bombini verlag

Erik Bartmann

Mit Scratch 3 programmieren lernen

Systematisch programmieren lernen
Mit Scratch die ganze Coding-Welt entdecken
Für Schule und Sebststudium geeignet

Mit Scratch 3 programmieren lernen

Erik Bartmann

Kommentare und Fragen können Sie gerne an uns richten:

Bombini Verlags GmbH
Kaiserstraße 235
53113 Bonn
E-Mail: service@bombini-verlag.de

Bibliografische Information Der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.d-nb.de abrufbar.

Umschlaggestaltung: Michael Oreal, Köln (www.oreal.de)
Satz: III-satz, Husby (www.drei-satz.de)

ISBN 978-3-946496-15-1

Inhalt

Eine kurze Geschichte

Meine Freude über die Veröffentlichung von Scratch 3 ist riesig. Ich finde, Scratch ist eine der bedeutsamsten Entwicklungen der letzten 15 Jahre in der digitalen Welt. Scratch ist weltweit DIE Programmiersprache für Einsteiger und für Neugierige geworden. Es ist ein unverzichtbarer Bestandteil der digitalen Bildungswelt geworden, Scratch ist aus Schulen nicht mehr wegzudenken!

»*Sie wollen programmieren lernen und suchen einen geeigneten Einstieg? Wir sagen Ihnen, welche Sprachen bzw. Dialekte aktuell und sinnvoll sind und wie Sie schnell ans Ziel kommen.*«

Diese und ähnliche Versprechungen finden sich zuhauf im Internet. Sie sollen dem Einsteiger einen Überblick über die schier unglaubliche Vielfalt der Programmiersprachen für die unterschiedlichsten Zwecke liefern. Egal wofür, sei es ein Personal Computer, ein Tablet, ein Smartphone oder Embedded Chips, die in Mikrocontrollern zu finden sind. Alle benötigen sie Software, um zu funktionieren. Die Hardware allein stellt dafür lediglich die Grundlage zur Verfügung. Sie ist ohne entsprechende Software nutzlos und bedarf eines mehr oder weniger intelligenten Konzepts, um zu laufen. Hier kommt der Programmierer mit seinen Ideen ins Spiel. Er muss sich natürlich mit der Hardware auskennen und wissen, wie sie programmiert wird.

Gibt es überhaupt eine »richtige Programmiersprache«? Was ist richtig und was ist falsch oder vielleicht nur suboptimal? Für viele gestandene Programmierer stellt eine derartige Frage fast einen Angriff auf ihre Persönlichkeit dar, denn es ist doch wohl klar, dass nur C oder C++ mit Compiler die richtige Wahl ist, denn eine hardwarenahe Programmierung ist nur damit möglich. Oder es sollte eine Skriptsprache wie Python oder Perl sein, die mit einem Interpreter arbeiten und sehr schnell zu installieren sind. Über einen beliebigen Editor sind die Programme – oder besser gesagt Skripten – einzugeben und dann auszuführen.

Das führt uns zum nächsten Punkt: der »richtigen« Entwicklungsumgebung. Das ist das Werkzeug, mit dem die jeweilige Programmiersprache zu nutzen ist. Die Wahl der IDE (Integrated Development Environment) zur Aufnahme des Quellcodes stellt einen zweiten wichtigen Aspekt bei der Programmierung dar. Für einen Einsteiger kann z. B. das Programmieren mit C++ und der Entwicklungsumgebung Visual Studio von Microsoft oder Eclipse

eine enorme Hürde darstellen, denn die Lernkurve ist recht steil. Man muss sich Gedanken über Klassen und Objekte machen und das Programmierparadigma der OOP (objektorientierten Programmierung) ist nicht für jeden sofort verständlich. Dagegen kann zum Beispiel in einem Python-Skript – auch hier gibt es komfortable Entwicklungsumgebungen – sofort drauf los getippt werden und das Ergebnis ist unmittelbar sicht- bzw. erkennbar. Viele Dinge, die mit C++ von Hand erledigt werden müssen, sind in Python über entsprechende Bibliotheken schon fest eingebaut. Das soll aber nicht bedeuten, dass es in C oder C++ keine Bibliotheken zur Vereinfachung der Programmierung gibt. Moderne Sprachen haben sich dem Schmerz der Programmierer aus den Anfängen angenommen und vieles einfacher gestaltet, so dass das Rad nicht immer wieder neu erfinden werden muss. Standardbibliotheken bieten heutzutage eine Fülle an Funktionalitäten, die das Leben enorm erleichtern, sei es beim Zugriff auf Dateisysteme, der Kommunikation über Netzwerke oder der drahtlosen Verbindungsaufnahme via Wifi oder Bluetooth, um nur einige wenige Beispiele zu nennen. Entsprechende Interfaces innerhalb der Programmierung machen derartige Aktivitäten mehr oder weniger zu einem Kinderspiel.

Bei der Nutzung herkömmlicher Programmiersprachen ist das Eingeben von Quellcode – das sogenannte *Coding* oder auch *Coden* – in eine Entwicklungsumgebung unumgänglich. Die Sprachsyntax, also das Regelsystem zur Kombination elementarer Zeichen bzw. Wörter, ist bei jeder Programmiersprache anders, obwohl es dort auch Überschneidungen und Gemeinsamkeiten geben kann, wie man es an C++ und Java sieht. Viele sogenannte Schlüsselwörter sind in der betreffenden Programmiersprache zu erlernen und das ist nicht immer so einfach wie erhofft, da diese Schlüsselwörter meistens in englischer Sprache sind und recht kryptisch anmuten können.

Es gibt jedoch einen anderen Ansatz, der sich mehr auf das visuelle Arbeiten konzentriert, wobei ich nicht unterstellen möchte, dass alle anderen Programmierer blind arbeiten. Die Programmiersprache, die ich jetzt ansprechen möchte, nutzt das Programmieren in visuellen Blöcken. Was könnte das bedeuten? Vielleicht hat der eine oder andere schon einmal ein Flussdiagramm mit verschiedenen Blöcken gesehen, die untereinander angeordnet sind. Jeder einzelne Block hat eine bestimmte Beschreibung und Aufgabe, wobei ein einzelner Block eine oder mehrere Aufgaben erfüllen kann.

In Grund- oder auch weiterführenden Schulen ist Scratch mittlerweile sehr verbreitet und findet großen Anklang. Scratch wurde von der Lifelong Kindergarten Group am MIT Media Lab entwickelt. Im Gegensatz zu den konventionellen Sprachen werden Scratch-Programme durch das Zusammenfügen von Blöcken mit der Maus auf dem Bildschirm erstellt. Jeder einzelne Block besitzt eine bestimmte Funktionalität, kann aus einem Vorrat von vorhandenen Blöcken entnommen und wie ein Puzzle zusammengefügt werden. Dabei verhält es sich wie bei einem echten Puzzlespiel. Es passen nur die Teile zusammen, die auch einen Sinn ergeben und logisch zueinanderpassen. Der Einsteiger lernt dadurch einerseits die Struktur der Programme kennen und andererseits das Konzept von Algorithmen.

