Hauke Fehr

Let's code Scratch!

Aktuell zu Scratch 3

+ Programmieren lernen – nicht nur für Kinder!
+ Spiele und mehr: Labyrinth, Springball, Space Shooter, Handgestensteuerung
+ Alle Programmier-Basics: Abfragen, Schleifen, Variablen und Co.

Vierfarben

Kapitel 1
Spielend Programmierer werden – mit Scratch

Möchtest du Programmierer werden? Oder willst du einfach nur wissen, wie du leicht eigene, richtige Spiele bauen kannst? Mit diesem Buch wirst du beides erreichen: Eigene Spiele erstellen macht Spaß – nach und nach lernst du dabei automatisch zu denken wie ein Programmierer. Und am Ende bist du tatsächlich einer!

Herzlichen Glückwunsch! Du hast beschlossen, *Scratch* zu erlernen. Das ist der Anfang eines großen und sehr spannenden Weges.

Den Computer bedienen können viele – und Spiele spielen sowieso. Aber eigene Spiele programmieren ist etwas ganz anderes.

Ist das nicht viel zu schwierig?

Nein, behaupte ich, ist es nicht. Vor allem nicht mit *Scratch*.

Muss man dazu nicht Informatik studieren oder zumindest ein Mathegenie sein?

Nein, das ist nicht nötig. Wirklich nicht.

Was braucht man denn, um Scratch-Spieleprogrammierer zu werden?

Eigentlich vor allem Lust dazu und Spaß daran. Vorkenntnisse sind nicht so wichtig. Wenn du gerne mit dem Computer herumspielst und schon immer gern mal ein Spiel selber gestalten wolltest oder neugierig bist, wie man so etwas macht, dann bringst du schon alles mit, was man braucht, um *Scratcher* zu werden – also ein Spieleprogrammierer mit *Scratch*. Den Rest lernst du Schritt für Schritt in diesem Buch.

Programmieren ist leichter, als du denkst

Viele glauben, dass das Programmieren von Computern im Grunde etwas mit höherer Mathematik zu tun hat. Man müsse dazu sehr komplizierte Zusammenhänge begreifen. Und mit schwieriger Technik: Es gibt Begriffe wie »Prozessoren«, »RAM-Speicher«, »Datenstrukturen«, »Protokolle« ..., dazu eine Menge völlig abstrakter, rätselhafter Befehle, die man lernen und begreifen müsse ... es scheint fast unmöglich.

Diese Vorstellung stammt aus einer Zeit, als die Computer noch jung waren. Damals konnte man nur programmieren, wenn man die gesamte Technik des Computers in- und auswendig beherrschte.

Man programmierte damals in Maschinensprache. Am Anfang waren die Programme in Lochkarten gestanzt, damit man den Computer überhaupt damit füttern konnte. Die Einführung von Tastatur und Bildschirm war schon eine große Erleichterung. Ein ganz simples Programm, das nur die beiden Wörter »Hallo Welt!« auf den Bildschirm schreibt, sieht in (vereinfacht geschriebenem) Maschinencode etwa so aus:

```
segment code

 start:
 mov ax, data
 mov ds, ax

 mov dx, hallo
 mov ah, 09h
 int 21h

 mov al, 0
 mov ah, 4Ch
 int 21h

 segment data
 hallo: db 'Hallo Welt!', 13, 10, '$'
```

Klar, dass programmieren so früher eine echte Herausforderung war.

Zum Glück hat die Technik sich heute erheblich weiterentwickelt. Man könnte zwar noch immer Programme in Maschinencode schreiben, denn die Hardware basiert noch immer darauf – aber das muss heute so gut wie niemand mehr machen.

Was bedeutet denn eigentlich »programmieren«?

Programmieren heißt im Grunde nichts anderes, als *einem Computer zu sagen, was er tun soll*, damit er genau das macht, was man selber möchte.

