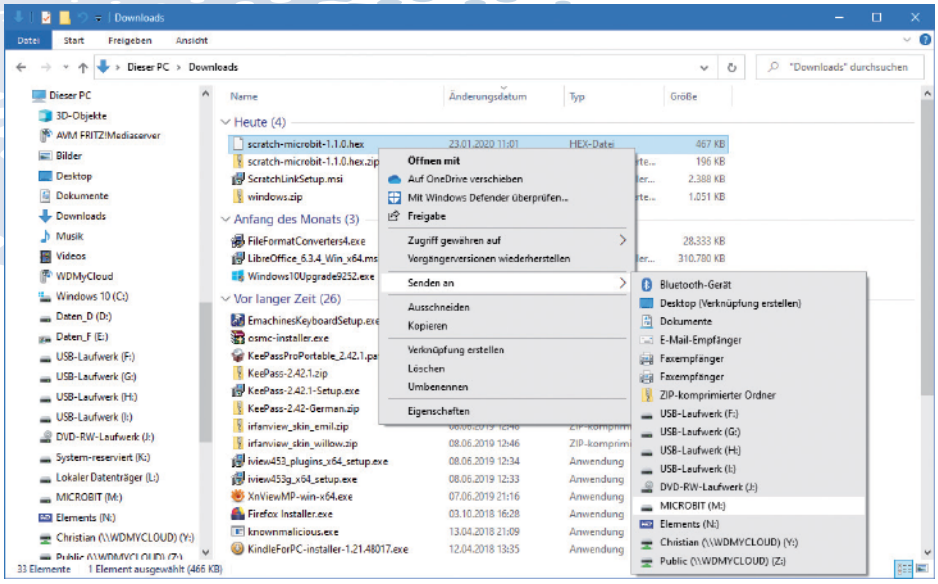
Sollte ein verbundenes Bluetooth-Gerät angezeigt werden, wie zum Beispiel ein Smartphone oder ein Bluetooth-Lautsprecher, trenne diese Verbindung.

kleines Programm herunter, das auf dem micro:bit die Verbindung zum PC steuert. Entpacke das Zip-Archiv auf der Festplatte.

4 Kopiere die entpackte Datei ***scratch-microbit-1.1.0.hex*** auf den micro:bit. Du kannst im Explorer auch einfach den Kontextmenüpunkt ***Senden an*** nutzen.

5 Jetzt kannst du den micro:bit wieder vom PC trennen und anderweitig mit Strom versorgen. Die weitere Kommunikation erfolgt über Bluetooth. Der micro:bit zeigt auf der LED-Matrix einen automatisch generierten Gerätenamen an, der später bei der Bluetooth-Verbindung erscheint. Dieser ist wichtig, um den richtigen zu identifizieren, falls mehrere micro:bits in Bluetooth-Reichweite sind.

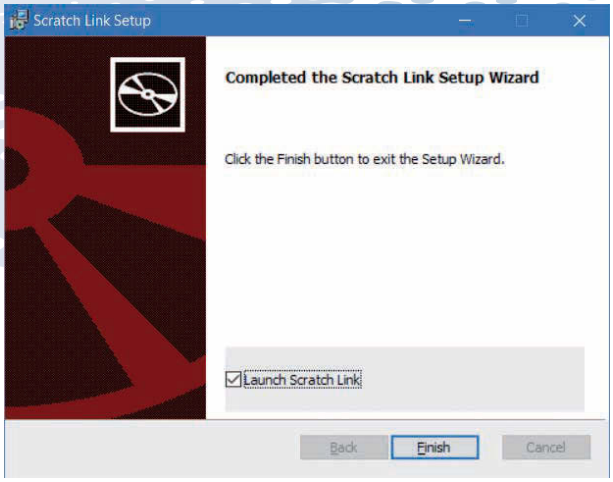
6 Lade dir bei ***downloads.scratch.mit.edu/link/windows.zip*** das Windows-Programm ***Scratch Link*** herunter, entpacke das Zip-Archiv auf der Festplatte und installiere die einzige darin enthaltene Datei per Doppelklick. Setze im letzten Dia-

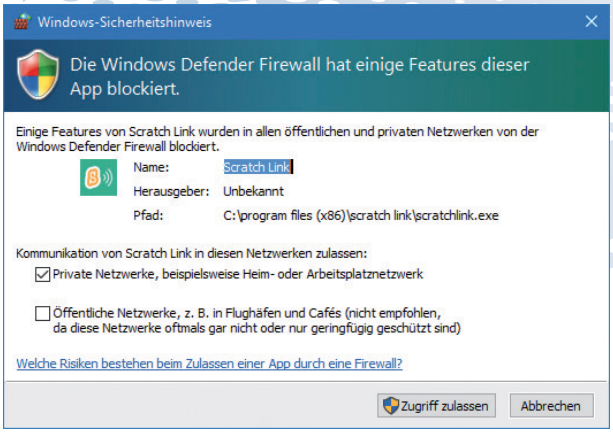


7 ***Scratch Link*** erscheint jetzt als Symbol im Infobereich der Taskleiste. Hier kannst du jederzeit sehen, ob es auch wirklich läuft. Nach einem Neustart des PCs musst du dieses Programm wieder aus dem Startmenü starten, wenn du die Scratch-Verbindung zum micro:bit nutzen möchtest.

8 Bei der ersten Verbindung meldet sich die Windows Defender Firewall. Dort musst du den Zugriff zulassen, damit sich ***Scratch Link*** per Bluetooth verbinden kann.

logfeld der Installation das Häkchen bei ***Launch Scratch Link***.





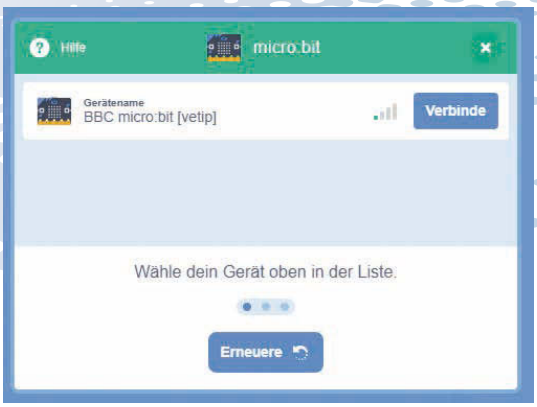
14 micro:bit mit Scratch steuern

EINFACHES PROGRAMM ZUM AUSPROBIEREN

Ein einfaches Programm zeigt, wie Scratch mit dem micro:bit funktioniert. Das Programm zeigt beim Drücken der Taste A oder B auf dem micro:bit ein ***A*** oder ein ***B*** an.

1. Installiere in Scratch die Erweiterung ***micro:bit***.

2. Direkt nach der Installation wird die Verbindung zum micro:bit hergestellt. Es erscheint eine Liste aller micro:bits in der Nähe. In den meisten Fällen ist es nur einer. Bei jedem Gerät wird der Gerätename angegeben, den der micro:bit auf seiner LED-Matrix anzeigt. Wähle deinen micro:bit aus und klicke auf ***Verbinde***.



3. Sobald der micro:bit erfolgreich verbunden ist, klicke auf ***Gehe zum Editor***. Die Erweiterung zeigt zusätzliche Blöcke an.

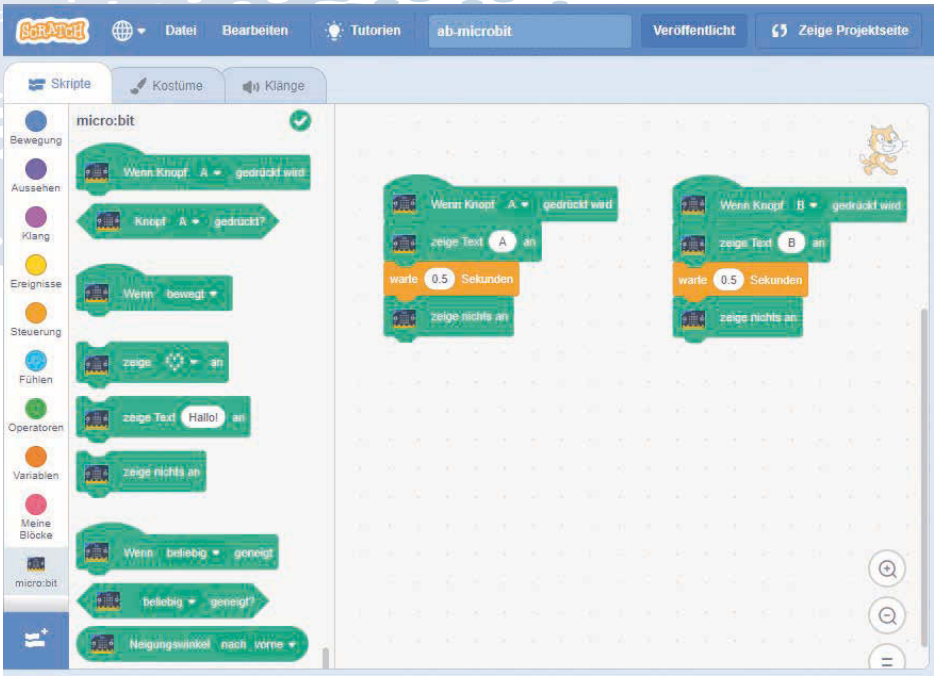
4. Baue das abgebildete Programm zusammen. Der Block ***Wenn Knopf A gedrückt wird*** startet die angehängten Blöcke, wenn du die linke Taste A auf dem micro:bit drückst.

5. Anschließend zeigt ein Block ***zeige Text ... an*** den Buchstaben ***A*** auf der LED-Matrix. Schreibe dazu ein ***A*** in das Textfeld dieses Blocks.

6. Danach wartet das Programm eine halbe Sekunde und löscht die Anzeige auf der LED-Matrix mit dem Block ***zeige nichts an*** wieder.

7. Der zweite Programmteil für die rechte Taste B funktioniert ganz ähnlich. Du kannst den ersten Teil duplizieren und im Block ***Wenn Knopf ... gedrückt wird*** die Taste B wählen. Schreibe dann noch ein ***B*** in das Textfeld des Blocks ***zeige Text ... an***.

8. Du kannst das Programm sofort mit dem micro:bit ausprobieren. Es müssen keine



und der Ball diesen Rand nicht berühren darf.

In der ersten Version steuerst du den Schläger mit den Pfeiltasten nach links und rechts auf der Tastatur. Probiere es mit einem Klick auf das grüne Fähnchen aus.

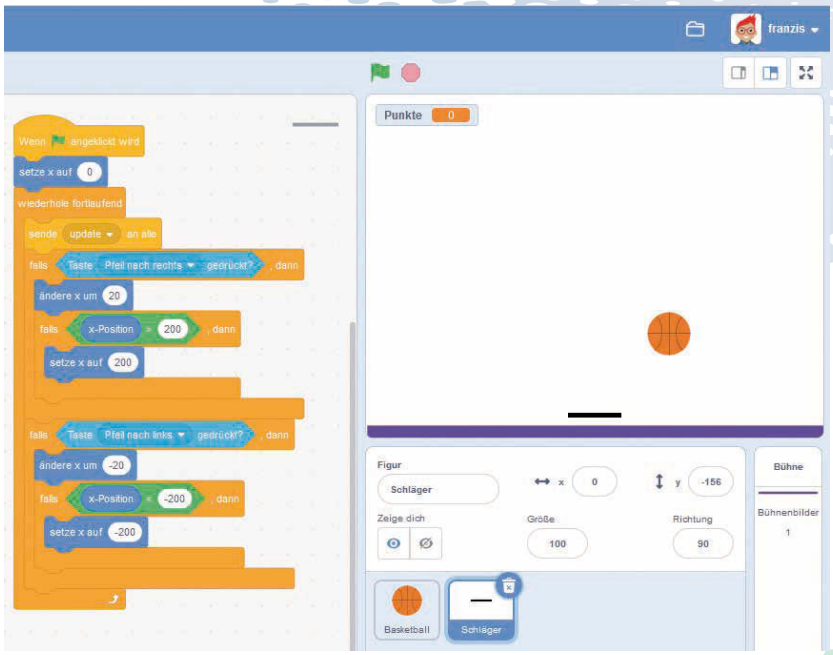
Baue jetzt das Programm für die Verwendung mit dem micro:bit um:

0 Installiere in Scratch die Erweiterung ***micro:bit***.

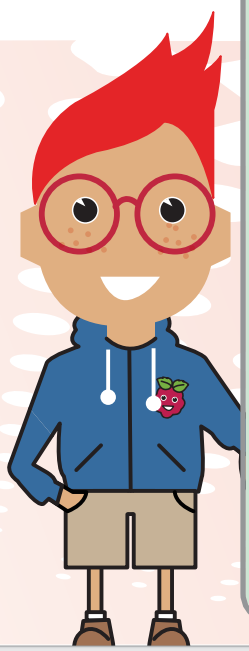
Dateien mehr übertragen werden, und das grüne Fähnchen brauchst du auch nicht.

PONG MIT DEM MICRO:BIT STEUERN

Nach dem ersten sehr einfachen Programm verwenden wir jetzt den micro:bit als drahtloses Gamepad zur Steuerung eines PC-Spiels. Das Spiel ***pong-microbit*** basiert auf dem Pong-Spiel vom Anfang dieses Buchs. Der wesentliche Unterschied besteht darin, dass sich der Schläger waagerecht am unteren Bildschirmrand bewegt



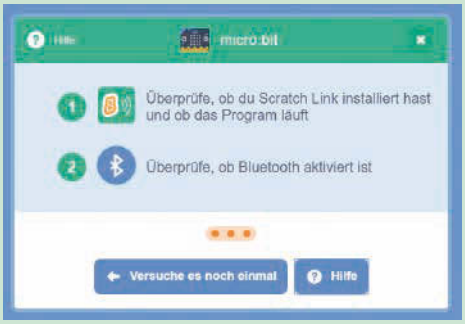
14 micro:bit mit Scratch steuern



VERBINDUNG NACH NEUSTART NEU AUFBAUEN

Wenn du Scratch verlassen, den PC neu gestartet oder einfach nur ein anderes Programm ausgewählt hast, muss der micro:bit neu verbunden werden. Die Erweiterung ***micro:bit*** zeigt oben auf der Blockpalette ein grünes Häkchen an, wenn eine Verbindung besteht.

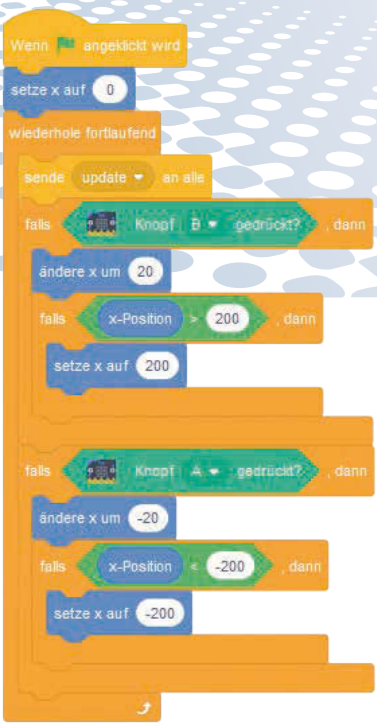
Wurde die Verbindung getrennt, erscheint ein orangefarbenes Ausrufezeichen. Klicke darauf, um die Verbindung neu aufzubauen. Das funktioniert genau so wie am Anfang bei der Installation der Erweiterung. Falls kein micro:bit gefunden wird, erscheint ein Hinweis, woran es liegen könnte.

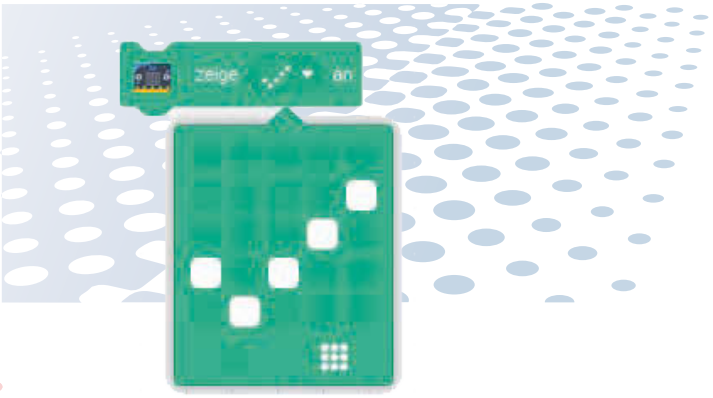


Prüfe in diesem Fall außerdem, ob der micro:bit mit Strom versorgt wird und seinen Gerätenamen auf der LED-Matrix anzeigt.