Was ist ein Algorithmus?

Ein Algorithmus stellt eine Vorgehensweise zur Lösung eines Problems dar. Algorithmen bestehen aus endlich vielen einzelnen Schritten, die in einer Weise abgearbeitet werden, die der Programmierer für richtig erachtet hatte. Dabei gibt es theoretisch unendlich viele Algorithmen zur Lösung eines Problems.

Der Vorteil der Verwendung von Scratch-Blöcken besteht unter anderem darin, sich nicht mit der Syntax der Programmiersprache auseinandersetzen zu müssen. Etwaige Syntaxfehler werden somit ausgeschlossen. Ein Blick auf einen Block und dessen Beschreibung ist sprechend und es erschließt sich sofort – mit einem Blick – dessen Bedeutung bzw. Funktion. Zudem sind die einzelnen Blöcke hinsichtlich ihrer Funktion in verschiedenen Codekategorien zusammengefasst und farblich gekennzeichnet, was das Verständnis über die Aufgabe zusätzlich vereinfacht. Auch hier kommt das Visuelle als unterstützender Faktor wieder zum Tragen. Viele grafische Programmiertools sind intuitiv und einfach zu verstehen. Dieser Vorteil kommt Scratch zugute. Menschen mit keinen oder geringen Computerkenntnissen haben darüber schnell Erfolgserlebnisse und werden motiviert statt desillusioniert. Natürlich stellt sich in diesem Zusammenhang die Frage, welche Zielgruppe Scratch im Visier hat. Das MIT empfiehlt Scratch für Menschen zwischen 8 und 16 Jahren. Dem kann ich zustimmen, möchte aber diese Aussage erweitern, denn auch Erwachsene können einiges lernen und Spaß mit Scratch haben. Scratch besitzt eine sehr große Community und bei Problemen gibt es in der Regel sofort Hilfe, denn die Mitglieder fühlen sich verpflichtet, sich gegenseitig bestmöglich zu unterstützen. Es gibt unzählige kostenlose Unterrichtsmaterialien und Beispiele. Scratch ist also keine Insellösung oder ein kurzzeitiger Hype, sondern eine ernstzunehmende Chance, Menschen mit der IT (Informationstechnologie) in Berührung zu bringen, die nicht nur effektiv ist, sondern auch noch Spaß macht. Wo findet man das heutzutage noch? Nach den Scratch-Versionen 1.4 und 2.0 ist gerade die Version 3.0 erschienen. Wer sich mit den Vorversionen schon auskennt und damit gearbeitet hat, dem möchte ich an dieser Stelle die Neuerungen nennen, die Scratch 3.0 bietet:

- Bessere Handhabung der Blöcke in den unterschiedlichen Kategorien: Sie sind scrollbar.
- Neue Anordnung der Bühne (Stage): Sie ist jetzt auf der rechten Seite.
- Neue Sprites
- Neuer und verbesserter Zeicheneditor
- Neuer und verbesserter Soundeditor
- Neue Programmierblöcke mit intuitiver Handhabung
- Neue Erweiterungsmöglichkeiten in Bezug auf Hardwareansteuerung (z. B. micro:bit und LEGO)
- Übersichtlichere Verwendung von Variablen
- Verbesserte Handhabung der Farbblöcke zur Farbmanipulation (Colorpicker-Blocks)
- Bessere Unterstützung bei der Ausführung von Programmen auf Tablets

Scratch 3 basiert nicht mehr auf Flash, sondern unterstützt den Industriestandard HTML 5. Die folgenden Webbrowser werden dabei unterstützt:

- Desktop (Windows, Linux, Mac OSX)
  - Chrome (63+)
  - Edge (15+)
  - Firefox (57+)
  - Safari (11+)
- Tablet (iOS 11+ und Android 6+)
  - Mobile Chrome (62+)
  - Mobile Safari (11+)

Der Internet Explorer wird nicht unterstützt! WebGL ist eine Browsertechnologie, die von Scratch 3.0 verwendet wird, um Projekte auf die Scratch-Bühne zu bringen. Während WebGL in allen modernen Browsern unterstützt wird, können einige ältere Computer und Betriebssysteme es nicht unterstützen. Für Benutzer, die WebGL nicht ausführen können, wird der Scratch 2.0-Offline-Editor empfohlen. Für Entwickler, die sich mit der Programmierung von Extensions befassen möchten, gibt es hilfreiche Guidelines mit der Vorstellung der Spezifikationen. Gerade wenn es um professionelle Extensions (Erweiterungen) von Scratch 3.0 geht, ist die Firma Makeblock zu nennen. Die Software mBlock in der Version 5 verwendet Scratch 3.0 als Grundlage und erstellt sowohl visuelle als auch textbasierte Programmiersoftware, die darauf abzielt, die beste STEAM-Ausbildung zu liefern. STEAM ist ein pädagogischer Ansatz für das Lernen von Naturwissenschaften, Technik, Ingenieurwesen, Kunst und Mathematik für die Schulung bzw. Ausbildung in Dialogform und der Zuhilfenahme kritischen Denkens. Die schon erwähnten älteren Scratch-Versionen 1.4 und 2.0 werden online nicht mehr verfügbar sein und lediglich als Offlineversionen zur Verfügung stehen. Die unter Scratch 2.0 erstellten Projekte sind weiterhin in der Version 3.0 lauffähig und können dort weiter programmiert werden.

Ich wünsche dir nun viel Spaß und Erfolg beim Entdecken und Ausprobieren der neuen Programmiersprache Scratch 3.0!

KAPITEL 1

Vorhang auf für Scratch 3

Als ich vor vielen Jahren mit dem Programmieren anfing, wurde die meiste Zeit dafür verwendet, den Programmcode einzugeben. In den 80er-Jahren des letzten Jahrhunderts waren Rechner wie zum Beispiel der Apple II sehr beliebt. Diese frühen Computer hatten eins gemeinsam: Die Kommunikation mit dem Rechner und die Programmierung erfolgte durch Texteingabe. Das bedeutet, dass alle Befehle an den Computer Zeichen für Zeichen eingetippt werden mussten. Eine recht aufwändige Angelegenheit und je nach körperlicher oder geistiger Verfassung des Programmierers mit einer recht hohen Fehlerquote behaftet.

Eine damals beliebte Programmiersprache hieß *Basic*, die übrigens auch heute noch von vielen genutzt wird. Die damaligen Betriebssysteme, also die Programme, die den Computer überhaupt erst zum Laufen bringen, waren aus heutiger Sicht ziemlich einfach. Wenn es beispielsweise um die simple Aufgabe ging, Programmcode oder Dateien auf Speichermedien außerhalb des Computers zu speichern, mussten manchmal sehr kryptische Zeichenfolgen eingetippt werden. Das war mühsam und fehleranfällig. Programmierer machten sich deshalb Gedanken, wie man die Eingaben bequemer machen könnte.