Früher musste man dazu ganz genau verstehen, wie der Computer im Inneren arbeitet. Man musste ihm genau sagen, wann und wie er Einsen und Nullen durch seinen Speicher zu schieben hatte, damit – nach viel Testen – wirklich das passierte, was man wollte. Heute ist es die Aufgabe des Computers zu verstehen, was wir meinen, wenn wir ihm eine Anweisung geben. Wir sagen dem Computer in leicht verständlicher Sprache, was er tun soll, und der Computer weiß, wie er das in seinem Inneren umsetzen muss.

Scratch macht vor allem Spaß

Und besonders einfach geht das Programmieren mit *Scratch*. *Scratch* ist erst einmal wie ein buntes Spiel, ein Logikbaukasten, der von Anfang an Spaß macht. Und doch wirst du damit beim Herumspielen nach und nach wie von selbst richtig ernsthaftes Programmieren lernen.

Um in *Scratch* zu programmieren, musst du nicht einen einzigen Befehl tippen oder komplizierte Codes lernen. Alles, was du tust, ist, *Figuren* auf deiner *Theaterbühne* Anweisungen zu geben, indem du klar verständliche Anweisungsblöcke aus der Code-Bibliothek in ihr Programmfenster ziehst und sie mit anderen Blöcken verbindest. Und damit machst du am Ende das Gleiche wie ein Programmierer in früheren Zeiten, der Tausende von Zeilen schwierige Befehle eintippte. Nur komplett befreit vom technischen Ballast. Um die Technik kümmert *Scratch* sich selber.

In *Scratch* sieht ein Programm, das »Hallo Welt!« sagt, einfach so aus:

Das ist verständlicher als Maschinensprache, oder?

Was ist Scratch?

Scratch ist eine kostenlos nutzbare Programmiersprache und visuelle Entwicklungsplattform, die von amerikanischen Wissenschaftlern entwickelt wurde und erstmals im Jahr 2007 erschien. Sie ist speziell dafür erstellt worden, Kindern, Jugendlichen und allen Einsteigern eine einfache, aber dennoch leistungsfähige Möglichkeit zu geben, selber Spiele und animierte Programme zu bauen, die ohne kompliziertes Tippen von Befehlen und das

Lernen von Code erstellt werden können. Damit können Anfänger sich perfekt mit den Grundlagen des Programmierens vertraut machen und gleichzeitig motivierende und individuelle grafische Spiele nach ganz eigenen Vorstellungen umsetzen. *Scratch* ist in viele Sprachen übersetzt worden und wird heute an unzähligen Schulen und Universitäten weltweit eingesetzt, um Schüler und Studenten an die Prinzipien der Programmierung heranzuführen. Die neueste Version, *Scratch 3*, erschien im Jahr 2019 und wird stetig weiterentwickelt.

Lernen durch Spielen

Ich verspreche dir: Wenn du diesem Buch von Anfang an Kapitel für Kapitel folgst und die Beispiele selbst ausprobierst, nachbaust und damit herumspielst, wirst du eine Menge Spaß haben, Figuren steuern, laufen und tanzen lassen, witzige Animationen daraus machen und sie auf deine Eingaben reagieren lassen. Du wirst spannende Code-Experimente machen und am Ende nach den ersten einfachen Versuchen schon bald selber richtige, coole Programme und Spiele bauen, die du ganz nach deinen Ideen gestalten und immer weiter ausbauen kannst.

Und dabei lernst du gleichzeitig programmieren, zuerst fast, ohne es zu merken. Richtiges Programmieren – genau dieselben Techniken und Methoden, die Profi-Programmierer tagtäglich verwenden. Nur dass es in *Scratch* von Anfang an Spaß macht.

Du brauchst dafür keine Mathematik (außer vielleicht ab und zu ein bisschen plus und minus), keine Kenntnisse über den Aufbau des Computerspeichers, über Datenformate, den Prozessor oder Sonstiges, du musst kein hochbegabter Nerd sein, auch kein genialer Erfinder. Jeder kann in *Scratch* seine eigenen Ideen umsetzen – auf niedrigem oder hohem Level. Du brauchst nur die Lust daran, eine neue kleine Welt zu entdecken, etwas Eigenes zu bauen und zu gestalten, sowie ein wenig Freude an logischem Denken. Der Rest kommt dann von selbst.