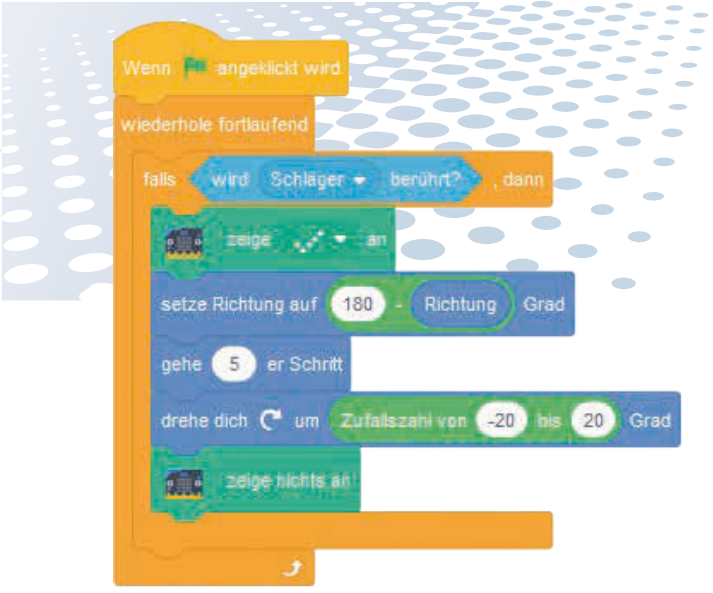
2 Entferne als Erstes die beiden ***Fühlen***-Blöcke ***Taste Pfeil nach rechts gedrückt*** und ***Taste Pfeil nach links gedrückt*** aus dem Programm für den Schläger. Ersetze sie durch die Blöcke ***Knopf B gedrückt*** und ***Knopf A gedrückt*** aus der Erweiterung ***micro:bit***.

3 Zusätzlich soll der micro:bit, wenn du einen Ball erfolgreich abgewehrt hast, ein Häkchen auf der LED-Matrix anzeigen. Ziehe dazu einen Block ***zeige ... an*** aus der Erweiterung ***micro:bit*** in das Programm, klicke auf das Symbolfeld in diesem Block und zeichne das Häkchen, indem du per Mausklick einzelne LEDs ein- oder ausschaltest.

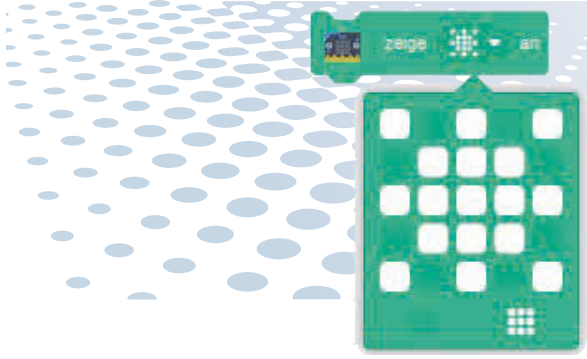




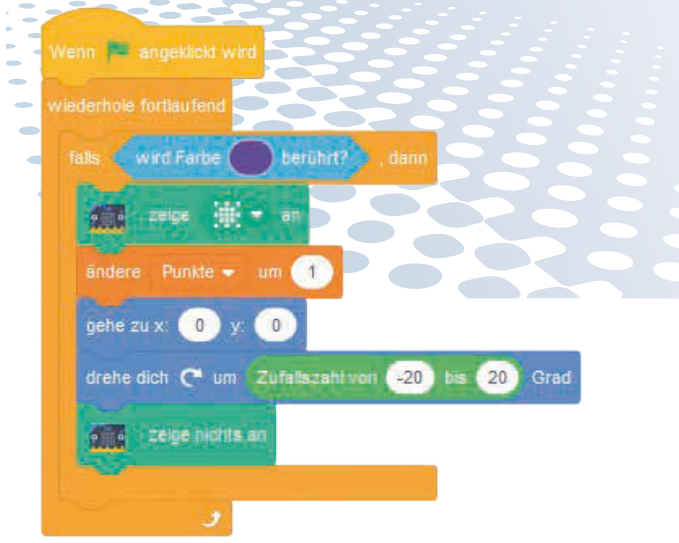
4 Ziehe diesen Block in die ***falls wird Schläger berührt?, dann***-Abfrage bei der Figur ***Basketball***. Nachdem sich der Ball einen 5er-Schritt in die neue Richtung bewegt hat, schalte die LED-Matrix mit dem Block ***zeige nichts an*** wieder aus. Die LEDs leuchten nur einen winzigen Bruchteil einer Sekunde. Da das menschliche Auge sehr träge ist, kann man das Bild aber deutlich aufleuchten sehen, ohne dass du in das Programm Wartezeiten einbauen musst.



5 Auf die gleiche Weise soll das Programm ein Explosionssymbol anzeigen, wenn der Ball die violette Grundlinie der Bühne berührt. Auch dafür verwenden wir einen Block ***zeige ... an*** aus der Erweiterung ***micro:bit***.



6 Ziehe diesen Block in die ***falls wird Farbe Violett berührt?, dann***-Abfrage bei der Figur ***Basketball***. Nachdem der Punkt gezählt und der Ball wieder in die Mitte gesetzt wurde, schalte die LED-Matrix mit dem Block ***zeige nichts an*** wieder aus.



7 Das waren schon alle notwendigen Änderungen. Klicke auf das grüne Fähnchen und nutze den micro:bit als Gamepad zur Steuerung des Spiels.

15 Elektronik mit Scratch auf dem Raspberry Pi steuern

Mit einem normalen PC oder gar einem Notebook einfache Elektronik zu steuern ist – auch wenn es nur ein paar LEDs sind – für Hobbyprogrammierer mit kaum vertretbarem Aufwand verbunden. Dem PC fehlen einfach die dafür notwendigen Schnittstellen. Außerdem ist das Windows-Betriebssystem denkbar ungeeignet, um mit Elektronik zu kommunizieren.

Der Bastelcomputer Raspberry Pi ist – obwohl er auf den ersten Blick gar nicht so aussieht – ein vollwertiger Computer. Du kannst damit Texte schreiben, ins Internet gehen, Spiele spielen oder Filme ansehen. Vieles geht etwas langsamer, als man es von modernen PCs gewohnt ist, dafür ist der Raspberry Pi ja auch viel kleiner und vor allem billiger als ein PC.

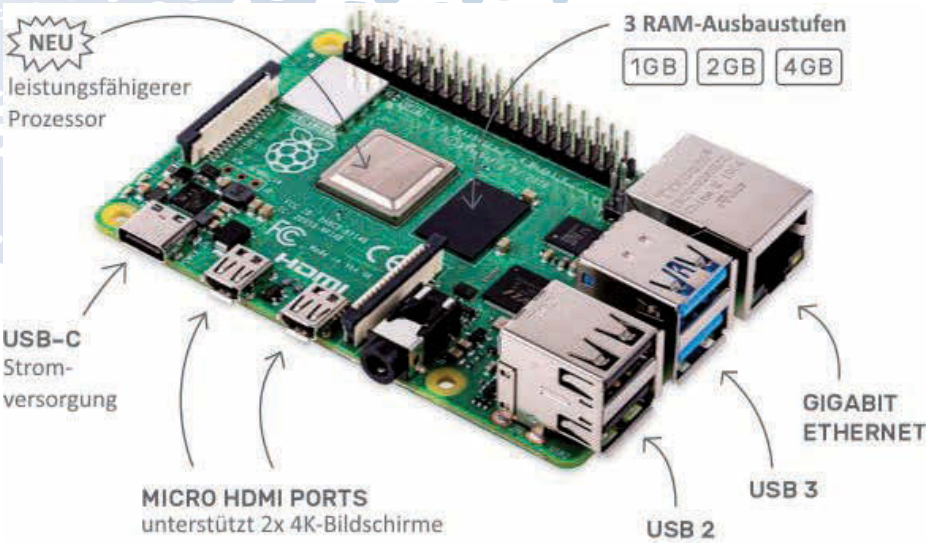
Und du kannst den Raspberry Pi programmieren – das geht sogar viel leichter als auf dem PC, weil alle erforderlichen Werkzeuge bereits vorinstalliert sind und einfachere, verständliche Programmiersprachen verwendet werden.

Die folgenden Beschreibungen beziehen sich auf den neuesten Raspberry Pi 4. Vieles funktioniert aber auch mit den Vorgängermodellen Raspberry Pi 3 und 3B+.

DAS BRAUCHST DU NOCH

Wie für einen „großen" PC braucht man auch für den Raspberry Pi noch einiges an Zubehör, um ihn wirklich benutzen zu können:

- Handyladegerät mit USB-C-Stecker als Netzteil
- Monitor mit HDMI-Anschluss und Micro-HDMI-Adapter
- USB-Tastatur und Maus
- Netzwerkkabel oder WLAN



(Grafik: Raspberry Pi Foundation – Creative-Commons-Lizenz)

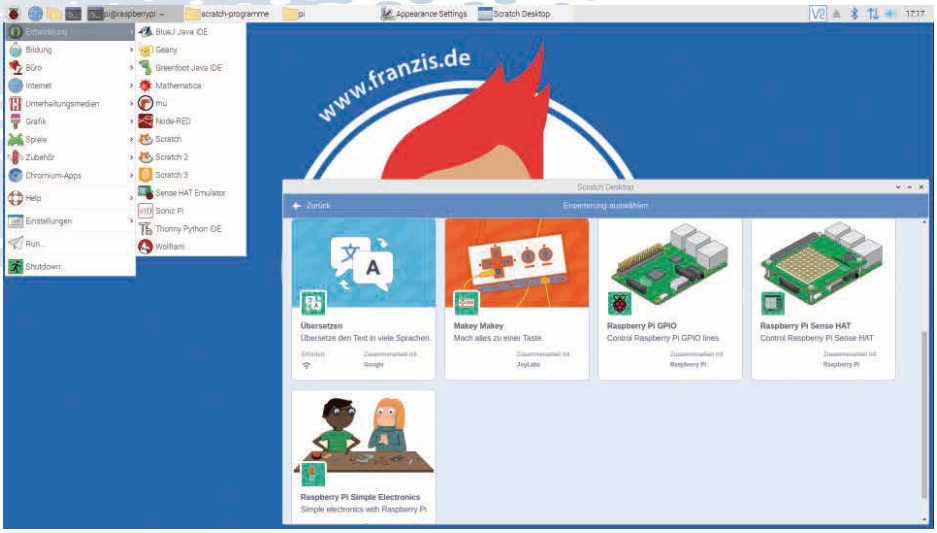
- Raspbian-Betriebssystem auf einer MicroSD-Speicherkarte (***www.raspberrypi.org/downloads***)

Für die aktuelle Originalversion des Betriebssystems Raspbian Buster muss die Speicherkarte mindestens 8 GByte groß sein. Alternativ gibt es eine Version mit dem gleichen Desktop, aber ohne weitere vorinstallierte Software, die

mit 4-GByte-Speicherkarten auskommt. Da kann Scratch problemlos nachinstalliert werden.

SCRATCH AUF DEM RASPBERRY PI

Wenn du bereits einen Raspberry Pi nutzt, hast du das vorinstallierte Scratch möglicherweise schon entdeckt. Das aktuelle Raspbian-Betriebssystem liefert Scratch in drei Versionen mit. Das aktuelle Scratch 3.0 läuft auf dem Raspberry Pi 4 optimal. Für ältere Hardware kannst du Scratch 2.0 verwenden. Die erste Version Scratch 1.4 ist nur noch aus historischen Kompatibilitätsgründen vorinstalliert. Du findest Scratch im Startmenü auf dem Raspbian-Desktop unter ***Entwicklung***.



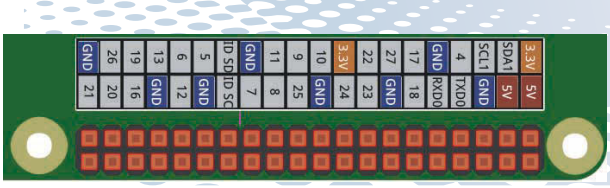
Teilweise heißen die Blöcke in den älteren Versionen etwas anders, und einige Blöcke fehlen gegenüber der Version 3.0. Im alten Scratch 1.4 gab es noch keine Erweiterungen im heutigen Stil. Dort kann die GPIO-Schnittstelle aber über eine mitgelieferte Zusatzsoftware ebenfalls angesteuert werden. An manchen Stellen wirkt die Bedienung der älteren Scratch-Versionen etwas umständlicher, man gewöhnt sich aber schnell um, wenn man Scratch 3.0 kennt.

GPIO-STEUERUNG MIT SCRATCH AUF DEM RASPBERRY PI

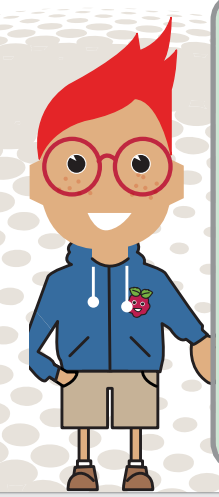
Was den Raspberry Pi bei Makern so beliebt macht, ist die Möglichkeit, Elektronik anzuschließen und diese mit eigenen Programmen zu steuern.

Die Stiftleiste an der einen Seite des Raspberry Pi wird als GPIO bezeichnet. Die englische Abkürzung ***General Purpose Input Output*** bedeutet auf Deutsch einfach ***Allgemeine Ein- und Ausgabe***.

Alle die Pins, die nur mit einer Zahl bezeichnet sind, können wahlweise als Eingang oder als Ausgang programmiert und so für vielfältige Hardwareexperimente genutzt werden. Die übrigen sind für die Stromversorgung und andere Zwecke fest eingerichtet. Es gibt je zwei Pins mit +3,3 V und +5 V Spannung sowie insgesamt acht Pins mit Masseanschluss GND.



15 Elektronik mit Scratch auf dem Raspberry Pi steuern



IMMER +3,3 V VERWENDEN

Verwende für elektronische Experimente immer die +3,3-V-Anschlüsse des Raspberry Pi und nicht die +5-V-Anschlüsse. Käme 5-V-Spannung durch eine elektronische Schaltung wieder zurück auf einen GPIO-Pin, würde es den Raspberry Pi beschädigen. Die +5-V-Anschlüsse sind nur für externe Geräte wie zum Beispiel Servomotoren oder größere Anzeigemodule gedacht und nicht für Logiksignale.

SCRATCH MIT GPIO-UNTERSTÜTZUNG

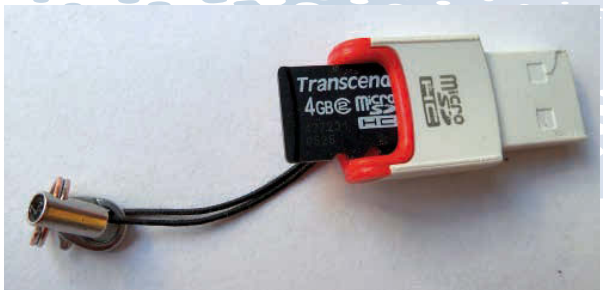
Der Raspberry Pi verwendet Linux und kein Windows, aber das sieht auch nicht viel anders aus. Linux hat den großen Vorteil, dass es längst nicht so hardwarehungrig ist wie Windows. Außerdem ist Linux „freie Software", das heißt, jeder darf Linux nutzen, für seine eigenen Zwecke anpassen und beliebig weitergeben, ohne Seriennummern verwenden oder sich mit Lizenzen herumärgern zu müssen.

Die Linux-Version für den Raspberry Pi nennt sich Raspbian und unterstützt natürlich auch die GPIO-Schnittstelle des Raspberry Pi.

In Scratch 3 werden Erweiterungen angeboten, um die GPIO-Schnittstelle zu steuern. Aber natürlich kannst du auch die anderen Scratch-Programme aus diesem Buch, die sonst auf dem PC im Browser laufen, auf dem Raspberry Pi einsetzen.

SPEICHERKARTE ZUR BETRIEBSSYSTEM-INSTALLATION VORBEREITEN

Zuerst bereiten wir die Speicherkarte auf dem PC vor. Dazu brauchst du einen Kartenleser am PC. Er kann fest eingebaut oder per USB angeschlossen werden. Wenn dein PC noch keinen Kartenleser hat, besorg dir am besten einen kleinen Kartenleser in USB-Stick-Form. Die einfachen Modelle, die nur MicroSD-Karten lesen und bei manchen Speicherkarten sogar schon mitgeliefert werden, reichen völlig aus, da wir genau dieses Kartenformat für den Raspberry Pi brauchen und andere Kartentypen inzwischen weitgehend ungebräuchlich sind. Smartphones und Tablets verwenden üblicherweise ebenfalls MicroSD-Karten. Bei im PC eingebauten SD-Kartenlesern ist für MicroSD-Karten manchmal ein Adapter erforderlich, der bei neuen MicroSD-Karten oft bereits mitgeliefert wird.



ACHTUNG

Egal ob neu formatiert oder nicht, bei der Installation des Raspberry Pi-Betriebssystems werden alle Daten, die sich auf der Speicherkarte befinden, unwiderruflich gelöscht. Also solltest du vorher alles sichern, was wichtig ist.