Kleine Geschichte der PC-Maus

Ein entscheidender Durchbruch für die bequemere Kommunikation mit dem Rechner kam mit der Erfindung der Maus als Eingabegerät, die in Verbindung mit einer grafischen Oberfläche – auch *GUI* (Graphical User Interface) genannt – die Bewegungen der Hand auf einem Monitor nachzeichnet. Das Konzept der GUIs stammt aus den 1970er-Jahren, als an einem Forschungszentrum in Kalifornien, am *Xerox PARC* (Palo Alto Research Center), eben das Zusammenspiel von Maus und grafischer Benutzeroberfläche auf einem Computer entwickelt wurde. Nach einem Besuch dieser Firma und von dieser Technologie inspiriert hatte Steve Jobs, der damalige Firmenchef von *Apple*, eine Vision für seinen eigenen Computer und ließ ein grafisches Betriebssystem auf Grundlage dessen entwickeln, was er bei Xerox PARC gesehen hatte. Der ehemalige Chef von *Microsoft*, Bill Gates, hatte ebenfalls Wind von dieser bahnbrechenden Entwicklung bekommen und tat das Seine, um das heute vorherrschende Betriebssystem *Windows* aus der Taufe zu heben. Der Rest ist Geschichte.

Warum erzähle ich dir das alte Zeugs in einem Buch, in dem es um die moderne Programmiersprache Scratch geht? Weil es Parallelen zwischen damals und heute gibt: Damals wollte man mithilfe einer grafischen Bedienoberfläche des Computers und der Maus die Bedienung des Computers erleichtern, weg von der Texteingabe hin zu dem Anklicken von Grafikelementen. Über ein grafisch orientiertes Betriebssystem ist die Befehlseingabe einfacher, denn die eigentlichen Befehle, beispielsweise zum Anlegen eines Unterverzeichnisses, verbergen sich hinter kleinen Bildchen, die auch Piktogramme genannt werden. Und genau das geschieht jetzt auch bei Scratch: Du programmierst mit der Maus.

So sieht ein Scratch-Programm aus

Schluss mit dem Gerede, jetzt zeige ich Dir erst einmal, wie ein Scratch-Programm aussieht. Du musst das jetzt noch nicht alles verstehen, ich werde dir in den folgenden Kapiteln jeden einzelnen Schritt genau erklären, den du machen musst, um zu eigenen Scratch-Programmen zu kommen.

Schau dir das folgende Bild an. Zugegeben, es ist kein Rembrandt-Gemälde, aber die Grafik sieht doch schon mal recht ansprechend aus, oder? Mir gefällt diese Grafik, weil sie so schön bunt ist und weil sie so exakt gezeichnet ist. Es ist eine schöne Spiralform:

Abb. 1.1: Scratch zeichnet eine bunte Spirale auf den Bildschirm

Was ist, wenn ich dir verspreche, dass du schon nach wenigen Buchseiten in der Lage sein wirst, diese Grafik zu erzeugen, indem du das in Scratch selber programmierst?

Nun zeige ich dir das Scratch-Programm, das diese Grafik erzeugt hat (Abbildung 1.2).

Versuche doch einmal, nur anhand der ineinander verschachtelten Puzzlestücke herauszufinden, wie das ganze Programm funktioniert. Übrigens: Ein Programm in Scratch wird auch *Skript* genannt, Programm und Skript bezeichnet in Scratch dasselbe. Wenn du dir das Programm von oben nach unten anschaust, wirst du es noch nicht restlos verstehen, es sei denn, du bist ein Genie. Doch wenn du die erzeugte Grafik ansiehst und das mit dem Programm in Beziehung setzt, wirst du bestimmt schon ein paar Ideen bekommen, was im Computer passiert, wenn er das Skript umsetzt.

Ich gebe dir ein bisschen Starthilfe. Wenn du die grüne Flagge innerhalb von Scratch anklickst, dann startet die Ausführung. Das Skript rennt praktisch los und arbeitet die Puzzlesteine von oben nach unten ab. Die Puzzlesteine werden in Scratch *Blöcke* genannt. Der

nächste Block beeinflusst die Bewegung und setzt sie auf den angezeigten Wert. Der nächste Block... doch Stopp! Mehr will ich nicht verraten. Ein bisschen sollen die kleinen grauen Zellen schon in Bewegung kommen. Ganz so wie das Skript.

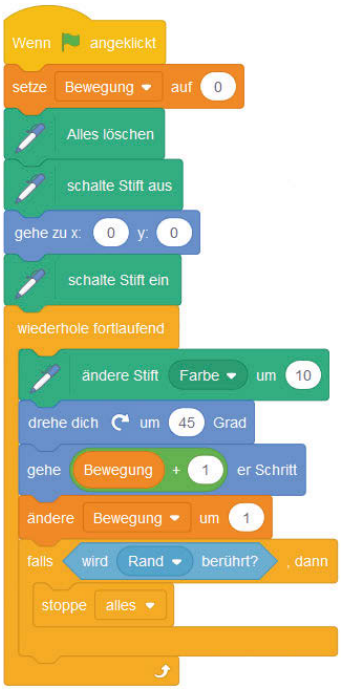


Abb. 1.2: Das Scratch-Programm, das die Spirale erzeugt

Wenn du möchtest, dann probiere doch einfach mal selbst, die Puzzlesteine in Scratch so ineinander zu schachteln, wie es auf der Abbildung zu sehen ist. Keine Angst, du kannst nichts an deinem Computer damit kaputt machen, aber mit etwas Glück bekommst du durch reines Ausprobieren bereits wichtige Informationen, die du später noch gut gebrauchen kannst. Das Skript kannst du von Hand erstellen, doch ich biete es zum Download auf meiner Internetseite `www.erik-bartmann.de` an.

Scratch für Eilige

Zwar kannst du das Programm noch nicht komplett verstehen, dafür fehlen dir noch wichtige Informationen, die du alle in den folgenden Kapiteln erhalten wirst. Aber du erkennst sicherlich, dass die Puzzlestücke, also die Blöcke, unterschiedliche Farben haben und irgendwie ineinander geschachtelt werden können. Vermutlich kann man in den Blöcken auch noch Einstellungen vornehmen. In einem Ausgabefenster von Scratch, das *Bühne* genannt wird, finden sämtliche Aktionen statt, die frei programmiert werden können. Durch das Platzieren von ein oder mehreren Bildern auf der Bühne, die sich *Figuren* nennen, kannst du über das geschickte Zusammenfügen unterschiedlicher Codeblöcke der ganzen Szenerie Leben einhauchen. Es finden *Bewegungen* statt, *Klänge* sind hörbar, *Hintergrundbilder* ver-

ändern sich und Eingaben über die Tastatur gestatten eine Interaktion mit dem Programm. Jeder der vorhandenen Codeblöcke führt eine bestimmte festgelegte Aktion aus. Das Spannende daran liegt in dem geschickten Zusammenbauen mehrerer Codeblöcke zu einer *Sequenz*, die dann nach dem Start des Skriptes in der angegebenen Reihenfolge abgearbeitet wird. Falls es auf der Bühne mehrere Figuren geben sollte, besitzt jede einzelne ihren eignen Codebereich, der individuell – je nach Programmierung – verschieden sein kann.