Und hinterher bist du *Scratcher* – ein richtiger Programmierer –, das ist versprochen!

Aber eins nach dem anderen. Erst einmal musst du *Scratch* kennenlernen – deine neue Heimat für die nächste Zeit. Ich bin sicher, du wirst dich schnell darin wohlfühlen.

Kapitel 2
Ganz einfach: so kriegst du Scratch auf deinen Computer

Der Einstieg ist wirklich ganz leicht. Um mit Scratch zu arbeiten, brauchst du eigentlich nur einen Computer oder Laptop – dazu einen Webbrowser, der dort sowieso schon drauf ist. Und los geht's!

Es gibt zwei Wege, *Scratch* auf deinem Computer zu verwenden. Der erste dauert nur ein paar Sekunden, und es kann sofort losgehen. Dafür brauchst du aber eine ständige Verbindung mit dem Internet, die nicht zu langsam ist. Die meisten Computer heutzutage haben das, wenn sie zu Hause stehen.

Die zweite Möglichkeit dauert ein paar Minuten. Aber damit kannst du *Scratch* fest auf dem Computer installieren und überall benutzen – größtenteils auch ohne Internet. Wenn die erste Methode bei dir nicht funktioniert, dann kannst du jederzeit die zweite Methode verwenden, sofern dein Gerät es erlaubt. Auf die eine oder andere Weise wirst du es auf jeden Fall hinbekommen und kannst in kürzester Zeit loslegen!

Erste Methode: Scratch direkt im Webbrowser

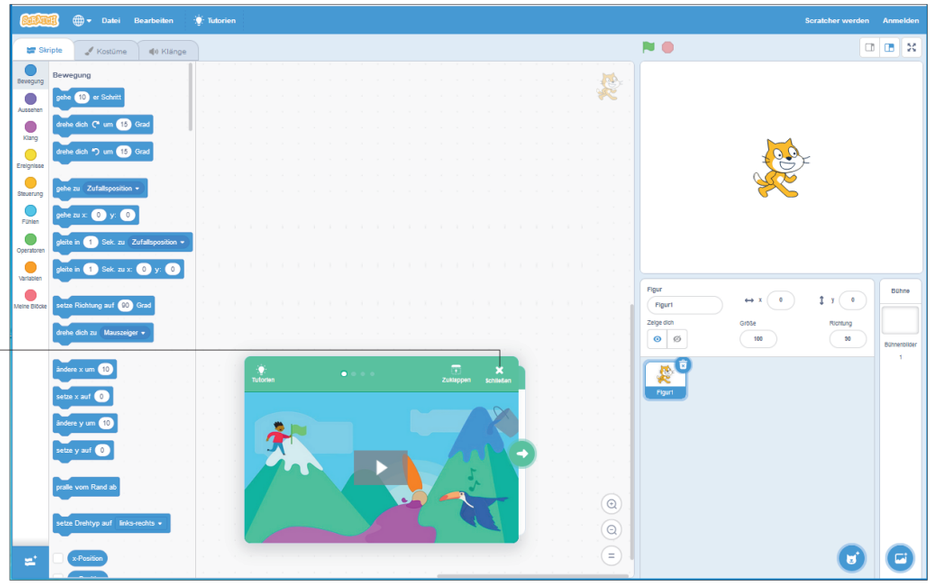
Es ist wirklich supereinfach. Egal, ob du einen Mac, einen Windows-PC oder einen Linux-Rechner verwendest, ja, sogar auf einem Tablet oder iPad funktioniert es: Starte einfach den »normalen« Webbrowser auf deinem Gerät. Zum Beispiel den Internet Explorer, den Edge-Browser, Firefox, Safari oder Chrome. Ganz wie du möchtest und was auf deinem System verfügbar ist – nur allzu alt sollte der Browser nicht sein.