1. Lade dir bei ***https://www.raspberrypi.org/downloads/*** das Programm Raspberry Pi Imager herunter, installiere es auf deinem PC und starte es.

2. Klicke auf ***Operating System (Choose OS)*** und wähle unter ***Raspbian (other)*** die Variante ***Raspbian Full***.

3. Klicke auf ***SD Card (Coose SD Card)*** und wähle die Speicherkarte im Kartenleser aus. In den meisten Fällen wird nur eine Speicherkarte zur Auswahl angeboten. Die Speicherkarte wird automatisch formatiert.

4. Klicke auf ***Write***. Dann wird das Betriebssystem heruntergeladen und automatisch auf die Speicherkarte übertragen. Warte, bis der Vorgang komplett abgeschlossen ist.

5. Boote den Raspberry Pi mit dieser Speicherkarte.

15 Elektronik mit Scratch auf dem Raspberry Pi steuern

DER ERSTE START DES RASPBERRY PI

Jetzt ist es so weit, und du kannst den Raspberry Pi zum ersten Mal booten. Wundere dich nicht, wie schnell das geht. Trotz der geringen Hardwareleistung bootet das komplette Linux-Betriebssystem deutlich schneller als ein Windows-PC.

1 Stecke die Speicherkarte in den Steckplatz auf der Rückseite des Raspberry Pi und schließe Tastatur, Maus und Monitor an. Der USB-Stromanschluss kommt als Letztes. Damit wird der Raspberry Pi eingeschaltet. Einen extra Einschaltknopf gibt es nicht.

2 Der Raspberry Pi bootet und zeigt dabei auf einem schwarzen Bildschirm diverse Linux-Kommandos, die schnell durchrauschen und uns erst einmal nicht weiter interessieren müssen.

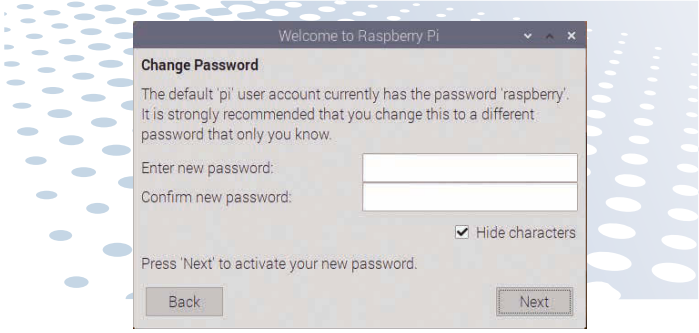
3 Jetzt erscheint der Installationsbildschirm von NOOBS. Wähle ganz unten ***Deutsch*** als Installationssprache aus und setze das Häkchen beim vorausgewählten Raspbian-Betriebssystem. Nachdem du den Sicherheitshinweis, dass die Speicherkarte überschrieben wird, bestätigt hast, startet die Installation, die einige Minuten dauert. Während der Installation werden kurze Informationen zum Raspbian-Betriebssystem angezeigt.

4 Nach abgeschlossener Installation klicke auf die Meldung ***Betriebssystem(e) erfolgreich installiert***. Danach bootet der Raspberry Pi neu.

5 Beim ersten Start erscheint ein Konfigurationsassistent, der – leider nur auf Englisch – einige wichtige Grundeinstellungen abfragt. Klicke im ersten Fenster mit der Himbeere unten rechts auf ***Next***.

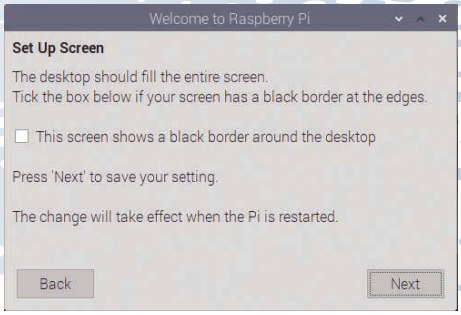
6 Wähle im nächsten Schritt bei ***Country*** das Land ***Germany***. Damit werden alle anderen Einstellungen für Deutschland automatisch richtig eingestellt. Die Benutzeroberfläche erscheint nach dem nächsten Neustart auf Deutsch, und die Zeitzone wird richtig ausgewählt, damit der Raspberry Pi aus dem Internet die korrekte Uhrzeit bekommt. Klicke danach erneut auf ***Next***.

7 Im nächsten Schritt wird empfohlen, das Standardpasswort zu ändern. Der Standardbenutzer ***pi*** wird beim Booten automatisch ange-

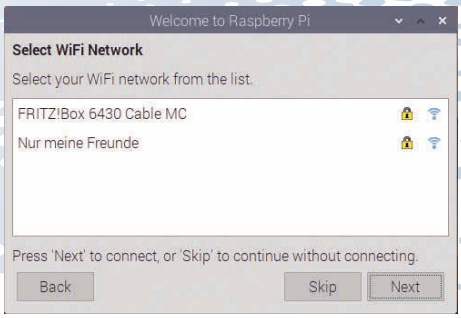


meldet, sodass man das Passwort nur selten braucht. Du kannst diesen Schritt auch einfach mit einem Klick auf ***Next*** überspringen.

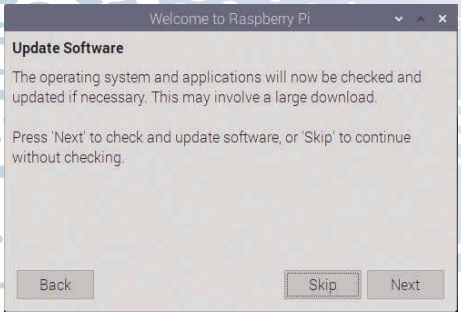
8 Bei manchen Bildschirmen kann es passieren, dass sie nicht automatisch komplett ausgenutzt werden und an zwei oder allen vier Rändern schwarze Balken erscheinen. Schalte in solchen Fällen das Kästchen bei ***This screen shows a black border around the desktop*** ein. Wenn das Desktophintergrundbild den ganzen Bildschirm ausfüllt, klicke einfach auf ***Next***.



9 Wenn du keine Möglichkeit hast, ein Netzwerkkabel an den Raspberry Pi anzuschließen, kannst du auch WLAN nutzen. Das nächste Fenster des Konfigurationsassistenten zeigt alle WLANs in der Nähe. Wähle dein WLAN aus und gib danach das Passwort dafür ein. Hast du eine Verbindung über ein Netzwerkkabel, klicke einfach auf ***Next***.



10 Im letzten Schritt kannst du die vorinstallierte Software automatisch aktualisieren lassen, was aber nicht unbedingt nötig ist. Möchtest du gleich loslegen, klicke einfach auf ***Next***.

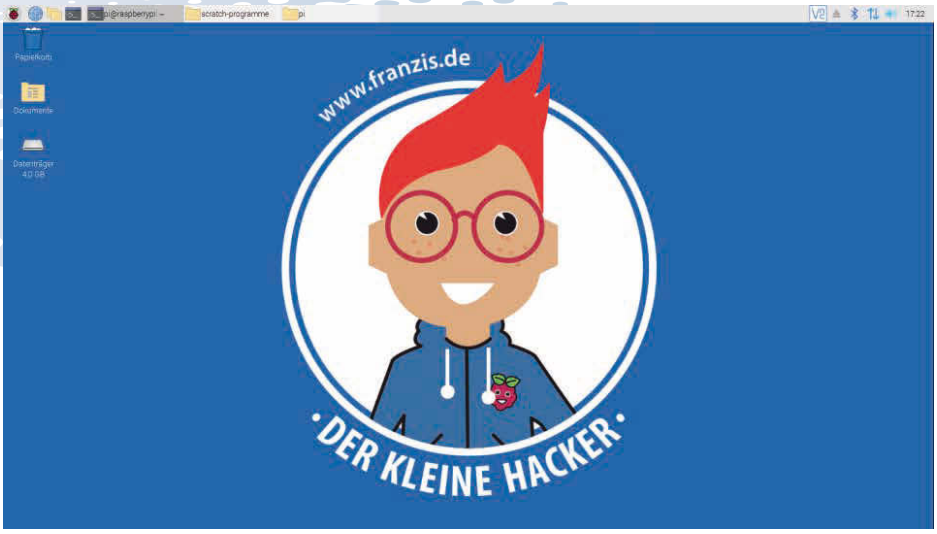


11 Der Raspberry Pi startet neu und zeigt die Benutzeroberfläche jetzt auf Deutsch. Es erscheint ein Desktop, der Windows relativ ähnlich sieht, jedoch mit dem Unterschied, dass Taskleiste und Startmenü am oberen Bildschirmrand erscheinen.

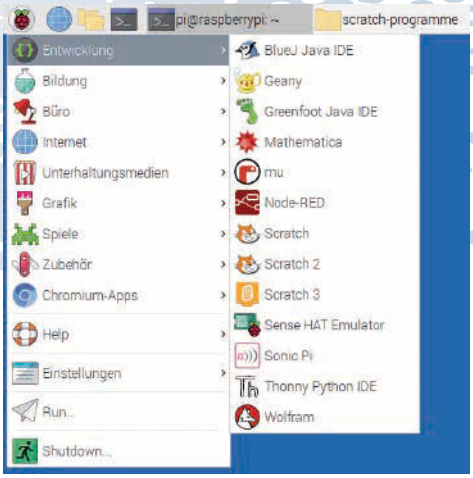
12 Das Menüsymbol mit der Himbeere ganz links oben öffnet das Startmenü, die Symbole daneben den Webbrowser und den Dateimanager. Das Startmenü ist wie unter Windows mehrstufig aufgebaut. Um die Experimente in diesem Kapitel nachzubauen und zu programmieren, brauchst du keines der Programme im Startmenü. Wir verwenden nur Scratch.

13 Starte Scratch 3 aus dem Startmenü unter ***Entwicklung***. Wenn du keinen Raspberry Pi 4, sondern ein Vorgängermodell hast, verwende Scratch 2.0.

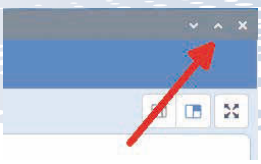
15 Elektronik mit Scratch auf dem Raspberry Pi steuern



15 Mit dem zweiten Symbol rechts oben kannst du das farbige Scratch-Fenster auf die volle Bildschirmgröße bringen, um mehr Platz zum Programmieren zu haben. Möchtest du zwischendurch die Meldungen im Textfenster sehen, kannst du wie unter Windows auf das Symbol ***Scratch 3*** in der Taskleiste klicken oder einfach mit der Tastenkombination Alt + Tab zwischen den Fenstern hin und her schalten.



14 Wie unter Windows kannst du die Fenster über den Bildschirm schieben, indem du sie mit der Maus an der Titelzeile anfasst.



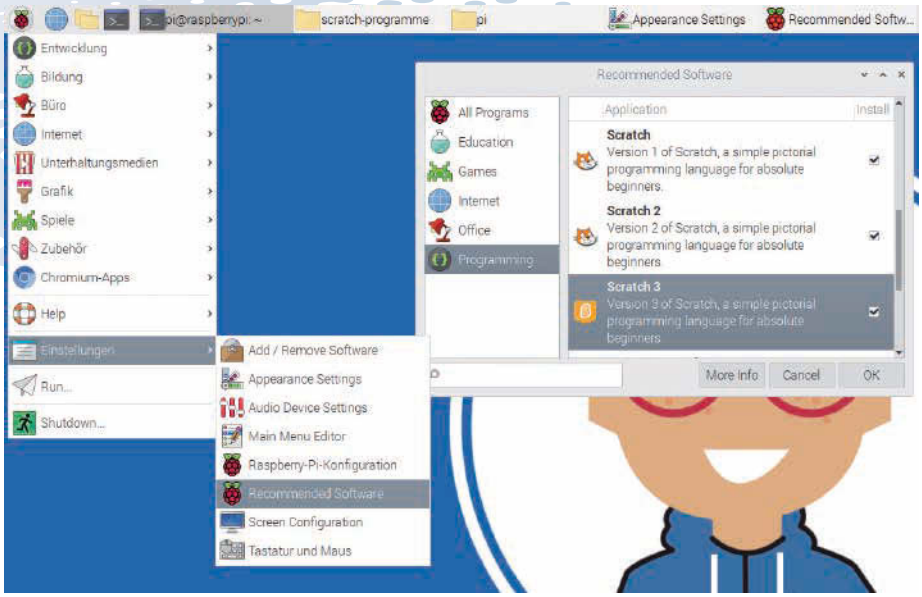
SCRATCH 3 NACHINSTALLIEREN

Wenn du eine Raspbian-Version verwendest, auf der Scratch 3 noch nicht vorinstalliert ist, brauchst du nicht gleich ein neues Betriebssystem. Du kannst Scratch 3 ganz einfach nachinstallieren.

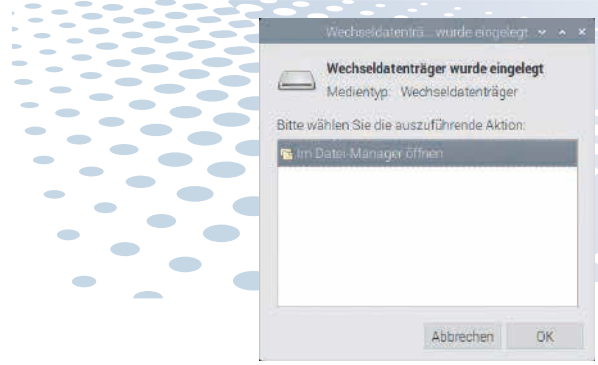
1 Wähle im Startmenü ***Einstellungen/Recommended Software***. Dort empfiehlt die Raspberry Pi-Stiftung einige nützliche Programme und bietet auch gleich Installationspakete zum Download an.

2 Suche unter ***Programming*** das Programm ***Scratch 3***, setze das Häkchen rechts und klicke auf ***OK***. Scratch 3 wird heruntergeladen und auch gleich automatisch installiert.

3 Nach der Installation findest du es über den Menüpunkt ***Entwicklung*** im Startmenü.



SCRATCH-PROGRAMME AUS DEM DOWNLOAD NUTZEN



Wenn du die Programme zum Buch bereits auf den PC heruntergeladen hast, kannst du sie mit einem USB-Stick auf den Raspberry Pi übertragen. Natürlich kannst du sie auch einfach anhand der Abbildungen im Buch selbst auf dem Raspberry Pi erstellen.

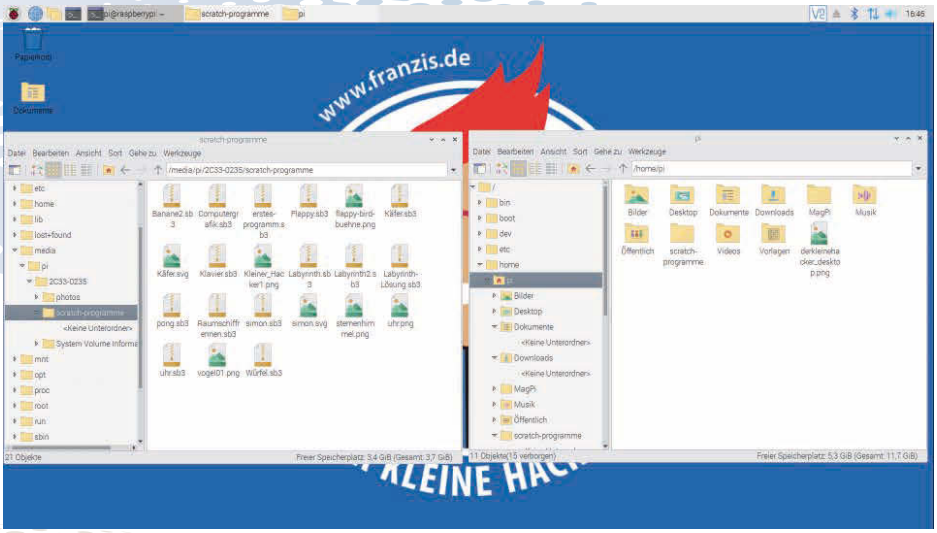
1. Kopiere auf dem PC den entpackten Ordner ***scratch-programme*** aus dem Download auf einen USB-Stick.

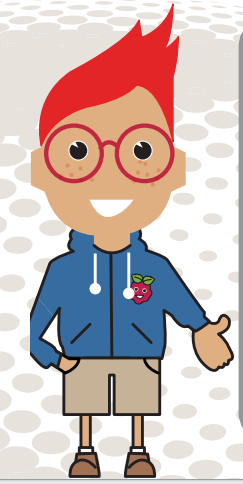
2. Stecke den USB-Stick an den Raspberry Pi. Dort erscheint das abgebildete Fenster. Klicke hier auf ***OK***.