Blöcke, Blöcke, Blöcke

Ein Block ist das wichtigste Element bei Scratch, so wie Worte die wichtigsten Elemente einer gesprochenen Sprache darstellen. Ich zeige dir jetzt ein paar Codeblöcke. Zuvor noch eine kleine formelle Unterscheidung der verschiedenen Blockarten. Die folgende Tabelle zeigt dir die Umrisse der Blöcke, anhand derer sie zu identifizieren sind und in der rechten Spalte die entsprechenden Erläuterungen.

	*HAT-Block*: Er startet das Programm bzw. Skript.
	*C-Block*: Er wird für Bedingungen und Schleifen verwendet. Er hat seinen Namen von der Form des Buchstaben »C«.
	*Reporter-Block*: Er beinhaltet oder liefert einen Wert. Bei einem Wert kann es sich sowohl um eine Zahl als auch eine Zeichenkette (»Hallo Welt«) handeln.
	*Boolean-Block*: Es handelt sich um einen Wahrheitsblock. Er ist eine spezielle Variante des Reporter-Blocks und liefert einen Wert, der nur wahr oder falsch sein kann. Diese Logik entspricht dem Datentyp Boolean.
	*CAP-Block*: Er stoppt das Programm bzw. Skript.
	*Stack-Block*: Er ist so geformt, dass er über und unter andere Blöcke passt. Diese sogenannten Stapelblöcke machen die Mehrheit der in Scratch verfügbaren Blöcke aus und sind in jeder Palette bzw. Kategorie mit Ausnahme von Operatoren verfügbar.

Tabelle 1.1: Die verschiedenen Blockarten in Scratch

Die verschiedenen Blöcke sind in neun unterschiedlichen Paletten organisiert, auf die ich im nächsten Kapitel zu sprechen komme, wenn du die Scratch-Oberfläche kennenlernst.

Doch nun zu ein paar Beispielen. Der erste Block bewirkt eine Bewegung auf der Bühne. Was eine Bühne im Detail ist, werde ich noch erläutern. Auf einer Bühne wie zum Beispiel in einem Theater wird ein Stück oder eine Show aufgeführt. Auf besagter Bühne bewegen sich also einer oder mehrere Schauspieler, um anhand von Bewegungen und Kommentaren dem Theaterstück eine besondere Note zu geben. Nichts anderes passiert auf der Scratch-Bühne. Über eine dort abgelegte Figur werden durch Aufrufen bzw. Abarbeiten der gerade

schon erwähnten Sequenz verschiedene Codeblöcke ausgeführt und zum Beispiel eine Bewegung der Figur bewirkt. Der folgende Block macht genau dies:

Abb. 1.3: Der Gehe-Block

Dieser *Gehe*-Block führt beim Aufruf des Scratch-Programms bzw. -skriptes eine Bewegung aus und zwar in einzelnen Schritten, wobei jeder einzelne Schritt einer Bewegung mit einer festgelegten Schrittweite entspricht. Ganz ohne manuelle Eingaben über die Tastatur geht es hier leider nicht, denn du kannst die Schrittweite anpassen. Das bedeutet jedoch nicht, dass die eigentliche Programmierung über die Tastatur erfolgt, sondern dass sie lediglich in der Modifikation eines vorgegebenen Parameters besteht.

Der Block hat eine Farbe, die auf die Funktion schließen lässt. Hellblaue Puzzlestücke bedeuten *Bewegung*. Dann besitzt dieser Block – gekennzeichnet durch das weiße Textfeld in der Mitte – eine Möglichkeit, den dort angezeigten Parameter anzupassen. Der Wert 10 legt fest, dass beim Aufruf die Schrittweite eben 10 Schritte beträgt. Durch einen Doppelklick auf den angezeigten Wert kannst du diesen über die Tastatur anpassen.

Auf der folgenden Abbildung sind drei Blöcke zu erkennen, die nacheinander beim Aufruf des Skriptes von oben nach unten abgearbeitet, also ausgeführt werden:

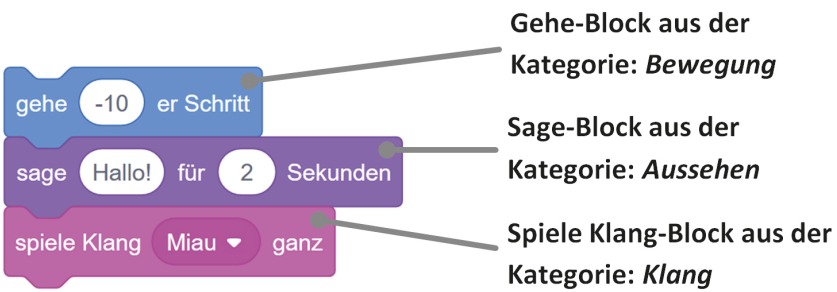


Abb. 1.4: Mehrere Blöcke sind miteinander verbunden

Die Abarbeitung erfolgt also von oben nach unten und erst, wenn ein Block seine Arbeit komplett verrichtet hat, wird die Ausführungskontrolle an den darunter liegenden Block weitergereicht. Der zweite, dunkelblaue Block kommt aus der Kategorie *Aussehen* und der dritte, lila Block aus der Kategorie *Klang*.

An den Blöcken ist oben eine Einbuchtung und unten eine Ausstülpung zu erkennen. Das deutet darauf hin, dass dort weitere Blöcke angesetzt werden können. Es ist an diesem kurzen Beispiel recht gut zu erkennen, wie die einzelnen Blöcke ineinandergreifen. Die nach unten weisende Ausstülpung – oder Nase – des ersten Blocks passt wunderbar in die Ausbuchtung des darunter liegenden Blocks.