Jetzt gibst du in der URL-Zeile des Browsers, also ganz oben, Folgendes ein und drückst die ↵-Taste:

https://scratch.mit.edu

Das war's eigentlich schon. Anschließend klickst du nur noch im oberen Menü auf **Entwickeln**.

Warte ein paar Sekunden – und schon bist du mittendrin im *Scratch*-Editor. Herzlichen Glückwunsch! Es kann losgehen.



❶

Hier siehst du den Scratch-Editor. Das kleine »Tutorial« kannst du mit Klick auf »Schließen« ❶ beenden.

Du solltest dir am besten gleich ein Lesezeichen (Bookmark) für diese Seite im Browser anlegen. Dann kommst du immer wieder mit einem Klick dorthin.

Jetzt bist du bereit und kannst schon ins dritte Kapitel wechseln.

Zweite Methode: Scratch-Desktop-Editor fest auf dem Computer installieren

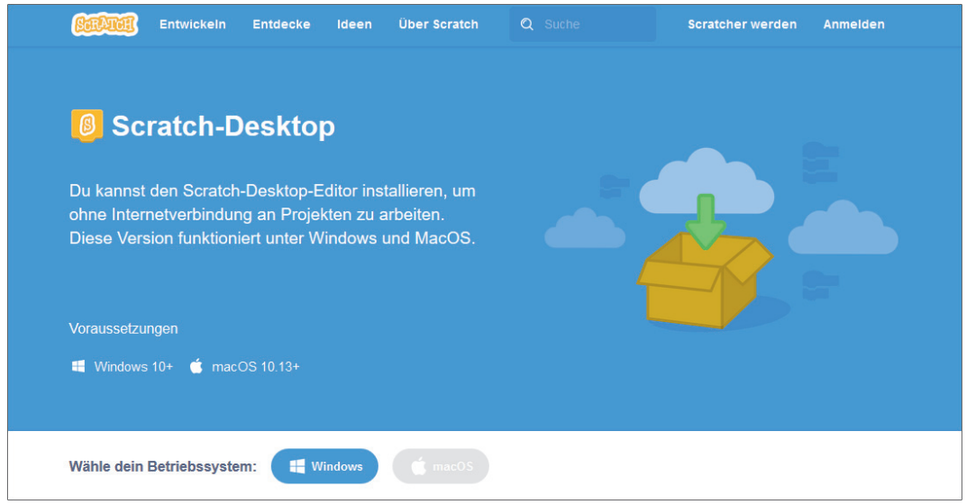
Wenn du keinen ständigen Internetzugang hast oder das Programm einfach gerne fest auf deinem Computer installieren möchtest, damit du es jederzeit

überall verwenden kannst, dann ist das auch nicht schwer, sofern du einen halbwegs aktuellen Windows-PC oder einen Mac-Computer hast. Scratch steht als kostenloses Desktop-Programm zur festen Installation zur Verfügung.

Hierzu gehst du mit einem beliebigen Webbrowser auf diese Seite:

https://scratch.mit.edu/download

Scratch Desktop kann zur Zeit auf jedem PC installiert werden, der Windows 10 als Betriebssystem hat – sowie auf jedem Mac-Computer, der mindestens mit macOS 10.13 ausgestattet ist. Auf Linux-Computern, Chromebooks oder Android/iOS-Geräten kann derzeit noch keine feste Version von *Scratch* installiert werden – das wird sich allerdings in der Zukunft, nach Erscheinen dieses Buches, wohl noch ändern, denn auch für diese Geräte sind Versionen angekündigt.



Wähle hier einfach dein Betriebssystem (Windows oder macOS), klicke auf **Herunterladen**, und starte dann die geladene Datei per Doppelklick (Windows), oder ziehe sie in den Ordner *Anwendungen* (Mac).

Anschließend findest du das *Scratch*-Symbol auf deinem Desktop (Windows) bzw. in deinem Anwendungsordner (Mac) und kannst *Scratch* von nun an jederzeit damit starten.