3. Automatisch öffnet sich ein Dateimanager, der dem Explorer von Windows sehr ähnlich sieht.

4. Öffne mit einem Klick auf das Dateimanagersymbol in der Taskleiste ein weiteres Dateimanagerfenster. Dieses startet automatisch in deinem Home-Verzeichnis ***/home/pi***.

15 Elektronik mit Scratch auf dem Raspberry Pi steuern





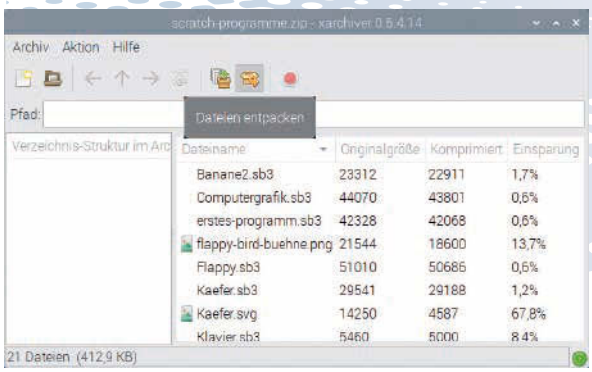
RASPBERRY PI AUSSCHALTEN

Theoretisch kann man bei dem Raspberry Pi einfach den Stecker ziehen, und er schaltet sich ab. Besser ist es jedoch, ihn wie einen PC sauber herunterzufahren. Wähle dazu im Startmenü *Shutdown* und klicke im nächsten Fenster erneut auf *Shutdown*.

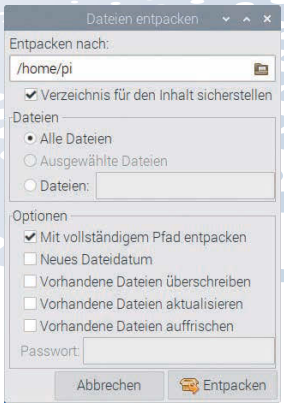
SCRATCH-PROGRAMME AUF DEN RASPBERRY PI HERUNTERLADEN

Du kannst die Scratch-Programme auch mit dem Browser auf den Raspberry Pi herunterladen. Die heruntergeladenen Dateien befinden sich dann im Ordner ***Downloads*** in deinem Home-Verzeichnis.

6 Klicke doppelt auf die heruntergeladene Zip-Datei. Das Entpackprogramm ***xarchiver*** öffnet sich automatisch und zeigt den Inhalt des Archivs an.



5 Ziehe den Ordner vom USB-Stick in das Dateimanager-Fenster, das dein Home-Verzeichnis zeigt.



2 Klicke auf das Symbol ***Dateien entpacken***.

3 Trage im nächsten Fenster oben dein Home-Verzeichnis ***/home/pi*** ein und klicke auf ***Entpacken***. In deinem Home-Verzeichnis wird automatisch ein Unterordner mit den Programmen angelegt.

DAS HINTERGRUNDBILD

Möchtest du auch das Hintergrundbild mit dem kleinen Hacker nutzen, das auf den Bildern in diesem Buch abgebildet ist? Du findest es im Download.

1 Klicke mit der rechten Maustaste auf den Desktop und wähle im Kontextmenü ***Desktop-Einstellungen***.

2 Klicke auf die Schaltfläche neben ***Picture***. Hier steht der Dateiname des gerade verwendeten Hintergrundbilds. Jetzt kannst du das Bild ***derkleinehacker_desktop.png*** aus dem entpackten Downloadarchiv auswählen.

3 Wähle bei ***Layout*** noch die Option ***Fill screen with image***, damit das Bild den gesamten Bildschirm ausfüllt. Verlasse dieses Fenster mit ***OK***.

DIESE TEILE BRAUCHST DU FÜR DIE HARDWAREEXPERIMENTE

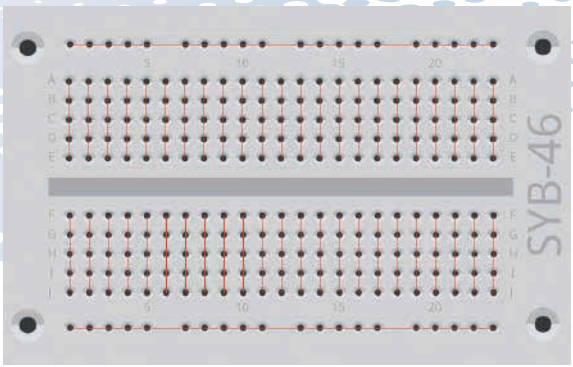
Wir verwenden für die Experimente in diesem Buch gängige Bauteile, die im Elektronikhandel oder in den Ersatzteilkisten von Elektronikbastlern leicht zu finden sind.

STECKBRETTER

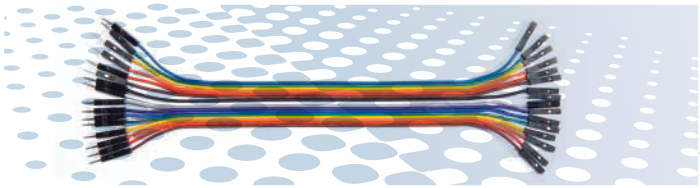
Für den schnellen Aufbau elektronischer Schaltungen, ohne löten zu müssen, verwendet man Steckbretter. Dort können elektronische Bauteile direkt in ein Lochraster eingesteckt werden.

Bei größeren Steckbrettern sind die äußeren Längsreihen über Kontakte (X und Y) alle miteinander verbunden. Diese Kontaktreihen werden oft als Plus- und Minuspol zur Stromversorgung der Schaltungen genutzt. In den anderen Kontaktreihen

sind jeweils fünf Kontakte (A bis E und F bis J) quer miteinander verbunden, wobei in der Mitte der Platine eine Lücke ist. So können dort größere Bauelemente eingesteckt und nach außen hin verdrahtet werden. Kleine Steckbretter haben keine Längsreihen. Das Bild zeigt, welche Löcher auf dem Steckbrett miteinander verbunden sind.

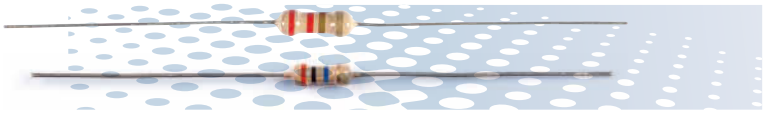


VERBINDUNGSKABEL



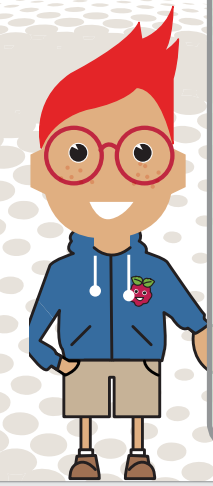
Die Verbindungskabel zwischen Raspberry Pi und Steckbrett brauchen alle auf einer Seite einen dünnen Drahtstecker, mit dem sie sich auf dem Steckbrett einstecken lassen. Auf der anderen Seite muss eine Steckbuchse vorhanden sein, die auf einen GPIO-Pin des Raspberry Pi passt. Solche Kabel kann man entweder günstig kaufen, oder du verwendest den Kabelstrang für die Front-LEDs und die Reset-Taster aus einem ausgedienten Computergehäuse. Die Kontakte, mit denen diese Kabel am Motherboard angeschlossen sind, passen auf den GPIO-Port des Raspberry Pi.

WIDERSTÄNDE



Ein Widerstand begrenzt den Strom, der durch eine Leitung fließt. Man kann ihn sich vorstellen wie eine dünnes Stück Gartenschlauch, das in ein Wasserrohr eingebaut wird. Durch die Rohrleitung fließt dann im Ganzen weniger Wasser, nämlich nur noch so viel, wie durch den Schlauch hindurchkommt.

Widerstände werden zur Strombegrenzung an empfindlichen elektronischen Bauteilen sowie als Vorwiderstände für LEDs verwendet. Die Maßeinheit für Widerstände ist Ohm. 1.000 Ohm entsprechen einem Kiloohm, abgekürzt kOhm.



LÖTEN – EINFACH UND RICHTIG

Wer sich mit Hardwarebasteleien rund um den Raspberry Pi beschäftigt, wird ab und an auch mal etwas löten müssen. Für Profis ist das kein Problem, für Anfänger eigentlich auch nicht, wenn du ein paar wichtige Tricks beachtest. *Löten ist einfach* ist ein unterhaltsamer Comic mit Basiswissen für Hobbylöter: *bit.ly/178qobA*.

1.000 kOhm entsprechen einem Megaohm, abgekürzt MOhm. Oft wird für die Einheit Ohm auch das Omega-Zeichen Ω verwendet.

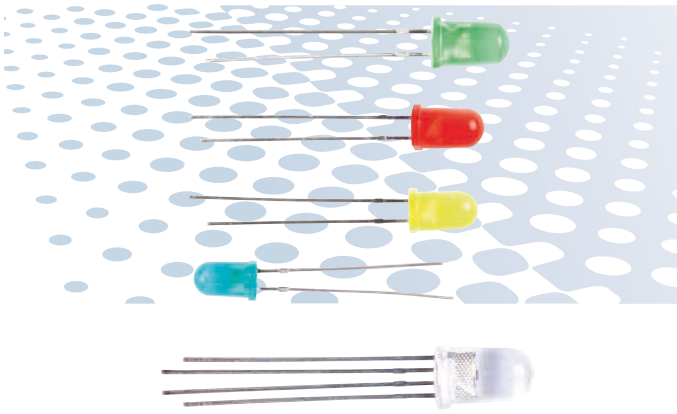
Die farbigen Ringe auf den Widerständen geben den Widerstandswert an. Mit etwas Übung sind sie deutlich leichter zu erkennen als die winzig kleinen Zahlen, die man nur noch auf ganz alten Widerständen findet.

Die meisten Widerstände haben vier solcher Farbringe. Die ersten beiden Farbringe bezeichnen die Ziffern, der dritte einen Multiplikator und der vierte die Toleranz. Dieser Toleranzring ist meistens gold- oder silberfarben, Farben, die auf den ersten Ringen nicht vorkommen. Dadurch ist die Leserichtung immer eindeutig. Der Toleranzwert selbst spielt in der Digitalelektronik kaum eine Rolle. Die Tabelle zeigt die Bedeutung der farbigen Ringe auf Widerständen.

In welcher Richtung ein Widerstand eingebaut wird, ist egal. Bei LEDs dagegen spielt die Einbaurichtung eine wichtige Rolle.

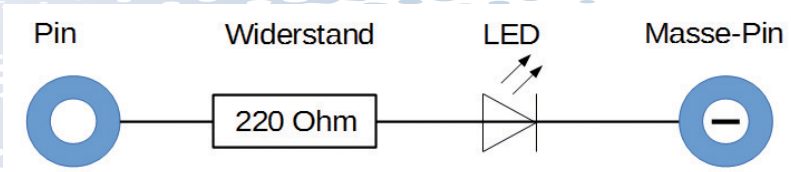
Farbe	Widerstandswert in Ohm			
	1. Ring (Zehner)	2. Ring (Einer)	3. Ring (Multiplikator)	4. Ring (Toleranz)
Silber			10^{-2} = 0,01	±10 %
Gold			10^{-1} = 0,1	±5 %
Schwarz		0	10^{0} = 1	
Braun	1	1	10^{1} = 10	±1 %
Rot	2	2	10^{2} = 100	±2 %
Orange	3	3	10^{3} = 1.000	
Gelb	4	4	10^{4} = 10.000	
Grün	5	5	10^{5} = 100.000	±0,5 %
Blau	6	6	10^{6} = 1.000.000	±0,25 %
Violett	7	7	10^{7} = 10.000.000	±0,1 %
Grau	8	8	10^{8} = 100.000.000	±0,05 %
Weiß	9	9	10^{9} = 1.000.000.000	

LEDS



LED ist die Abkürzung für das englische Wort ***Light Emitting Diode***, was wörtlich übersetzt „Licht abstrahlende Diode" oder einfach Leuchtdiode bedeutet. LEDs können schön bunt leuchten, wenn Strom in einer Richtung durch sie fließt. In der anderen Richtung lassen sie nichts durch und leuchten auch nicht.

LEDs werden in Schaltungen mit einem pfeilförmigen Dreieckssymbol dargestellt, das die Flussrichtung vom Pluspol zum Minuspol oder zur Masseleitung angibt.



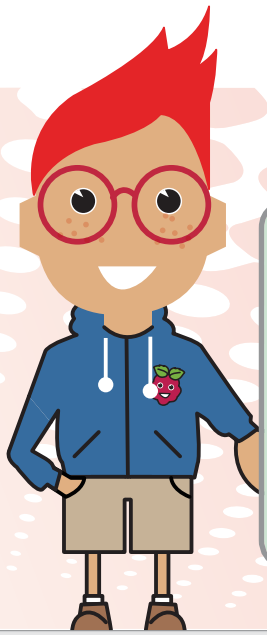
Schaltplan einer LED mit Vorwiderstand.

Eine LED lässt in Durchflussrichtung nahezu beliebig viel Strom durch, sie hat nur einen sehr geringen Widerstand. Um den Durchflussstrom zu begrenzen und damit ein Durchbrennen der LED zu verhindern, muss zwischen dem verwendeten GPIO-Pin und der Anode der LED ein 220-Ohm-Vorwiderstand (Rot-Rot-Braun) eingebaut werden. Dieser Vorwiderstand schützt auch den GPIO-Ausgang des Raspberry Pi vor zu hohen Stromstärken.

Es gibt LEDs, bei denen die Vorwiderstände bereits eingebaut sind. Sie sind in Onlineshops nur unwesentlich teurer, man spart sich aber den Aufwand, zusätzliche Vorwiderstände anzuschließen. Wenn du nicht sowieso LEDs herumliegen hast, besorge dir einfach welche mit eingebauten Vorwiderständen.

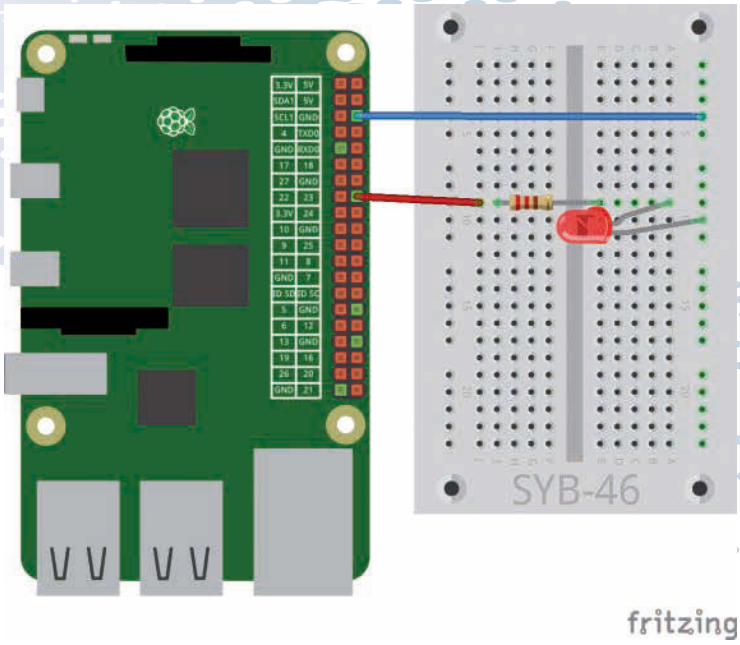
DIE ERSTE LED BLINKT

Jetzt geht es los. Nachdem der Raspberry Pi läuft, soll unser erstes Programm eine LED blinken lassen. Das klingt nicht weiter aufregend, liefert aber alles wichtige Fachwissen, um später weitere Elektronik anzuschließen. Im Vergleich zum Versuch, mit einem Windows-PC eine extern angeschlossene LED zum Blinken zu bringen, ist alles ganz einfach.

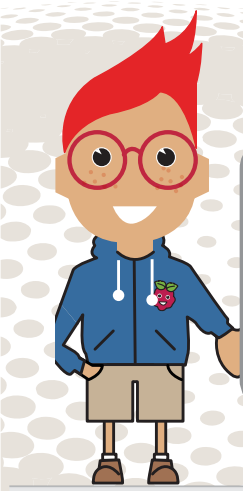


LED IN WELCHER RICHTUNG ANSCHLIESSEN?

Die beiden Anschlussdrähte einer LED sind unterschiedlich lang. Der längere ist der Pluspol, die Anode, der kürzere die Kathode. Einfach zu merken: Das Pluszeichen hat einen Strich mehr als das Minuszeichen und macht damit den Draht quasi optisch etwas länger. Außerdem sind die meisten LEDs auf der Minusseite abgeflacht, wie ein Minuszeichen. Leicht zu merken: Kathode = kurz = Kante.