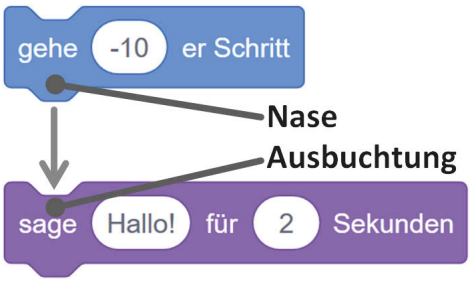


Abb. 1.5: Die beiden Blöcke passen zueinander

Im oberen Gehe-Block können lediglich numerische Werte, positiv oder negativ – je nach gewünschter Bewegungsrichtung – eingegeben werden. Andere Zeichen werden nicht akzeptiert. Positive Werte bewirken eine Bewegung nach rechts, negative eine Bewegung nach links. Der darunter liegende Block stellt zwei Parameter zur Verfügung. Im ersten Feld kann ein beliebiger anzuzeigender Text eingegeben werden und im zweiten Block wieder nur ein numerischer Wert, der die Länge der Anzeige festlegt. Der dritte Block bietet ebenfalls eine Modifikationsmöglichkeit des Parameters an, doch dieser wird nicht als Freitext angeboten, sondern muss aus einer vordefinierten Liste ausgewählt werden.

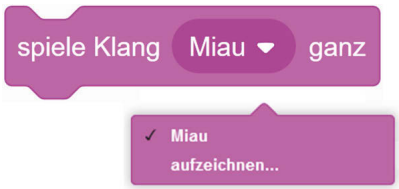


Abb. 1.6: Das Auswahlmenü des Blocks

Durch einen Mausklick auf den nach unten weisenden weißen Pfeil öffnet sich eine Auswahlliste mit den zur Verfügung stehenden Klängen. Im Moment ist der Klang *Miau* ausgewählt. Durch die Wahl des Menüpunktes *Aufzeichnen* gibt Scratch dir die Möglichkeit, über ein Mikrofon einen neuen Klang aufzuzeichnen und abzuspeichern, der dann in Zukunft zur Verfügung steht. Natürlich gäbe es zu den Blöcken noch vieles zu erzählen, doch für diese kurze Einleitung sollte es genügen.

Welche Ängste haben Menschen vor der ersten Begegnung mit einer Programmiersprache?

Ich kann mich noch gut an meine Befürchtungen erinnern, bevor ich das Programmieren gelernt habe. Heute weiß ich, dass sich diese Ängste schnell verflüchtigt haben, nachdem ich einen Einstieg gefunden hatte. Aber es erscheint mir sinnvoll, sich darüber klar zu werden, welche Ängste das sind:

- Das ist meine erste Begegnung mit einer Programmiersprache. Was mag da wohl alles schief gehen?

- Muss ich mir das antun, so viele Befehle zu lernen?
- Ich habe auch Schwierigkeiten in Mathe und das Programmieren ist ja so ähnlich wie Mathe.
- Ich habe Angst, dass ich scheitere und ich die Lust verliere.
- Gibt es keinen einfacheren Weg, einen geeigneten Einstieg zu finden?
- Ist der harte Weg immer der bessere?
- Bin ich der Einzige, der solche Probleme hat oder geht es auch noch anderen so?
- Ist das denn die geeignete Programmiersprache für mich?
- Ist denn Scratch überhaupt zukunftsträchtig oder muss ich später eine weitere Programmiersprache lernen?

Die Ängste sind da und man sollte sie auch ernst nehmen. Aber ich halte es für didaktisch sinnvoll, beim Erlernen von Scratch auf die Neugierde zu setzen, die jeder beim Kennenlernen von etwas Neuem hat. Und bei Scratch gibt es so viel Neues zu entdecken! Da es mit Scratch möglich ist, sehr schnell zu beeindruckend schönen Ergebnissen zu kommen (siehe beispielsweise die Farbspirale in der Abbildung weiter oben), sollte der Fokus beim Lernprozess auf Erfolgserlebnisse gelegt werden. Aus meiner Erfahrung weiß ich, dass programmieren zu lernen für viele eine großartige Möglichkeit darstellt, in einem neuen Wissensgebiet ohne Vorwissen aus anderen Lernbereichen sehr schnell zu tollen, eigenen Ergebnissen zu kommen. Es kann die gesamte Persönlichkeitsentwicklung fördern, wenn Menschen durch das Beherrschen von Computersystemen lernen, ihre Umwelt zu messen, zu kontrollieren und zu verbessern.

Was macht Scratch so interessant und attraktiv? Ich denke, dass jeder für sich persönlich einige Merkmale findet, warum es so faszinierend ist, mit Scratch die Zeit zu verbringen. Ich möchte folgende Aspekte der Programmiersprache hervorheben, die meiner Meinung nach einen entscheidenden Vorteil gegenüber herkömmlichen Sprachen gerade für Einsteiger bietet:

- Scratch ist recht schnell zu erlernen und intuitiv zu bedienen.
- Scratch ist motivationssteigernd.
- Scratch ist kreativitätsfördernd.
- Scratch liefert in kurzer Zeit fantastische Resultate.
- Scratch fördert das selbstständige Arbeiten.
- Scratch ist kompatibel mit anderen Programmiersprachen.
- Scratch benötigt zu Beginn nur einen Browser und Internetzugang. (Es ist auch eine Offlineversion vorhanden.)
- Scratch hat weltweit viele Anhänger und eine sehr aktive Community.

So sieht Code in anderen Programmiersprachen aus

Möchten Programmierer eine Programmiersprache ausprobieren oder anderen vorführen, dann demonstrieren sie die Handlungsweise gern an einem *Hallo-Welt-Programm* (oder auch Hello-World-Programm, wenn man das für Leute macht, die kein Deutsch verstehen). Das Hallo-Welt-Programm hat nur eine kleine Aufgabe: Die Worte »Hello World« sollen am Monitor ausgegeben werden. Mehr nicht.

Sehen wir uns ein paar Hello-World-Beispiele in verschiedenen Programmiersprachen an, die dir möglicherweise sehr kryptisch vorkommen.

Das Hello-World-Programm in der Programmiersprache C++

```
#include <iostream>

main() {
 std::cout << "Hello World" << std::endl;
}
```

Abb. 1.7: Das Hello-World-Programm in C++

Das Hello-World-Programm in der Programmiersprache Java

```
public class HelloWorld {

    /**
     * @param args
     */
    public static void main(String[] args) {
        System.out.println("Hello World!");
    }
}
```

Abb. 1.8: Das Hello-World-Programm in Java

Das Hello-World-Programm in Python

```
def main():
    print('Hello World!')

if __name__ == '__main__':
    main()

```

Abb. 1.9: Das Hello-World-Programm in Python

Wenn du dir diese Vielfalt anschaust – und wir haben es hier lediglich mit drei Varianten zu tun –, einen einfachen Text wie *Hello World!* auszugeben, dann kann das schon etwas verwirrend sein. Wie du siehst, besteht ein Programm aus einer Abfolge von Befehlen, die dem Computer genau sagen, was er tun soll. Jeder dieser einzelnen Befehle muss akribisch formuliert und eingegeben werden und die trivialsten Fehler in der Schreibweise werden nicht verziehen und gnadenlos mit Programmabbruch quittiert. Viele Programmiersprachen unterscheiden zusätzlich zwischen Groß- oder Kleinschreibung, was uns zeigt, dass hier ebenfalls sehr präzise gearbeitet werden muss. Lass dich dadurch nicht verunsichern, denn aller Anfang muss nicht unbedingt schwer sein. Wenn es darum geht, dir Grundkenntnisse der Informationsverarbeitung bzw. der Programmierung anzueignen, dann ist die visuell orientierte Programmiersprache Scratch genau richtig für dich. Für weitere Informationen zu meinem Buch findest du unter der folgenden Internetadresse bestimmt etwas Interessantes für dich. Ich würde mich freuen, wenn du dort vorbeischaust.