Das war alles. So oder so – es ist ganz einfach!

Kapitel 10
Springball – du beginnst dein erstes Spiel

Nun hast du die wichtigsten Elemente des Programmierens kennengelernt und ausprobiert. Jetzt kann es richtig losgehen. Du wirst damit anfangen, dein erstes richtiges Spiel zu programmieren. Natürlich wirst du dabei noch eine Menge praktische Techniken dazulernen.

Es ist so weit: Wir werden uns an ein Spiel machen. Ein richtiges Spiel, das du später auch gerne weiter ausbauen kannst.

Die Spielidee

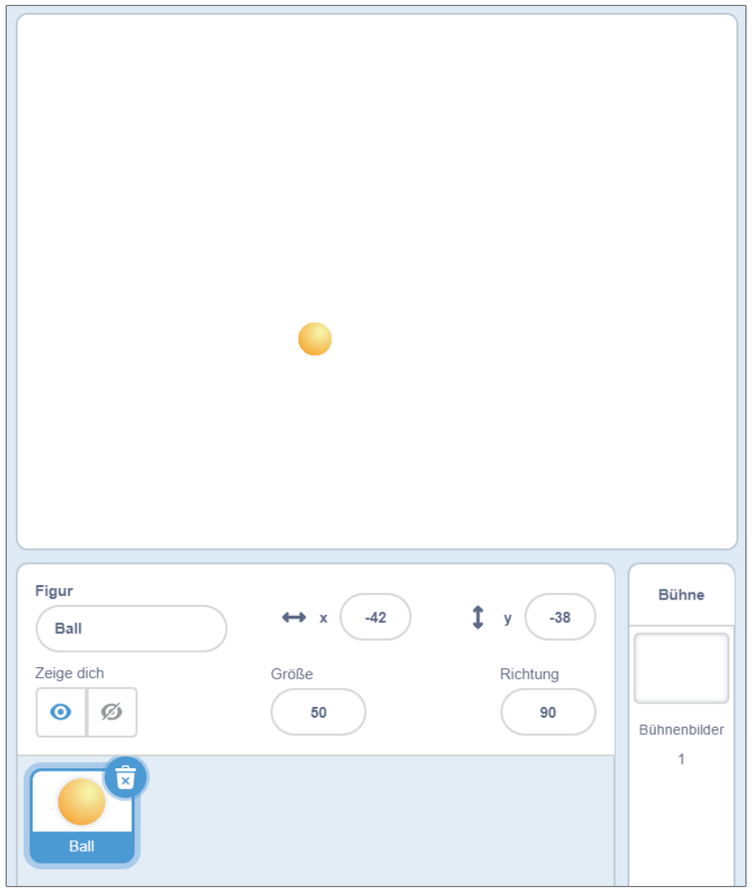
Wir wollen das Spiel *Springball* bauen, das auf dem Grundprinzip des berühmten Spieleklassikers *Breakout* basiert. *Breakout* war eines der ersten Computerspiele überhaupt und entstand bereits in den 1970er-Jahren. Seitdem sind zahlreiche Varianten des Spiels erschienen.

Das Spiel geht so: Es fliegt ein Ball über die Bühne, der links, rechts und oben vom Rand abprallt und den du mit einem Brett am unteren Rand immer wieder abfangen kannst. Wenn der Ball den Boden berührt, ist das Spiel zu Ende. Im nächsten Schritt sollen dann auch noch Ziele hinzukommen, die du mit dem Ball abschießen musst. Und zuletzt soll es dann auch noch Punkte geben und ein Spielende, bei dem man gewinnen oder verlieren kann.

Puh … da steht uns einiges an Arbeit bevor. Aber es ist wahrscheinlich trotzdem leichter, als du denkst. Die meisten Elemente, die du dafür brauchst, kennst du nämlich schon – und ein paar neue Kniffe wirst du kennenlernen. Und wenn man alles Schritt für Schritt nacheinander aufbaut, bleibt die Arbeit immer übersichtlich und verständlich. So machen es auch gute Profi-Programmierer.