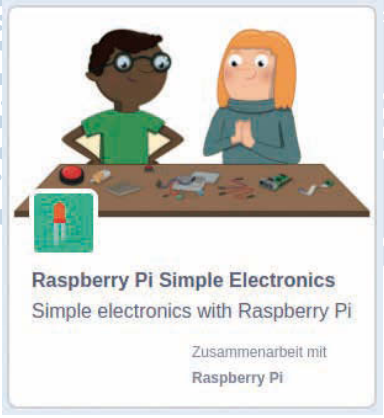
Baue die LED und den Vorwiderstand wie auf dem Bild gezeigt auf dem Steckbrett auf und schließe das Ganze am GPIO-Pin 23 und der Masseleitung am GND-Pin des Raspberry Pi an. Das gleiche Prinzip wird auch in den nächsten Experimenten beim Anschließen von LEDs angewandt.



VERWENDETE BAUTEILE

- 1 Steckbrett
- 1 LED
- 1 220-Ohm-Widerstand (Rot-Rot-Braun)
- 2 GPIO-Verbindungskabel

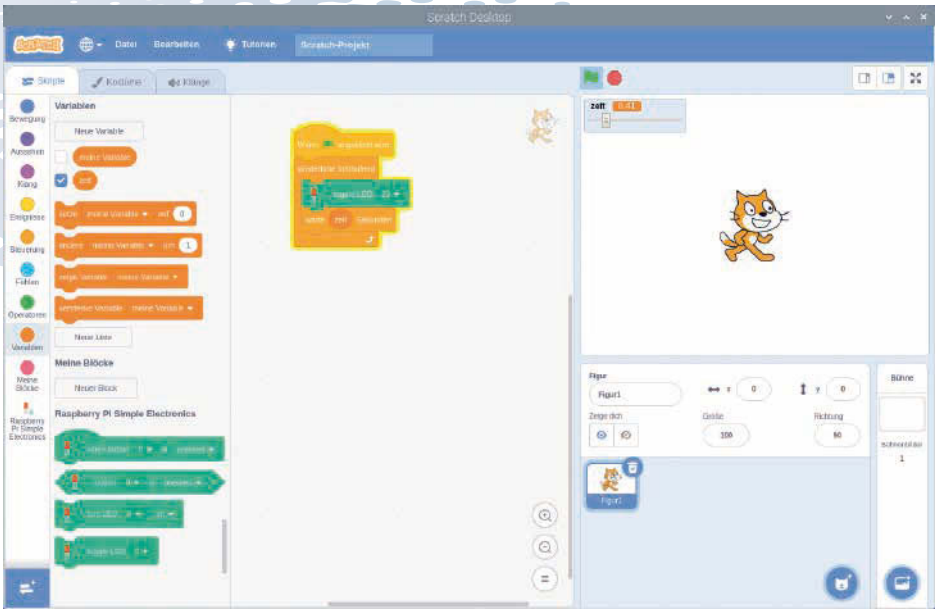
0 Starte Scratch 3 und installiere als Erstes die Erweiterung ***Raspberry Pi Simple Electronics***. Auf einer neuen Blockpalette ganz unten erscheinen vier Scratch-Blöcke zur Steuerung von Tastern und LEDs.



2 Lege in Scratch als Erstes auf der Blockpalette ***Variablen*** eine neue Variable ***zeit*** an, die auf der Bühne sichtbar ist. Richte für diese Variable einen Schieberegler mit einem Bereich von ***0.1*** bis ***2*** ein. Er stellt die Zeit ein, die die LED beim Blinken leuchten soll.

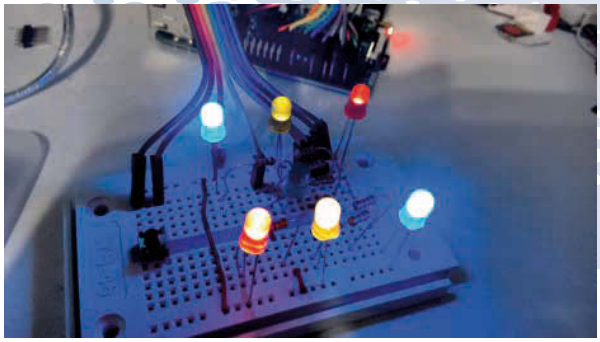
3 Baue das Programm wie in der Abbildung zusammen. Der Block ***toggle LED*** schaltet bei jedem Aufruf eine LED um – von aus auf ein oder umgekehrt. Im Zahlenfeld in diesem Block wählst du den verwendeten GPIO-Pin aus. Im abgebildeten Beispiel ist das die ***23***.

15 Elektronik mit Scratch auf dem Raspberry Pi steuern



Baue als Erstes die Schaltung auf dem Steckbrett mit sieben LEDs und Vorwiderständen und einem Taster auf. Für die Ansteuerung der LEDs werden nur vier statt sieben GPIO-Pins benötigt, da ein Würfel zur Darstellung gerader Zahlen die Augen paarweise nutzt. Die LEDs sind in der Abbildung paarweise in gleichen Farben dargestellt.

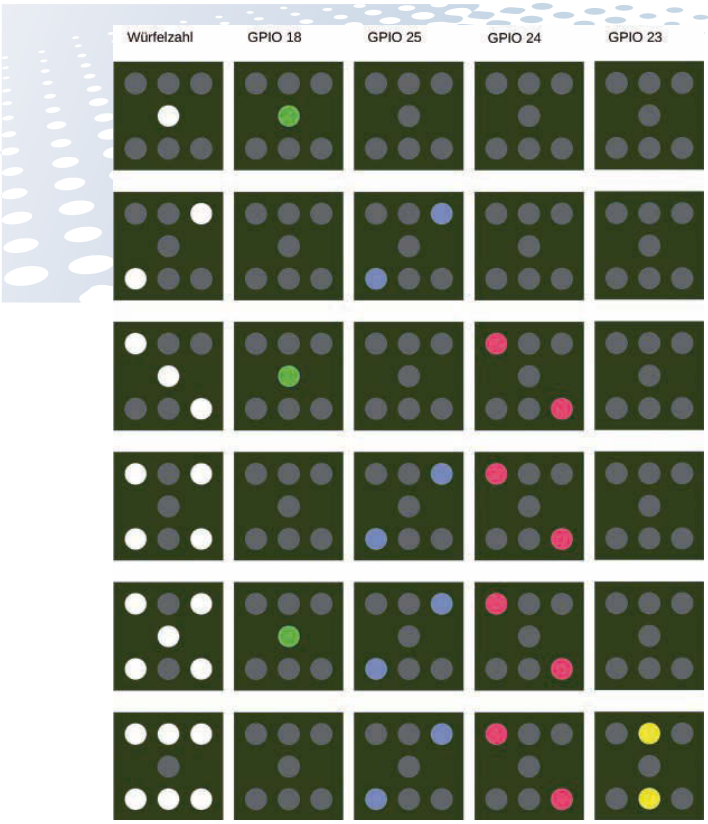
4 Klicke auf das grüne Fähnchen. Die LED blinkt. Jetzt kannst du mit dem Schieberegler die Geschwindigkeit einstellen.



Vergiss nicht, das fertige Programm mit dem Menüpunkt ***Datei/Auf deinem Computer speichern*** zu speichern, um es später wieder verwenden zu können.

LED-WÜRFEL MIT SCRATCH

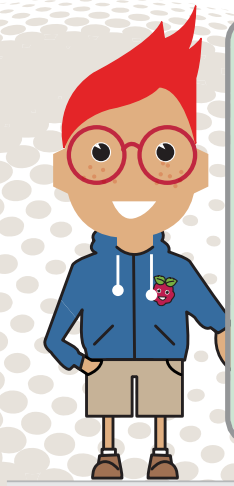
Eine einzelne LED ein- und wieder auszuschalten mag im ersten Moment ganz spannend sein, aber dafür braucht man eigentlich keinen Computer. Das nächste Programm steuert einen Würfel, der aus sieben LEDs besteht. Die verwendete Logik ist ähnlich der des Würfelprogramms weiter oben in diesem Buch.



In Scratch 3 brauchst du wieder die Erweiterung ***Raspberry Pi Simple Electronics***. Installiere sie und baue dann das Programm zusammen.

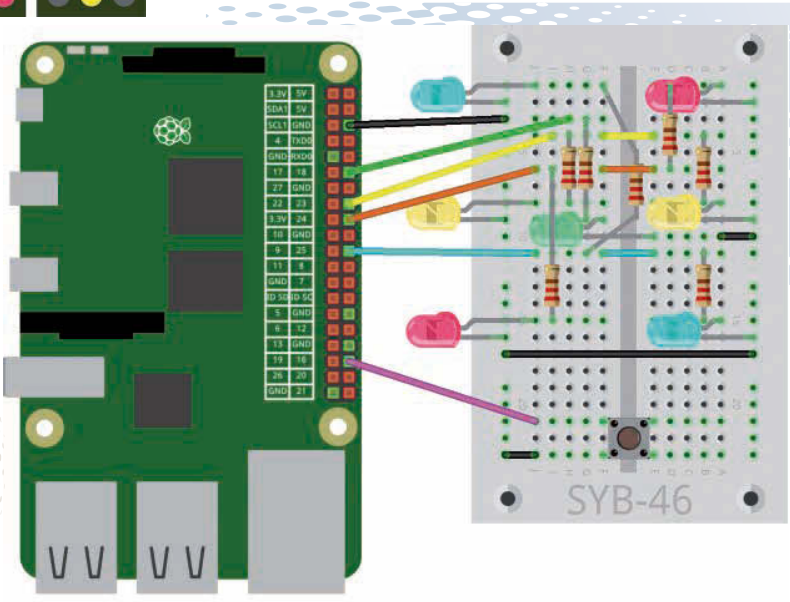
1 Der Block ***when button ... is ...*** in Scratch 3 startet eine Aktion, wenn ein Taster am angegebenen Pin, in unserem Beispiel Pin 16, gedrückt wird. Wähle diesen Pin im Zahlenfeld des Blocks aus.

2 Als Nächstes werden die vier GPIO-Ports mit den LEDs ausgeschaltet. Am Anfang sind die LEDs alle aus, aber später bleibt ein angezeigtes Würfelergebnis so lange bestehen, bis der Benutzer wieder die Taste drückt. Ziehe dazu vier Blöcke ***turn LED ...*** in das Programm, wähle in jedem einen der Pins aus, an denen die LEDs angeschlossen sind, und schalte das Feld ganz rechts auf ***off***.



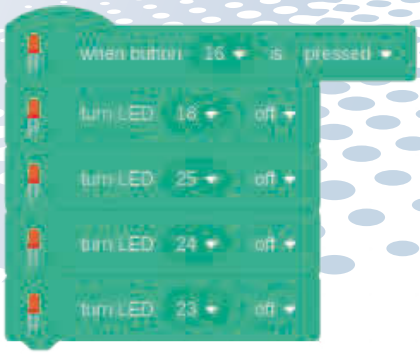
VERWENDETE BAUTEILE

- 1 Steckbrett
- 7 LEDs
- 7 220-Ohm-Widerstände (Rot-Rot-Braun)
- 1 Taster
- 6 GPIO-Verbindungskabel
- 6 Drahtbrücken (unterschiedliche Längen)



fritzing

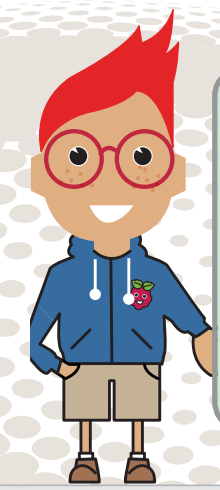
15 Elektronik mit Scratch auf dem Raspberry Pi steuern



3 Danach wird eine zufällige Zahl zwischen 1 und 6 erzeugt und in der Variablen ***w*** gespeichert. Lege diese Variable auf der Blockpalette ***Variablen*** an. Lasse sie auch auf der Bühne anzeigen. Dann kannst du jederzeit prüfen, ob die LEDs auch tatsächlich das richtige Würfelergebnis anzeigen.

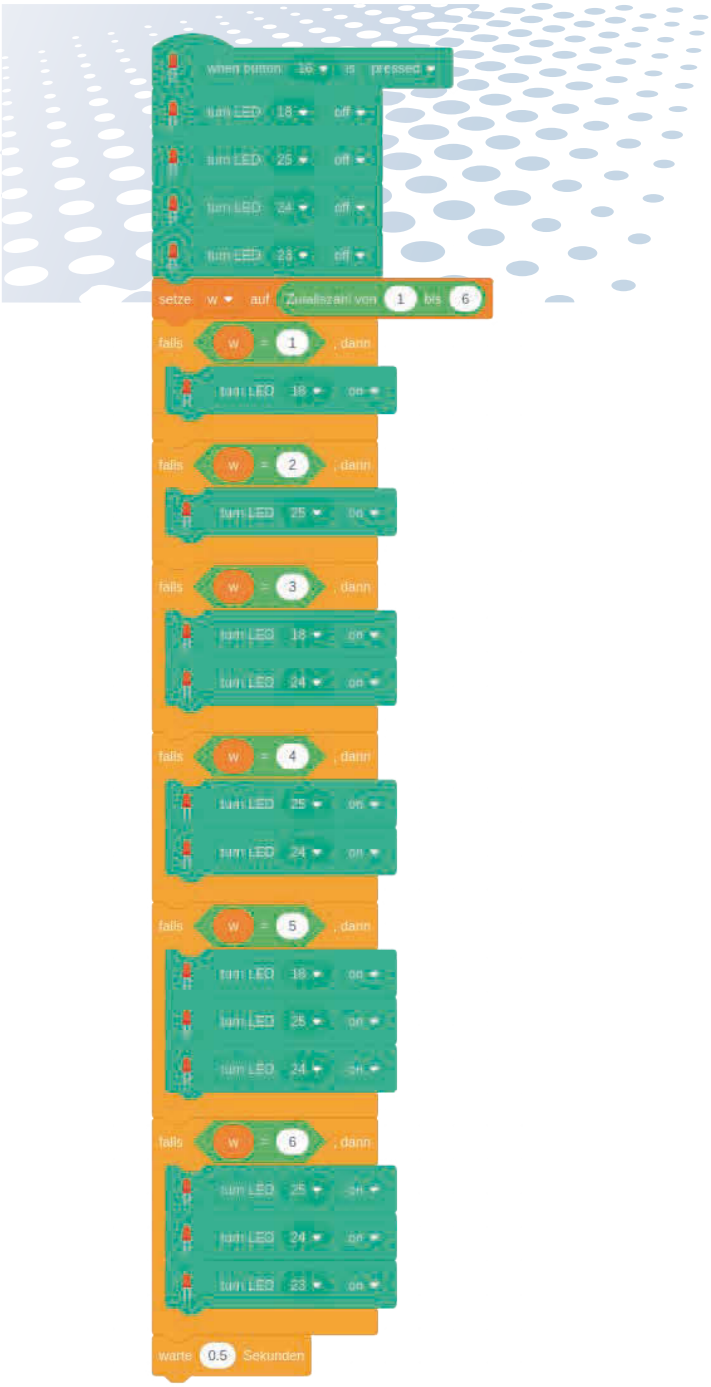
4 Nachdem die Zahl gewürfelt wurde, folgen sechs ***falls ..., dann***-Blöcke für jeden möglichen Würfelwert. Jeder dieser Blöcke schaltet, wenn eine bestimmte Zahl gewürfelt wurde, die entsprechenden LEDs ein.

5 Unabhängig vom Würfelergebnis wartet das Programm am Ende, immer wenn der Benutzer eine Taste gedrückt hat, noch eine halbe Sekunde, um zu vermeiden, dass durch Tastenprellen kurz hintereinander zwei verschiedene Würfelaktionen ausgelöst werden.



ANSCHLUSS DER TASTER

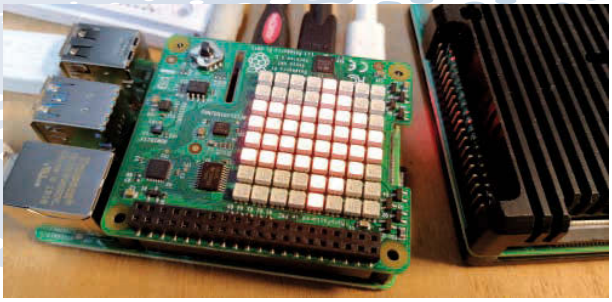
Taster gelten in Scratch 3 als gedrückt, wenn sie mit Masse verbunden sind, nicht mit +3,3 V, wie in einigen anderen Programmiersprachen. Deshalb verbindet der Taster in der Schaltung Pin 16 mit der Masseleitung auf dem Steckbrett.



6 Nachdem die LEDs eingeschaltet wurden, endet das Programm. Solange du die Taste nicht drückst, passiert einfach nichts, und das letzte Würfelergebnis bleibt angezeigt. Drückst du die Taste, startet das Programm, schaltet alle LEDs aus und würfelt dann eine neue Zahl. Das grüne Fähnchen wird in diesem Programm nicht benötigt.