`https://erik-bartmann.de/?Downloads___Scratch_3`

So geht's weiter

Scratch wird dir sicherlich viel Spaß bereiten und etwas in dir wecken, wovon du vielleicht noch nicht weißt, dass es in dir steckt. Und dabei ist Scratch nicht nur etwas für Einsteiger und diejenigen, die das erste Mal in Berührung mit einem Computer kommen. Auch Erwachsene haben ihre Freude daran und vielleicht gibt es einen Vater oder eine Mutter, die mit Scratch ihrem Sohn oder ihrer Tochter zeigen möchten, dass beim Programmieren der Faktor Spaß nicht zu kurz kommt. Es geht nicht darum, eine vorgegebene Checkliste abzuarbeiten, nach der du dann beurteilst wirst. Und wenn du bei den Beispielen, die du hier im Buch findest, eigene Ideen hast, ändere alles nach deinen Wünschen ab und gib deiner Kreativität Raum. Es gibt eigentlich nichts Schlimmeres, als wenn du streng nach Vorschrift alles hier Gezeigte abarbeitest und genau das tust, was ich dir vorgebe. Nimm die im Buch gezeigten Grundlagen, Techniken und Beispiele als Ausgangsbasis für weitere persönliche Forschungen und Experimente und dann wirst du sehr schnell erkennen, dass deine Intuition dich dort hinführen wird, wovon andere nicht einmal geträumt haben! Lass dich also – vielleicht auch von dir selbst – überraschen.

Nun habe ich schon so häufig die Scratch-Blöcke zur Sprache gebracht, dass es wirklich an der Zeit ist, uns diese Konstrukte genauer anzusehen. Im folgenden Kapitel erzähle ich dir alles, was ich zu der Scratch-Oberfläche weiß, zu der du gelangst, wenn du über deinen Browser dorthin gehst. Du wirst am Ende des nächsten Kapitels genau wissen, was eine Bühne in Scratch ist, auch was eine Figur und was ein Kostüm ist. Das ist alles nicht schwierig zu verstehen, du musst nur hinsehen und selbst ausprobieren.

KAPITEL 2

Scratch 3 unter der Lupe

Bevor ich mit konkreten Programmierbeispielen für Scratch 3 starte, möchte ich die Scratch-Oberfläche erläutern. Wenn du weißt, wo man alles bei Scratch 3 findet, kannst du auch schnell selbst dein erstes Programm schreiben.

Über die folgende Internetadresse ist Scratch 3 zu erreichen:

`https://scratch.mit.edu/`

Du hast mit Scratch 3 die beiden Möglichkeiten, entweder online direkt innerhalb des Webbrowsers zu arbeiten oder offline in einer zuvor installierten Scratch-Version, die sich *Scratch-Desktop* nennt. Ich beginne mit der Onlineversion und beschreibe am Ende dieses Kapitels die *Installation* von Scratch-Desktop.

Die Scratch-Oberfläche

Nach dem Aufrufen der Internetadresse hast du die Möglichkeit – falls noch nicht geschehen – dich als sogenannter *Scratcher* zu registrieren. Das ist kostenlos und macht dich zu einem aktiven Mitglied der weltweiten Scratch-Community. Klicke dazu auf die Schaltfläche, die ich in Abbildung 2.1 rot umrandet habe und registriere dich mit einem Namen und einem Passwort.

Abb. 2.1: Du willst Scratcher werden

Im Anschluss klickst du noch auf die große orange Schaltfläche

und Scratch 3 präsentiert sich in der Ansicht der Entwicklungsumgebung. Ganz zu Beginn deiner Tätigkeit als Scratcher erscheint innerhalb des Browserfensters noch ein kleines

Tutorial-Video, das dir ein paar Einstiegshilfen bietet. Schau es dir an, für den allerersten Überblick ist es bestimmt brauchbar. Du kannst es bei Bedarf über das in der rechten oberen Ecke befindliche Kreuz schließen.

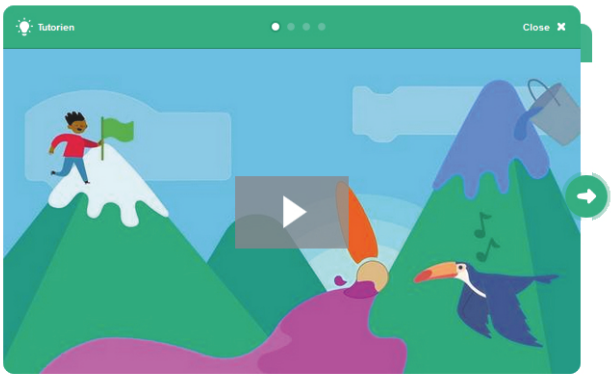


Abb. 2.2: Ein Tutorial wird angeboten

Nun zur eigentlichen Entwicklungsumgebung von Scratch. Neben den einzelnen Bereichen, die ich rot umrandet habe, befinden sich Ziffern, die weiter unten kurz erklärt werden.

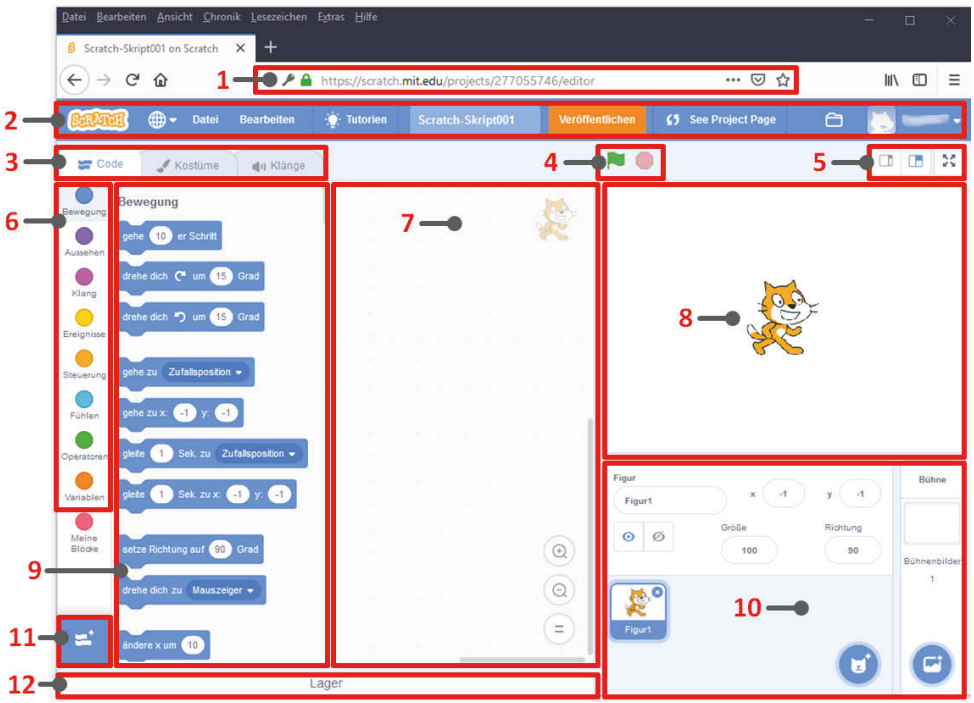


Abb. 2.3: Die Programmieroberfläche von Scratch 3.0

Nummer	Bedeutung
1	URL der Online-Scratch-Adresse
2	Menüzeile
3	Tab-Reiter für Code, Kostüme und Klänge
4	Start und Stopp des Skriptes
5	Dimensionierung der Oberfläche
6	Block-Paletten
7	Block- bzw. Skript-Bereich
8	Scratch-Bühne
9	Blöcke innerhalb der ausgewählten Palette
10	Eigenschaften der angewählten Figur bzw. des Hintergrundes
11	Erweiterungen hinzufügen
12	Lager

Tabelle 2.1: Elemente der Scratch-Oberfläche

Was fällt dir auf Anhieb hier auf? Nun, es ist, wie ich schon angekündigt habe, alles ist visuell. Du musst nicht eine einzige Zeile Code eingeben! Die Oberfläche ist in unterschiedliche Bereiche unterteilt, die logisch voneinander getrennt sind und jeweils andere Funktionen haben. Sie sind von mir rot umrandet. Alles ist für eine sehr intuitive Arbeit vorbereitet, so dass auch Einsteiger in kürzester Zeit zu beeindruckenden Ergebnissen kommen. Da gibt es eine sogenannte Bühne (in der Abbildung 2.3 der Bereich *8*), die sich ganz rechts befindet und auf der sich alle sichtbaren Aktivitäten abspielen. Sie wird im Moment noch von der orange-farbenen Katze auf der großen weißen Fläche in Beschlag genommen.

Ein Objekt auf der Bühne – ich sagte es schon einmal – wird in Scratch *Figur* genannt. Was die Akteure auf der Bühne für eine Show abziehen, wird durch Code gesteuert, der dem Computer mitteilt, was passieren soll. Dieser Code setzt sich aus den unterschiedlichsten Blöcken zusammen, die du hier als Beispiel in der geöffneten Block-Palette *Bewegung* siehst (Bereich *9*) und die in der Farbe Hellblau gekennzeichnet sind. Andere Block-Paletten sind mit abweichenden Farben versehen und dadurch gut zu unterscheiden. Die Eingabe über die Tastatur wird auf ein Minimum gesenkt und du musst wirklich nur selten deine Computertastatur verwenden. Dabei kann es nicht zu Situationen kommen, dass das erstellte Programm bzw. Skript abstürzt, weil du vielleicht einen Fehler in der Programmierung eingebaut hast. Das schließt jedoch nicht die logischen Fehler aus, die dazu führen, dass der Ablauf nicht in der geplanten Weise erfolgt.

Bevor ich zu einem ersten Beispiel komme, möchte ich dir die grundlegende Technik zeigen, die bei einer oder mehreren Figuren zum Einsatz kommt. Jeder Figur und sogar auch dem Bühnenbild sind drei unterschiedliche Bereiche zugeordnet. Im folgenden Schaubild ist das verdeutlicht.

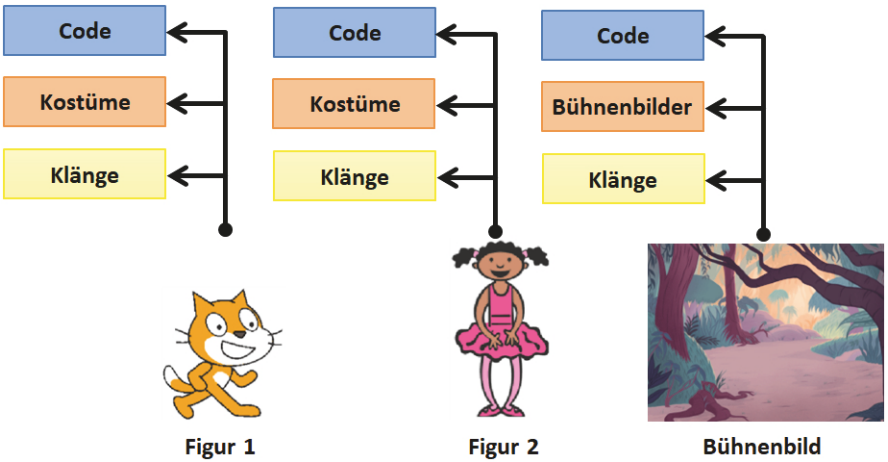


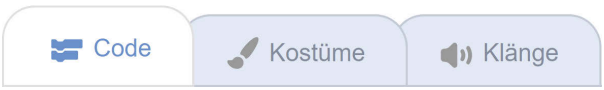
Abb. 2.4: Zwei Figuren und ein Bühnenbild

Damit die einzelnen Figuren, die du auf der Bühne platzierst und auch das Bühnenbild (Hintergrund) eigenständig und unabhängig voneinander agieren können, erhält jedes Objekt folgende separate Bereiche für

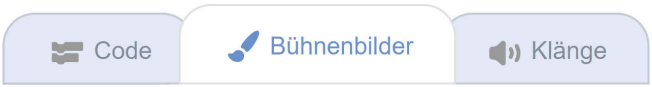
- Code (das wurde in Scratch 2 *Skripte* genannt) – sichtbar bei Figur und Bühnenbild
- Kostüme – nur sichtbar bei einer Figur
- Bühnenbilder – nur sichtbar bei einem Bühnenbild
- Klänge – sichtbar bei Figur und Bühnenbild

Hast du eines der Objekte in der Figurenliste bzw. das Bühnenbild mit der Maus ausgewählt, kannst du über bestimmte Tabulatoren am linken oberen Rand in die Bereiche wechseln, um sie mit gewünschten Aktionen bzw. Informationen zu versehen.

Die Tabulatoren für Figuren:



Die Tabulatoren für Bühnenbild:



Ein sehr einfaches Beispiel

Zum Einstieg schlage ich vor, ein sehr einfaches Beispiel für eine Animation zu programmieren, in dem du die Katze auf der Bühne einige Schritte in eine bestimmte Richtung

gehen lässt. Die für dieses Beispiel benötigten Blöcke sind in drei verschiedenen Paletten zu finden. Wenn einer oder mehrere Blöcke ausgeführt werden soll, dann gibt es dafür folgende Möglichkeiten: Entweder klickt man irgendwo innerhalb des betreffenden Codeblocks oder verwendet einen sogenannten HAT-Block mit der grünen Flagge. Dieser Block befindet sich in der *Ereignisse*-Palette.