16 Das SenseHAT für den Raspberry Pi

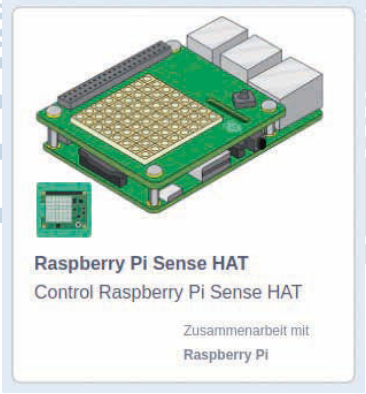
Das SenseHAT ist eine von der Raspberry-Pi-Stiftung entwickelte Aufsteckplatine für den Raspberry Pi. Es enthält eine 8-×-8-Matrix aus RGB-LEDs, einen kleinen Joystick sowie verschiedene Sensoren. Das Modul wurde für die Astro-Pi-Mission entwickelt und arbeitet seit Dezember 2015 auf der Internationalen Raumstation ISS. Es ist mittlerweile auch im Handel erhältlich.



Auf dem SenseHAT sind fest eingebaut:

- 8-×-8-Matrix aus RGB-LEDs
- Joystick, auch als Taster nutzbar
- Lagesensor
- Beschleunigungssensor
- Magnetometer
- Temperatursensor
- Feuchtigkeitssensor
- Barometer

Mit dem SenseHAT brauchst du weder Bauteile einzeln zu stecken, noch musst dich um die Steuerung der einzelnen GPIO-Anschlusspins kümmern. Die Erweiterung ***Raspberry Pi SenseHAT*** für Scratch 3 steuert alle Komponenten auf der SenseHAT-Platine.



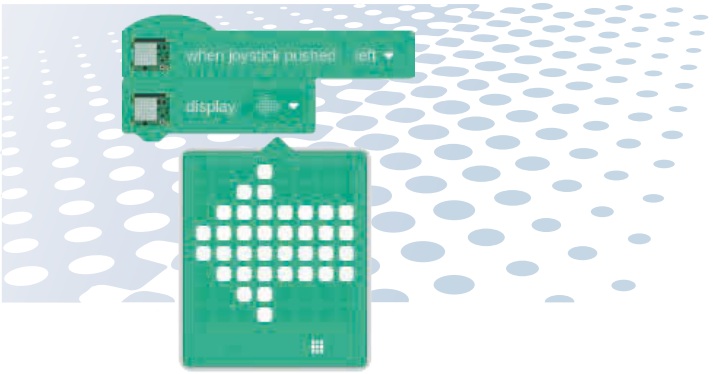
Zwei Programme zeigen ein paar der Möglichkeiten, die das SenseHAT bietet.

JOYSTICK UND SYMBOLE AUF DEM SENSEHAT

Drückst du den Joystick in eine Richtung, zeigt das SenseHAT einen Pfeil in diese Richtung an.

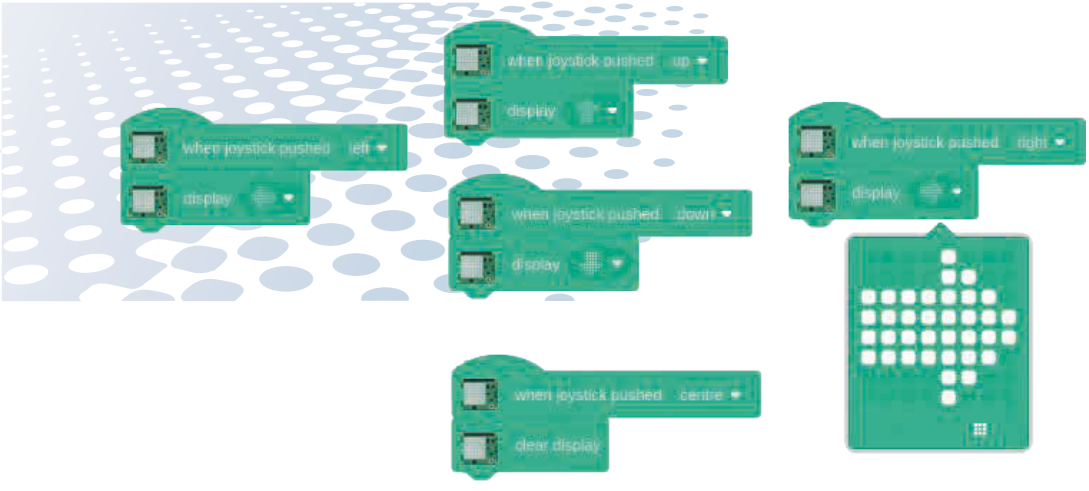
Der Block ***when joystick pushed ...*** aus der Erweiterung ***Raspberry Pi SenseHAT*** startet die angehängten Programmblöcke, wenn der Joystick in eine

wählbare Richtung gedrückt wird. Dabei bedeutet ***centre***, dass der Joystick wie eine Taste gedrückt wird, ohne dass man ihn in eine Richtung bewegt.



Der Block ***display ...*** zeigt ein beliebiges Punktmuster auf der LED-Matrix an. Klicke auf das Feld mit dem Muster rechts im Block. Es öffnet sich eine größere Darstellung. Hier kannst du durch Anklicken einzelne LEDs ein- oder ausschalten. Der Block stellt dann das so entworfene Muster auf der LED-Matrix dar.

Hänge diesen Block unter den Block ***when joystick pushed ...*** und zeichne als Muster einen Pfeil, der in die gleiche Richtung zeigt wie der Joystick im oberen Block. Baue dann diese Blöcke für alle vier Richtungen.



Wenn der Joystick in der Mitte gedrückt wird, soll die Anzeige ausgeschaltet werden. Hänge dazu an einen Block ***when joystick pushed centre*** einen Block ***clear display***. Dieser schaltet die ganze LED-Matrix aus.

Das grüne Fähnchen wird auch in diesem Programm nicht gebraucht. Alle Aktionen werden durch Drücken des Joysticks ausgelöst.

TEMPERATUR UND TEXT AUF DEM SENSEHAT

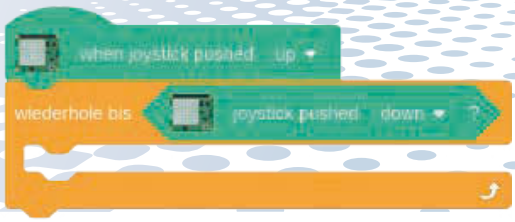
Das SenseHAT hat einen eingebauten Temperatursensor. Das nächste Programm zeigt als Laufschrift immer wieder die aktuelle Temperatur an. Das Programm wird gestartet, wenn du den Joystick nach oben drückst, und hält wieder an, wenn du ihn nach unten drückst. Kühle Temperaturen werden in Grün angezeigt, mittlere in Gelb und heiße in Rot.

Das Programm beginnt wieder mit einem Block ***when joystick pushed up***. Danach starte eine ***wiederhole bis ...***-Schleife, die so lange laufen soll, bis du den Joystick wieder nach unten drückst.

Um das abzufragen, gibt es den Block ***joystick***

16 Das SenseHAT für den Raspberry Pi

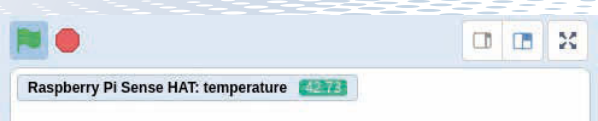
pushed ..., der den Wert ***wahr*** zurückgibt, wenn der Joystick in eine bestimmte Richtung gedrückt wurde, während das Programm läuft.



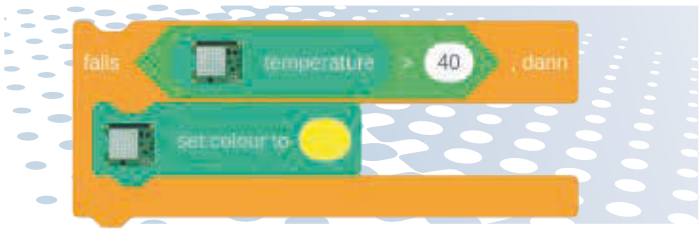
In jedem Schleifendurchlauf wird als Erstes die Farbe aller LEDs auf Grün gesetzt. Ziehe dazu den Block ***set colour to ...*** in die Schleife und stelle im Farbfeld dieses Blocks ein leuchtendes Grün ein.

Danach wird geprüft, ob die Temperatur über 40 Grad ist. In diesem Fall sollen LEDs gelb leuchten. Dazu verwenden wir einen ***falls ..., dann***-Block. Im Feld für die Bedingung dieses Blocks wird die Temperatur abgefragt und geprüft, ob sie größer als 40 ist. Der Block ***temperature*** aus der Erweiterung ***Raspberry Pi SenseHAT*** liefert jederzeit die aktuelle Temperatur und kann wie eine Variable überall eingesetzt werden, wo eine Zahl gefragt ist.

Wie bei einer Variablen kannst du auch hier das Häkchen setzen, um die Temperatur auf der Bühne in Echtzeit anzuzeigen.

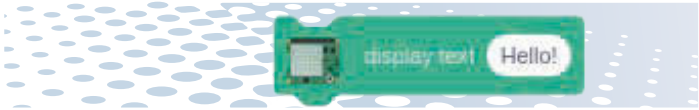


Ist die Temperatur größer als 40, setzt ein Block ***set colour to ...*** die Farbe der LEDs auf Gelb.



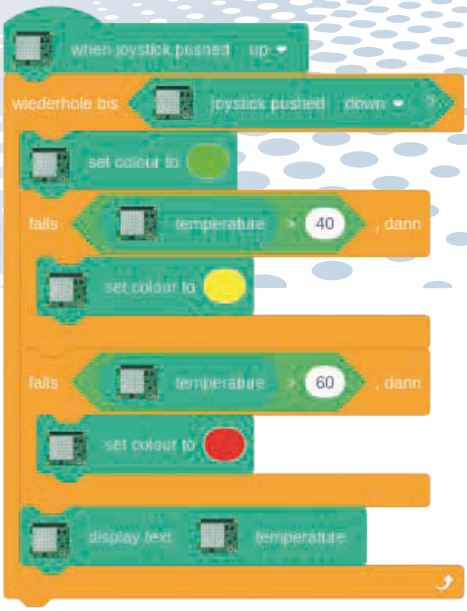
Auf die gleiche Weise wird abgefragt, ob die Temperatur über 60 Grad ist. In dem Fall wird es dem Raspberry Pi wirklich zu heiß. Er hat dann selbst eine noch höhere Temperatur, wenn das SenseHAT schon 60 Grad fühlt.

Nachdem die Farbe eingestellt ist, wird die Temperatur angezeigt. Die Erweiterung ***Raspberry Pi SenseHAT*** liefert dazu einen Block ***display text ...*** Hier kannst du automatisch einen beliebigen Text als Laufschrift auf der LED-Matrix anzeigen lassen.



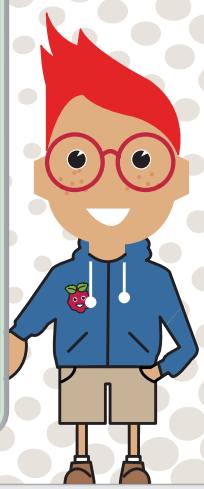
Ziehe den Block ***temperature*** in das Textfeld dieses Blocks, um die Temperatur anzuzeigen.

Baue das Programm zusammen und drücke den Joystick nach oben. Automatisch erscheint die aktuelle Temperatur als Laufschrift. Das grüne Fähnchen wird auch in diesem Programm nicht gebraucht. Um das Programm wieder anzuhalten, musst du den Joystick eventuell etwas länger nach unten drücken, da er nur einmal in jedem Schleifendurchlauf abgefragt wird, wenn die Laufschrift durchgelaufen ist.



TEMPERATUR DES SENSEHAT

Wundere dich nicht, wenn das SenseHAT ständig Temperaturen über 40 Grad anzeigt, auch wenn dein Zimmer deutlich kühler ist. Das ist völlig normal. Das SenseHAT liegt nur etwa einen Zentimeter oberhalb der Raspberry-Pi-Platine und bekommt die Wärme des Prozessors und der anderen Chips deutlich zu spüren. Du kannst es sogar mit dem Finger kühlen, wenn du den mit *Humidity* beschrifteten Chip berührst. Das ist der Sensor für Temperatur und Luftfeuchtigkeit.

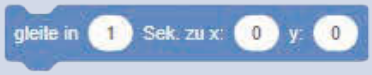


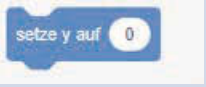
REFERENZ: ALLE SCRATCH-BLÖCKE IM ÜBERBLICK

Die folgenden Tabellen zeigen alle in Scratch 3.0 verfügbaren Blöcke. Die grau hinterlegten Blöcke sind in Scratch 2.0 nicht enthalten.

BEWEGUNG

Die Blöcke der Blockpalette ***Bewegung*** bewegen die aktuelle Figur oder helfen dabei, deren Position zu ermitteln. Bei Bewegungen unterscheidet man – nicht nur in Scratch – zwischen absoluten Bewegungen, die sich auf das Koordinatensystem beziehen, und relativen Bewegungen, die sich auf die aktuelle Position der zu bewegenden Figur beziehen.

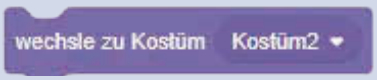
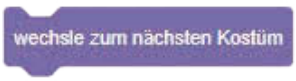
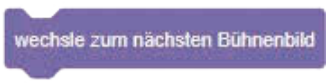
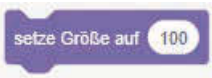
	relativ	Bewege die Figur um eine in diesem Block angegebene Entfernung in die eingestellte Richtung.
	relativ	Drehe die Figur im Uhrzeigersinn um den in diesem Block angegebenen Winkel. Die Bewegungsrichtung für die folgenden Bewegungen wird mit gedreht.
	relativ	Drehe die Figur gegen den Uhrzeigersinn um den in diesem Block angegebenen Winkel. Die Bewegungsrichtung für die folgenden Bewegungen wird mit gedreht.
	relativ	Bewege die Figur zu einer ausgewählten anderen Figur, zu einer zufälligen Position oder zum Mauszeiger.
	absolut	Bewege die Figur zu einem angegebenen Koordinatenpunkt.
	absolut	Bewege die Figur in einer angegebenen Zeit zu einer ausgewählten anderen Figur, zu einer zufälligen Position oder zum Mauszeiger.
	absolut	Bewege die Figur in einer angegebenen Zeit zu einem angegebenen Koordinatenpunkt.
	absolut	Setze die Bewegungsrichtung auf den angegebenen Winkel. Die Figur wird ebenfalls gedreht.
	relativ	Drehe die Figur so, dass ihre Richtung auf eine ausgewählte andere Figur oder zum Mauszeiger zeigt.

	relativ	Bewege die Figur um den angegebenen Wert in x-Richtung. Die y-Koordinate bleibt unverändert.
	absolut	Bewege die Figur an den angegebenen Punkt in x-Richtung. Die y-Koordinate bleibt unverändert.
	relativ	Bewege die Figur um den angegebenen Wert in y-Richtung. Die x-Koordinate bleibt unverändert.
	absolut	Bewege die Figur an den angegebenen Punkt in y-Richtung. Die x-Koordinate bleibt unverändert.
	relativ	Wenn die Figur bei ihrer Bewegung den Rand der Bühne berührt, prallt sie im gleichen Winkel vom Rand ab, wie sie zum Rand hingekommen ist.
	absolut	Setzt den Drehtyp der Figur. Er legt fest, ob sich eine Figur bei einer Drehbewegung rundherum dreht, nur nach links und rechts oder gar nicht. Die gleiche Einstellung lässt sich mit einem Klick auf das blaue i-Symbol der Figur manuell vornehmen.
	absolut	Zeigt die x-Position der Figur an. Mit diesem Wert kann wie mit einer Variablen gerechnet werden.
	absolut	Zeigt die y-Position der Figur an. Mit diesem Wert kann wie mit einer Variablen gerechnet werden.
	absolut	Zeigt die Richtung der Figur an. Mit diesem Wert kann wie mit einer Variablen gerechnet werden.

AUSSEHEN

Die Blöcke auf der Blockpalette ***Aussehen*** beeinflussen das Aussehen der aktuellen Figur – Kostüme, Farbeffekte, Größe usw. Hier sind auch Blöcke zu finden, die das Bühnenbild beeinflussen und Meldungen an den Benutzer ausgeben.

	Zeigt einen Text in einer Sprechblase an der aktuellen Figur an. Das Skript wartet eine angegebene Zeit, danach wird die Sprechblase ausgeblendet, und das Skript läuft weiter.
	Zeigt einen Text in einer Sprechblase an der aktuellen Figur an. Die Sprechblase bleibt sichtbar, bis ein ***sage ...***-Block mit einem leeren Textfeld folgt.