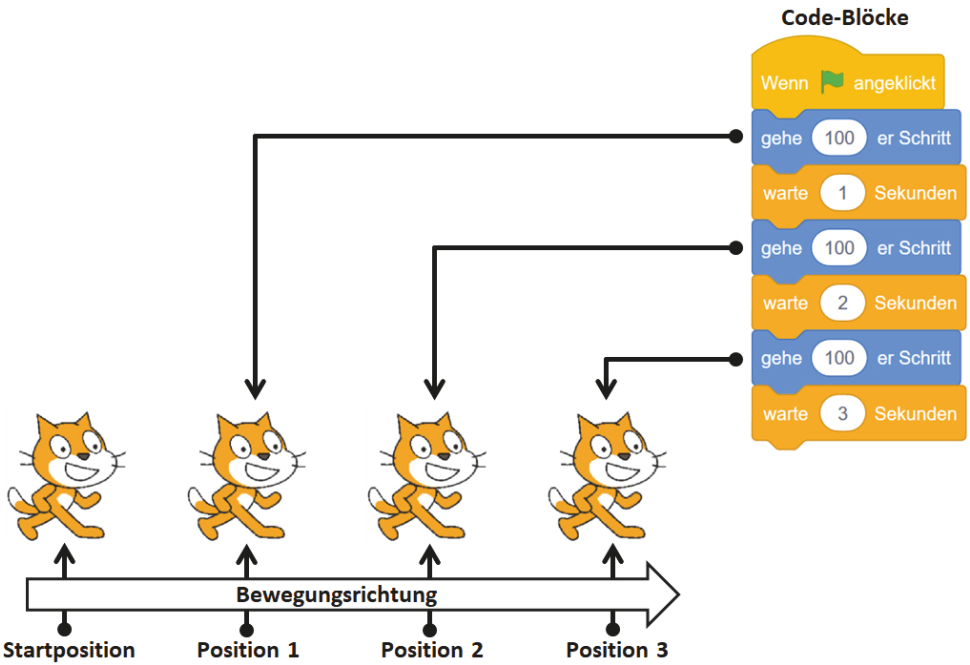


Abb. 2.5: Die Katze von links nach rechts bewegen

Bevor es jedoch losgeht, ist es wichtig, wie du einen Block aus einer vorhandenen Palette entnehmen und im Codebereich platzieren kannst. Die Handhabung der Blöcke ist wichtig für das Arbeiten mit Scratch. Blöcke auszusuchen, zu platzieren, zu löschen und zu verschieben sind Fertigkeiten, die du bald drauf haben wirst. Doch ich greife schon zu weit vor. Ich gehe am besten Schritt für Schritt vor.

Die Auswahl eines Blocks aus einer Palette

Wenn du einen Block aus einer Block-Palette in den Skript-Bereich übernehmen möchtest, klicke den gewünschten Block mit der linken Maustaste an und halte ihn gedrückt. Ziehe dann den Block nach rechts in den Block- bzw. Skript-Bereich und lasse ihn dort los. Das nennt man auch *Drag & Drop*. Beim Start der Scratch-3-Entwicklungsumgebung wird immer die Block-Palette *Bewegung* angezeigt. Ich komme gleich noch genauer darauf zu sprechen.

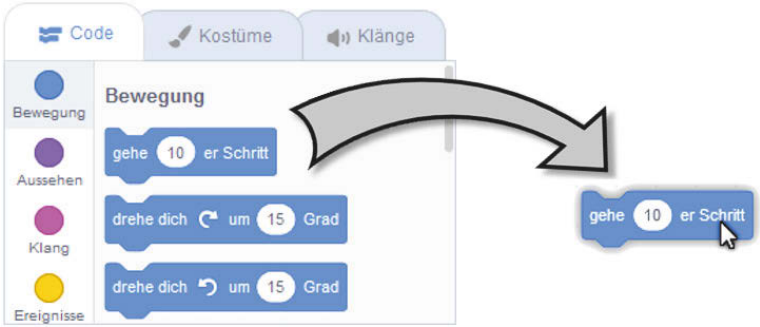


Abb. 2.6: Das Hinzufügen eines Blocks per Drag & Drop in den Skript-Bereich

Das Zusammenfügen mehrerer Blöcke

Möchtest du, dass mehrere Blöcke hintereinander ausgeführt werden, müssen sie miteinander verbunden werden. Du gehst wie folgt vor:

Den anzufügenden Block an den schon vorhandenen annähern:

Halte die linke Maustaste gedrückt und nähere dich von unten langsam an den oberen Block an.

Annäherung wurde erkannt:

Wenn du einen bestimmten Abstand unterschritten hast, meldet Scratch eine bevorstehende Kontaktaufnahme und es erscheint unterhalb des oberen Blocks eine weiße Linie.

Verbindung herstellen:

Jetzt kannst du die linke Maustaste loslassen und der untere Block schnappt automatisch ein, so dass eine Verbindung hergestellt wurde. Eine 100 % exakte Positionierung ist also nicht notwendig. Probier es aus, du wirst es schnell beherrschen.

Blöcke, die logisch nicht zusammenpassen, werden auch nicht ineinander schnappen, auch wenn du es immer und immer wieder probierst. Dadurch hast du eine gute visuelle Rückmeldung, welche Blöcke du miteinander kombinieren kannst und welche nicht. Probiere das selbst mit den unterschiedlichsten Blöcken aus. Verbinde einige Blöcke miteinander und versuche sowohl einen einzelnen als auch mehrere vom Gesamtblock zu separieren. Führe diese Aktionen mit unterschiedlichen Konstellationen durch, so dass du ein Gefühl für das Verhalten bekommst.

Das Entfernen eines Blocks

Wenn du den gerade angefügten Block wieder entfernen möchtest, ziehe ihn mit gedrückter linker Maustaste zurück auf die Block-Palette. Wenn du dann die linke Maustaste los lässt, ist der Block gelöscht. Bei mehreren zu entfernenden Blöcken musst du den obersten Block zurück auf die Block-Palette ziehen. Alle darunterhängenden Blöcke werden automatisch mitgezogen. Du verstehst das Verhalten der Blockverschiebungen bzw. -löschungen am besten, wenn du es einfach ausprobierst.

Abb. 2.7: Das Entfernen eines Blocks per Drag & Drop

Eine weitere Möglichkeit besteht im Aufruf des Kontextmenüs über die rechte Maustaste. Dort gibt es unter anderem den Punkt *Lösche Block*.

Abb. 2.8: Das Entfernen eines Blocks per Kontextmenü