	Zeigt einen Text in einer Denkblase an der aktuellen Figur an. Das Skript wartet eine angegebene Zeit, danach wird die Denkblase ausgeblendet, und das Skript läuft weiter.
	Zeigt einen Text in einer Denkblase an der aktuellen Figur an. Die Sprechblase bleibt sichtbar, bis eine Denkblase ***denke ...***-Block mit einem leeren Textfeld folgt.
	Wechselt zum ausgewählten Kostüm für die aktuelle Figur.
	Wechselt zum nächsten Kostüm in der Liste für die aktuelle Figur.
	Wechselt zum ausgewählten Bühnenbild.
	Wechselt zum nächsten Bühnenbild in der Liste.
	Ändert die Größe der aktuellen Figur um einen bestimmten Wert.
	Setzt die Größe der aktuellen Figur auf einen bestimmten Prozentwert gegenüber dem Original, unabhängig davon, ob die Größe bereits vorher verändert wurde.
	Ändert einen von sieben auswählbaren Grafikeffekten für die aktuelle Figur um einen angegebenen Wert. Bei Klick auf das rote Stoppsymbol werden alle Grafikeffekte wieder zurückgesetzt.

	Setzt einen von sieben auswählbaren Grafikeffekten für die aktuelle Figur auf einen angegebenen Wert. Im Gegensatz zum Block ***ändere … Effekt um …*** spielt es dabei keine Rolle, auf welchem Wert der Effekt vorher stand.
	Schaltet alle Grafikeffekte für die aktuelle Figur aus. Die Figur sieht wieder so aus, wie im aktuellen Kostüm festgelegt. Dieser Block bewirkt das Gleiche, das passiert, wenn alle Effekte auf 0 gesetzt werden.
	Zeigt die aktuelle Figur auf der Bühne, wenn sie vorher versteckt war.
	Versteckt die aktuelle Figur auf der Bühne. Eine versteckte Figur kann weiterhin bewegt und später an der neuen Position wieder gezeigt werden.
	Setzt die aktuelle Figur auf die vorderste Ebene vor alle anderen Figuren oder auf die hinterste Ebene hinter alle anderen Figuren.
	Verschiebt die aktuelle Figur um eine angegebene Anzahl von Ebenen weiter nach vorne oder hinten.
	Zeigt die aktuell verwendete Kostümnummer oder den Namen des Kostüms der Figur an. Mit diesem Wert kann wie mit einer Variablen gerechnet werden.
	Zeigt die aktuell verwendete Bühnenbildnummer oder den Bühnenbildnamen an. Mit diesem Wert kann wie mit einer Variablen gerechnet werden.
	Zeigt die aktuell dargestellte Größe der Figur an. Mit diesem Wert kann wie mit einer Variablen gerechnet werden.

KLANG

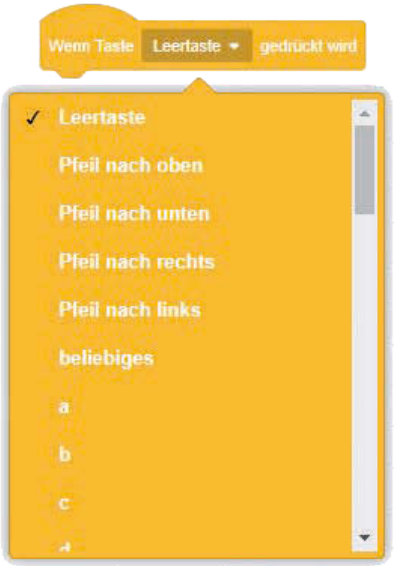
Die Blöcke auf der Blockpalette ***Klang*** lassen verschiedenartige Klänge erklingen.

spiele Klang Miau ganz	Spielt einen Klang ab. Das Skript wartet, bis der Klang zu Ende gespielt wurde. Ein Klang ***Miau*** wird mitgeliefert, weitere Klänge können über ein Mikrofon aufgenommen werden.
spiele Klang Miau	Spielt einen Klang ab. Das Skript läuft sofort weiter und wartet nicht, bis der Klang zu Ende gespielt wurde.
stoppe alle Klänge	Stoppt alle Klänge, die gerade abgespielt werden.
ändere Lautstärke um -10	Ändert die Höhe oder die Links/Rechts-Aussteuerung des Klangs um einen bestimmten Wert.
ändere Effekt Höhe um 10 ✓ Höhe Aussteuern links/rechts	Setzt die Höhe oder die Links/Rechts-Aussteuerung des Klangs auf einen bestimmten Wert.
schalte Klangeffekte aus	Schaltet alle veränderten Klangeffekte wieder aus.
ändere Lautstärke um -10	Ändert die Lautstärke um einen bestimmten Wert.
setze Lautstärke auf 100 %	Setzt die Lautstärke auf einen bestimmten Prozentwert gegenüber dem Original, unabhängig davon, ob die Lautstärke bereits vorher verändert wurde.
☐ Lautstärke	Zeigt die aktuelle Lautstärke des Lautsprechers an. Mit diesem Wert kann wie mit einer Variablen gerechnet werden.

EREIGNISSE

Die Blöcke auf der Blockpalette ***Ereignisse*** lassen die aktuelle Figur auf bestimmte Ereignisse wie Klicks, Tastendrücke oder Ähnliches reagieren. Hier sind auch die Blöcke, die Nachrichten an andere Figuren versenden oder von ihnen empfangen. In Scratch 1.4 sind die Blöcke der Blockpalette ***Ereignisse*** auf der Blockpalette ***Steuerung*** zu finden.

Startet das angehängte Skript für die aktuelle Figur, wenn man auf das grüne Fähnchen klickt.



Startet das angehängte Skript für die aktuelle Figur, wenn eine bestimmte Taste gedrückt wird. Dabei stehen die Leertaste, die Pfeiltasten sowie alle Buchstaben und Ziffern oder einfach eine beliebige Taste zur Auswahl.

Startet das angehängte Skript für die aktuelle Figur, wenn man auf diese Figur klickt.

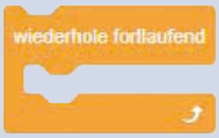
Startet das angehängte Skript für die aktuelle Figur, wenn das Bühnenbild zu einem ausgewählten Bühnenbild wechselt. Auf der Blockpalette ***Aussehen*** gibt es einen Block, der das Bühnenbild wechselt.

Startet das angehängte Skript für die aktuelle Figur, wenn eines dieser Ereignisse eintritt: ***Lautstärke*** oder ***Stoppuhr*** überschreitet einen angegebenen Wert.

	Startet das angehängte Skript für die aktuelle Figur, wenn eine bestimmte Nachricht empfangen wurde. Diese Nachrichten können von derselben oder einer anderen Figur mit einem ***sende ... an alle***-Block gesendet werden. Auf diese Weise können verschiedene Figuren miteinander kommunizieren.
	Sendet eine Nachricht an alle Figuren und auch an die Bühne. Diese Nachrichten können über ***Wenn ich ... empfange*** weitere Skripte starten.
	Sendet eine Nachricht an alle Figuren und auch an die Bühne. Danach wartet das Skript, bis alle Skripte abgearbeitet sind, die durch die Nachricht gestartet wurden.

STEUERUNG

Die Blöcke auf der Blockpalette ***Steuerung*** steuern den Programmablauf für die aktuelle Figur. Hier sind Schleifen, Wartezeiten und Abfragen zu finden.

	Das Skript wartet die angegebene Zeit und läuft dann weiter. Andere Skripte werden nicht angehalten.
	Der Inhalt dieses Blocks wird so oft ausgeführt, wie im Zahlenfeld angegeben.
	Der Inhalt dieses Blocks wird endlos wiederholt, bis das Skript durch einen Klick auf das rote Stoppsymbol oder einen Block ***stoppe ...*** angehalten wird.
	Der Inhalt dieses Blocks wird ausgeführt, wenn die angegebene Bedingung wahr ist. Als Bedingung kann ein Block mit spitzen Enden aus den Paletten ***Fühlen*** oder ***Operatoren*** verwendet werden.

	Der obere Inhalt dieses Blocks wird ausgeführt, wenn die angegebene Bedingung wahr ist, andernfalls wird der untere Inhalt ausgeführt. Als Bedingung kann ein Block mit spitzen Enden aus den Paletten ***Fühlen*** oder ***Operatoren*** verwendet werden.
	Das Skript wartet, bis die angegebene Bedingung wahr ist, und läuft dann weiter. Andere Skripte werden nicht angehalten. Als Bedingung kann ein Block mit spitzen Enden aus den Paletten ***Fühlen*** oder ***Operatoren*** verwendet werden.
	Der Inhalt dieses Blocks wird wiederholt, bis die angegebene Bedingung wahr ist, und läuft dann weiter. Als Bedingung kann ein Block mit spitzen Enden aus den Paletten ***Fühlen*** oder ***Operatoren*** verwendet werden.
	Je nach Auswahl werden alle Skripte, nur dieses Skript oder alle Skripte der aktuellen Figur gestoppt.
	Dieses Skript startet, wenn eine Figur geklont wird.
	Erzeugt einen Klon von der aktuellen oder einer anderen Figur. Dieser Klon ist eine temporäre Kopie und kann jederzeit wieder gelöscht werden. Der Klon erscheint an der Position der geklonten Figur, ist also als solcher nicht zu erkennen, bis er bewegt wird.
	Löscht den Klon, für den das Skript gilt. Wird das Programm mit einem Klick auf das rote Stoppsymbol gestoppt, werden alle Klone gelöscht.

FÜHLEN

Die Blöcke auf der Blockpalette ***Fühlen*** ermöglichen es der aktuellen Figur, auf Berührungen anderer Figuren oder Farben sowie auf Mausaktionen oder Tastendrücke zu reagieren. Im Gegensatz zu den Blöcken auf der Blockpalette ***Ereignisse*** werden diese Blöcke in Abfragen vorhandener Programmblöcke eingebaut. Außerdem sind hier die Blöcke zur Steuerung und Anzeige von Video, Stoppuhr, Datum und Uhrzeit auf der Blockpalette ***Fühlen*** zu finden.

Diese Abfrage liefert wahr, wenn die aktuelle Figur den Mauszeiger, den Rand der Bühne oder eine ausgewählte andere Figur berührt.

Diese Abfrage liefert wahr, wenn die aktuelle Figur die ausgewählte Farbe berührt. Dabei kann es sich um eine farbige Fläche auf dem Bühnenbild, eine vom Malstift gezeichnete Linie oder um eine farbige Fläche auf einer anderen Figur handeln.

Diese Abfrage liefert wahr, wenn eine Fläche in einer bestimmten Farbe auf der aktuellen Figur die ausgewählte Farbe berührt. Dabei kann es sich um eine farbige Fläche auf dem Bühnenbild, eine vom Malstift gezeichnete Linie oder um eine farbige Fläche auf einer anderen Figur handeln.

Gibt die Entfernung der aktuellen Figur vom Mauszeiger oder von einer ausgewählten anderen Figur an.

Die aktuelle Figur zeigt in einer Sprechblase eine Frage an. Die Antwort muss in das Texteingabefeld getippt werden. Das Skript wartet, bis der Benutzer die [Enter]-Taste drückt oder auf das Häkchen klickt. Die Antwort wird in der Variablen ***Antwort*** gespeichert.

Diese Variable speichert die eingegebene Antwort aus dem Block ***frage ... und warte***.

Diese Abfrage liefert wahr, wenn eine ausgewählte Taste gedrückt wurde. Dabei stehen die Leertaste, die Pfeiltasten sowie alle Buchstaben und Ziffern oder einfach eine beliebige Taste zur Auswahl.

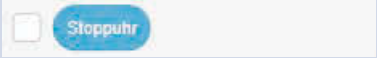
Diese Abfrage liefert wahr, wenn die Maustaste gedrückt wurde.

Zeigt die aktuelle x-Position des Mauspfeils auf der Bühne an. Der Wert kann zwischen -240 und 240 liegen. Mit diesem Wert kann wie mit einer Variablen gerechnet werden.

Zeigt die aktuelle y-Position des Mauspfeils auf der Bühne an. Der Wert kann zwischen -180 und 180 liegen. Mit diesem Wert kann wie mit einer Variablen gerechnet werden.

Setzt die Ziehbarkeit einer Figur. Ziehbare Figuren können, während das Programm läuft, interaktiv auf der Bühne verschoben werden.

Zeigt die aktuelle Lautstärke des Mikrofons an. Mit diesem Wert kann wie mit einer Variablen gerechnet werden. Im Gegensatz dazu zeigt der Block ***Lautstärke*** auf der Blockpalette ***Klang*** die Lautstärke des Lautsprechers an.

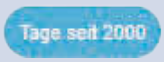


Zeigt die aktuelle Zeit der Stoppuhr in Sekunden an. Mit diesem Wert kann wie mit einer Variablen gerechnet werden. Die Stoppuhr wird mit dem Block ***setze Stoppuhr zurück*** bei 0 gestartet.

Setzt die Stoppuhr auf 0 zurück und startet sie.

Liefert verschiedene Parameter der Bühne. Mit diesen Werten kann wie mit einer Variablen gerechnet werden.

Liefert verschiedene Werte der aktuellen Uhrzeit. Mit diesen Werten kann wie mit einer Variablen gerechnet werden.



Liefert die Anzahl von Tagen seit Beginn des Jahres 2000. Dieser Wert erleichtert die Berechnung von Zeiträumen, da nicht mit Monaten und Jahren gerechnet werden muss.

Liefert den aktuellen Scratch-Benutzernamen des Benutzers, der das Programm gerade online ansieht.

OPERATOREN

Die Blöcke auf der Blockpalette ***Operatoren*** ermöglichen Berechnungen und Vergleiche in Scratch. Hier stehen alle wichtigen mathematischen und logischen Operatoren zur Verfügung. Alle Blöcke mit runden Enden liefern einen Zahlenwert zurück, Blöcke mit spitzen Enden ein logisches ***Wahr*** oder ***Falsch***.

Addiert zwei Zahlenwerte. Alle Operatoren, die Zahlen verarbeiten, können auch mit Variablen rechnen, die Zahlen enthalten.

Subtrahiert zwei Zahlenwerte.

	Multipliziert zwei Zahlenwerte.
	Dividiert zwei Zahlenwerte.
	Liefert eine ganzzahlige Zufallszahl im angegebenen Bereich.
	Liefert wahr, wenn der linke Zahlenwert größer ist als der rechte.
	Liefert wahr, wenn der linke Zahlenwert kleiner ist als der rechte.
	Liefert wahr, wenn beide Zahlenwerte gleich groß sind.
	Liefert wahr, wenn beide Abfragen wahr liefern.
	Liefert wahr, wenn mindestens eine der beiden Abfragen wahr liefert.
	Liefert wahr, wenn die Abfrage falsch liefert, und umgekehrt.
	Verbindet zwei Zeichenketten zu einer.
	Liefert das Zeichen an der angegebenen Position in der Zeichenkette.
	Liefert die Anzahl der Zeichen in der Zeichenkette.
	Liefert wahr, wenn die Zeichenkette das angegebene Zeichen enthält.
	Liefert den ganzzahligen unteilbaren Rest der Division zweier Zahlen.
	Liefert den mathematisch gerundeten Ganzzahlwert der Zahl.

Dieser Block bietet verschiedene mathematische Funktionen zur Auswahl. Trigonometrische Funktionen werden in Grad berechnet.

DATEN

Auf der Blockpalette ***Daten*** lassen sich Variablen und Listen anlegen. Wenn die erste Variable angelegt ist, werden Blöcke angeboten, um die Werte der Variablen zu verändern.

Neue Variable	Legt eine neue Variable an. Sie kann lokal nur für die aktuelle Figur oder global für alle Figuren gelten.
meine Variable	Liefert den Wert der Variablen. Ist das Kontrollkäschen eingeschaltet, wird die Variable auf der Bühne angezeigt. Mit einem Rechtsklick auf diesen Block kann die Variable umbenannt oder gelöscht werden.
setze meine Variable auf 0	Setzt den Wert einer Variablen auf einen beliebigen Wert. Im Listenfeld kann die Variable ausgewählt werden.
ändere meine Variable um 1	Ändert den Wert einer Variablen um einen bestimmten Wert.
zeige Variable meine Variable	Zeigt die Variable auf der Bühne.
verstecke Variable meine Variable	Versteckt die Variable auf der Bühne.

Neue Liste	Legt eine neue Liste an. Sie kann lokal nur für die aktuelle Figur oder global für alle Figuren gelten. Eine Liste kann beliebig viele Werte enthalten. Die Anzahl braucht vorher nicht festgelegt zu werden.
☐ meine Liste	Liefert aneinandergehängt alle Werte der Liste getrennt durch Leerzeichen. Ist das Kontrollkäschen eingeschaltet, werden die Elemente der Liste und ihre Anzahl auf der Bühne angezeigt. Mit einem Rechtsklick auf diesen Block kann die Variable gelöscht werden.
füge Ding zu meine Liste hinzu	Fügt ein Element am Ende der Liste hinzu. Die Liste wird damit um ein Element länger.
lösche 1 aus meine Liste	Löscht das angegebene Element aus der Liste. Die Liste wird damit um ein Element kürzer.
lösche alles aus meine Liste	Löscht alle Elemente aus der Liste. Die Liste ist danach leer.
füge Ding bei 1 in meine Liste ein	Fügt ein Element an einer bestimmten Position der Liste hinzu. Die Liste wird damit um ein Element länger.
ersetze Element 1 von meine Liste durch Ding	Ersetzt ein Element der Liste durch den angegebenen Zahlenwert oder die angegebene Zeichenkette.
Element 1 von meine Liste	Liefert das angegebene Element aus der Liste.
Nummer von Ding in meine Liste	Liefert die Nummer des angegebenen Elements in der Liste.
Länge von meine Liste	Liefert die Länge der Liste.
meine Liste enthält Ding ?	Liefert wahr, wenn der angegebene Zahlenwert oder die angegebene Zeichenkette in der Liste enthalten ist.
zeige Liste meine Liste	Zeigt die Liste auf der Bühne.
verstecke Liste meine Liste	Versteckt die Liste auf der Bühne.

WEITERE BLÖCKE

Auf der Blockpalette ***Weitere Blöcke*** kann man eigene Blöcke mit selbst definierten Funktionen anlegen.

Neuer Block	Legt einen neuen selbst definierten Block an.

REFERENZ: DIE BLÖCKE DER WICHTIGSTEN SCRATCH-ADD-ONS IM ÜBERBLICK

Die folgenden Tabellen zeigen die Blöcke der wichtigsten mitgelieferten Erweiterungen, die in diesem Buch erwähnt sind.

MALSTIFT

Die Blöcke der Erweiterung ***Malstift*** steuern den Malstift, mit dem die aktuelle Figur auf der Bühne Linien zeichnen kann.

lösche alles	Löscht alle Stiftspuren und Abdrücke. Danach ist das aktuelle Bühnenbild wieder zu sehen.
hinterlasse Abdruck	Die aktuelle Figur hinterlässt einen Abdruck auf der Bühne. Dieser Abdruck ist nur ein starres Bild und kann keine Programmblöcke ausführen.
schalte Stift ein	Schaltet den Stift ein. Die Figur hinterlässt dann bei jeder Bewegung eine Spur. Der Stift gilt nur für die aktuelle Figur und kann für jede Figur einzeln ein- und ausgeschaltet werden.
schalte Stift aus	Schaltet den Stift aus. Die Figur hinterlässt dann keine Spuren mehr.
setze Stiftfarbe auf	Setzt die Stiftfarbe der aktuellen Figur auf eine frei wählbare Farbe. Die Farbe kann mit den bekannten Reglern eingestellt oder mit der Pipette irgendwo auf der Bühne gewählt werden.

Ändert Stiftfarbe, Sättigung, Helligkeit oder Transparenz um einen bestimmten Wert.

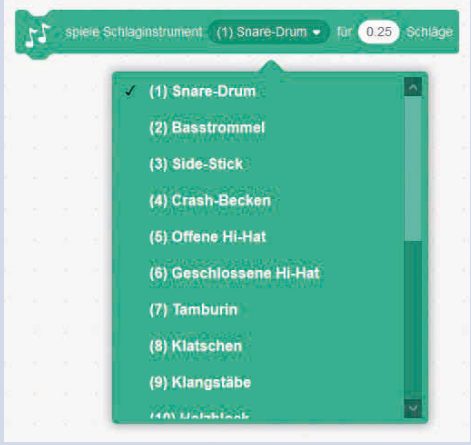
Setzt die Stiftfarbe der aktuellen Figur auf eine bestimmte Farbnummer. 0 steht für Rot, 30 für Grün, 60 für Blau, und 100 steht nach einer Umdrehung des Farbkreises wieder für Rot. Diese Farbnummern werden auf den Reglern in den Farbauswahlfeldern angezeigt. Mit dem gleichen Block lassen sich auch Sättigung, Helligkeit und Transparenz auf einen bestimmten Wert setzen.

Ändert die Stiftdicke der aktuellen Figur um einen bestimmten Wert.

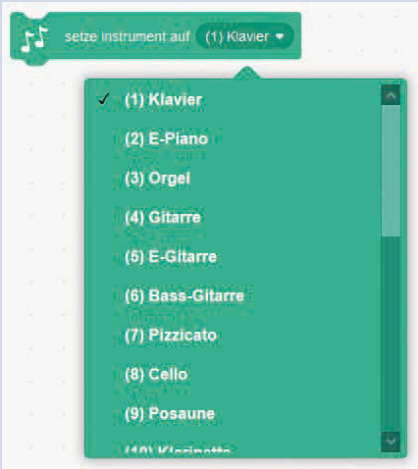
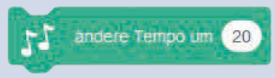
Setzt die Stiftdicke der aktuellen Figur auf einen bestimmten Wert, gemessen in Koordinateneinheiten.

MUSIK

Die Blöcke der Erweiterung ***Musik*** erzeugen verschiedene Töne in Scratch.

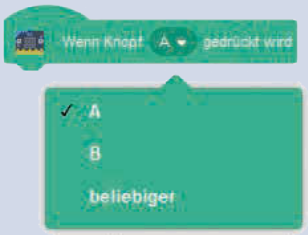


Spielt eine bestimmte Anzahl von Schlägen mit einem Schlaginstrument. Die Art des Schlaginstruments kann im linken Listenfeld ausgewählt werden. Die Dauer eines Schlags wird über den Block ***setze Tempo auf ...*** festgelegt.

	Das Programm legt eine Pause ein, die in Schlägen gemessen wird. Der ähnliche Block ***Warte*** aus der Blockpalette ***Steuerung*** legt eine Pause ein, die in Sekunden gemessen wird.
	Spielt einen auswählbaren Ton ab. Dieser Ton kann als Zahl eingegeben oder über eine Klaviatur ausgewählt werden. Die Abspieldauer wird in Schlägen angegeben.
	Wählt ein Instrument aus einer Liste aus, mit dem die Töne abgespielt werden. Diese Auswahl gilt nicht für das Schlaginstrument.
	Setzt das Tempo auf eine bestimmte Anzahl von Schlägen pro Minute, unabhängig davon, ob das Tempo bereits vorher verändert wurde.
	Ändert das Tempo um einen bestimmten Wert.
	Zeigt das aktuelle Tempo an. Mit diesem Wert kann wie mit einer Variablen gerechnet werden.

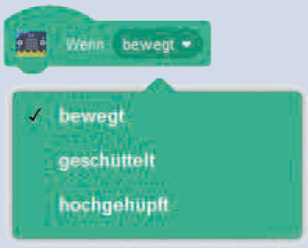
MICRO:BIT

Die Blöcke der Erweiterung ***micro:bit*** nutzen einen micro:Bit als Ein- und Ausgabegerät für Scratch-Programme. Das Programm läuft auf dem PC und kommuniziert über Bluetooth mit dem verbundenen micro:bit.

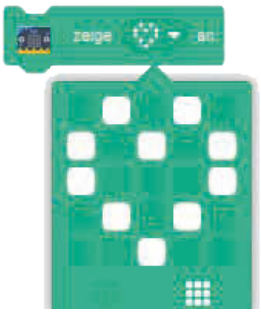


Startet das angehängte Skript, wenn die ausgewählte Taste gedrückt wurde.

Liefert wahr, wenn die ausgewählte Taste gedrückt wurde.



Startet das angehängte Skript, wenn der micro:bit bewegt, geschüttelt oder mit ihm hochgehüpft wird.



Stellt eine beliebige Grafik auf der LED-Matrix dar. Die einzuschaltenden LEDs können im Block festgelegt werden.

Stellt einen Text als Laufschrift auf der LED-Matrix dar.

Schaltet alle LEDs aus.

Startet das angehängte Skript, wenn der micro:bit in die angegebene Richtung geneigt wird.

Liefert wahr, wenn der micro:bit in die angegebene Richtung geneigt wird.

Neigungswinkel in die vier Richtungen.

Startet das angehängte Skript, wenn einer der drei Pins der unteren Kontaktleiste an Masse angeschlossen ist. Mit diesen Pins kann man berührungsempfindliche Sensorkontakte aus elektrisch leitfähigen Gegenständen selbst bauen.

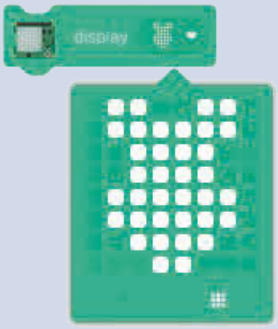
RASPBERRY PI SIMPLE ELECTRONICS

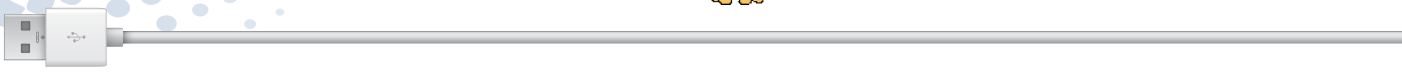
Die Blöcke der Erweiterung ***Raspberry Pi Simple Electronics*** steuern LEDs und Taster an den GPIO-Pins des Raspberry Pi. Auf dem PC wird diese Erweiterung nicht angeboten.

	Startet das angehängte Skript, wenn ein Taster gedrückt oder losgelassen wird.
	Liefert wahr, wenn ein Taster gedrückt oder losgelassen wird.
	Schaltet eine LED ein oder aus.
	Schaltet eine LED um. War sie eingeschaltet, wird sie ausgeschaltet, und umgekehrt.

RASPBERRY PI SENSEHAT

Die Blöcke der Erweiterung ***Raspberry Pi SenseHAT*** stellen Texte oder Grafik auf der LED-Matrix des SenseHAT dar und fragen die Sensoren ab. Auf dem PC wird diese Erweiterung nicht angeboten.

	Stellt einen Text als Laufschrift auf der LED-Matrix dar.
	Stellt ein einzelnes Zeichen auf der LED-Matrix dar. Das nächste Zeichen löscht das vorher dargestellte.
	Stellt eine beliebige Grafik auf der LED-Matrix dar. Die einzuschaltenden LEDs können im Block festgelegt werden.

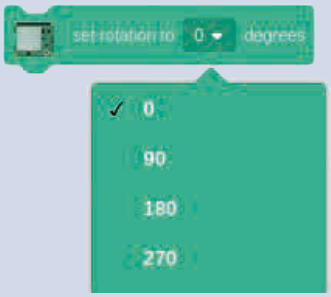


Schaltet alle LEDs aus.

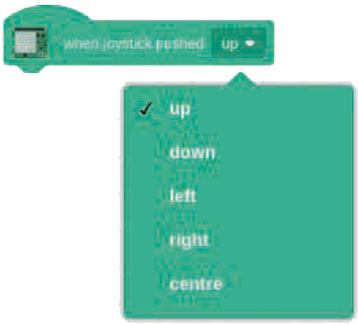
Setzt die Farbe aller LEDs für Texte oder Grafikzeichen auf eine frei einstellbare Farbe.

Setzt die Hintergrundfarbe der LED-Matrix für Texte oder Grafikzeichen auf eine frei einstellbare Farbe. Das betrifft die LEDs, die sonst ausgeschaltet wären.

Setzt die Farbe einer einzelnen LED auf eine frei einstellbare Farbe.

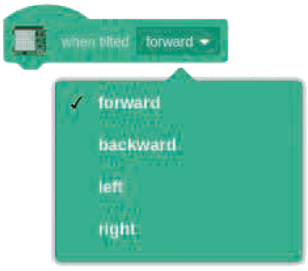
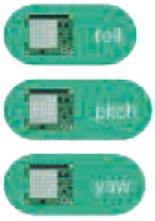


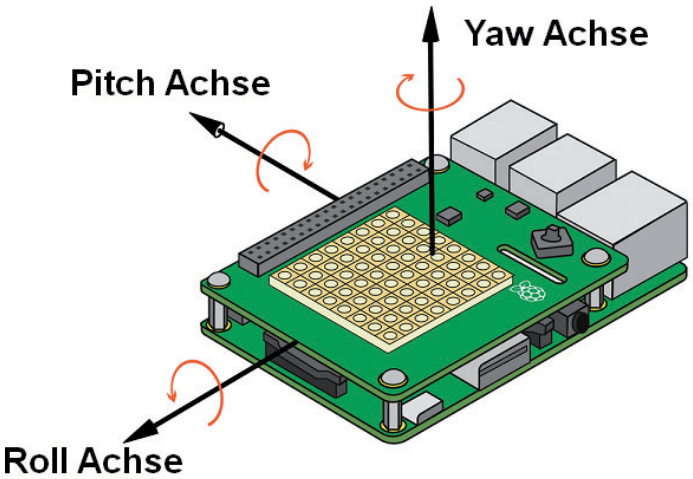
Dreht die Darstellung der gesamten LED-Matrix in Schritten von 90 Grad. Dies ist wichtig, wenn der Raspberry Pi fest eingebaut ist und nicht gedreht werden kann. Durch eine entsprechende Drehung lassen sich Schriften und andere Zeichen dann wieder richtig herum darstellen.



Startet das angehängte Skript, wenn der Joystick in die eingestellte Richtung gedrückt wurde.

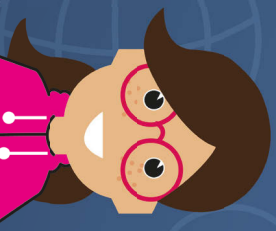
Liefert wahr, wenn der Joystick in die eingestellte Richtung gedrückt wurde.

	Startet das angehängte Skript, wenn der Raspberry Pi mit dem SenseHAT geschüttelt wird.
	Startet das angehängte Skript, wenn der Raspberry Pi mit dem SenseHAT in die angegebene Richtung geneigt wird.
	Temperatur auf dem SenseHAT.
	Luftdruck auf dem SenseHAT.
	Luftfeuchtigkeit auf dem SenseHAT.
	Drehung und Neigung in den drei Achsen (siehe Grafik).



(Grafik: Raspberry Pi Foundation – Creative-Commons-Lizenz)

Programmieren lernen mit Scratch



Richtig programmieren zu lernen mag für viele eine staubtrockene Angelegenheit sein, aber nicht mit Scratch! Was Scratch von anderen Programmiersprachen unterscheidet, ist der rein visuelle Ansatz. Das bedeutet, dass du keine ellenlangen Quelltexte mit komplizierter Syntax in deinen Computer tippen musst. Bei Scratch arbeitest du mit kleinen Bildbausteinen, die aneinandergereiht und ineinander verschachtelt werden. So lernst du schon nach kurzer Zeit, wie ein echter Informatiker zu denken.

Damit der Einstieg gelingt, ist von Anfang an alles da. So z. B. die bekannte Scratch-Katze, deren Farbe du verändern und die du unterschiedlichste Befehle ausführen lassen kannst.

Schnell geht es ans Eingemachte: Zeichne einfache Vektorobjekte, entwirf einen Spielewürfel, baue ein schon fast fertiges Raumschiff nach deinen Ideen weiter oder nutze Scratch für spannende Raspberry-Pi-Projekte.

Schon nach kurzer Zeit schreibst du Programme, mit denen du sogar echte Elektronik ansteuern kannst. Lass LEDs leuchten oder baue ein Keyboard, das wirklich Musik macht. Am Ende des Buches erfährst du, wie du Mitglied in der großen Scratch-Community wirst, deine Programme veröffentlichst und mit anderen Scratchern chattest. Und auch wenn es wirklich schwierig wird, steht dir der Autor Christian Immler mit Tipps und Ratschlägen zur Seite.

In diesem Buch geht es um:

- Scratch im Web-Browser nutzen
- Scratch auf dem Desktop offline einsetzen
- Scratch auf dem Raspberry Pi
- Ein neues Scratch-Projekt anlegen
- Das erste Pong-Spiel programmieren
- Figuren miteinander spielen lassen
- Programmtest in der Entwicklungsphase
- Einfache Vektorobjekte zeichnen
- Einen Spielewürfel programmieren
- Fertiges Raumschiff modifizieren
- Bewegungen im Koordinatensystem
- Steuerung für einen Käfer-Roboter
- Scratch-Erweiterungen installieren
- Micro:bit und SenseHAT einsetzen
- Eine Turtle-Grafik erstellen
- Musik machen mit Scratch
- Kultspiel Flappy Bird nachbauen
- Faszinierende Labyrinthe und Irrgärten
- Analoguhr mit Sekundenzeiger
- Einfache Spiele mit Farben
- und vieles mehr!

FRANZIS

Besuchen Sie unsere Webseite:
www.franzis.de