Schritt 1: Der fliegende Ball

Starte *Scratch* neu, lösche die Katze, und wähle als Figur einen einfachen gelben Ball (Name: **Ball**), den du verkleinerst (auf 50 %). Das ist erst einmal alles, was du brauchst.



Was soll der Ball im ersten Schritt tun? Er soll herumfliegen und am Rand der Bühne abprallen. Das ist in *Scratch* wirklich einfach, und du hast es ja schon mehrmals gemacht. Du setzt ein Startsignal (zum Beispiel die Flagge) und gibst dem Ball eine Anfangsrichtung, sagen wir mal 50 Grad, damit er schräg nach

oben fliegt. Er braucht natürlich auch eine Anfangsposition, damit er immer schön in der Mitte beginnt. Ich schlage Position 0, –120 vor.

Du brauchst außerdem eine fortlaufende Wiederholung: Der Ball soll sich ständig einen 10er-Schritt vorwärtsbewegen und, wenn nötig, am Rand abprallen. Alles, wie du es schon kennst.

Kannst du das selbst bauen? Versuche es ruhig einmal!

So könnte es zum Beispiel aussehen. Das solltest du einmal testen und sehen, wie der Ball über die Bühne fliegt.

Zwischenspiel: Der Ball zeichnet seine Bahn

Eigentlich nicht wirklich wichtig für unser Spiel – aber interessant, wenn du mal sehen willst, welche Bahn der Ball genau fliegt. Und außerdem cool.

Du brauchst dazu die **Malstift**-Befehle, die zu den Erweiterungen von *Scratch* gehören.

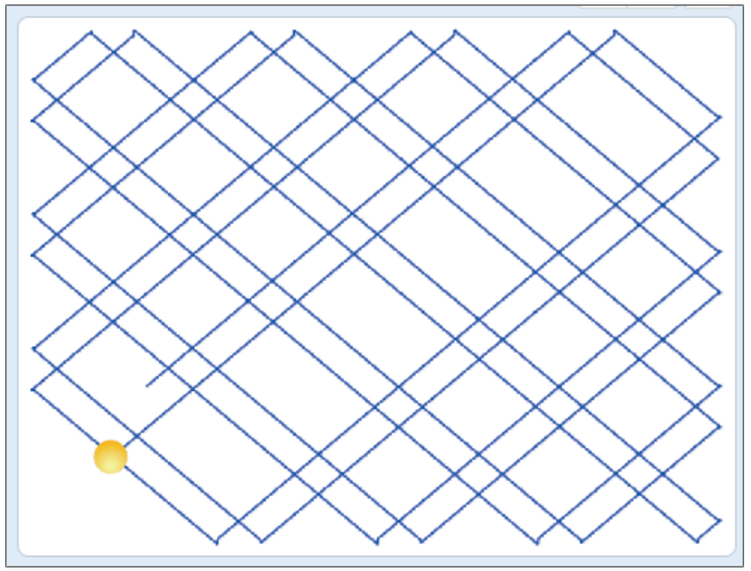
Klicke auf das Feld **Erweiterungen** ganz links unten in Scratch.

Wähle dort die Erweiterung **Malstift** aus.

Nun stehen dir die **Malstift**-Befehle zur Verfügung.

Du musst sie in diesem Fall gar nicht unbedingt in dein Programm einbauen. Klicke einfach mal auf den Befehl `schalte Stift ein`. Dann hat unser Ball einen Stift und malt, während er sich bewegt.

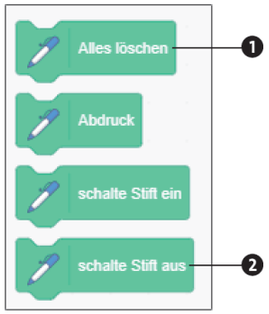
Starte anschließend den Ball.



Jetzt zeichnet der Ball beim Herumfliegen ein tolles Muster auf die Bühne.

Wow – das sieht cool aus! Und es zeigt, dass der Ball wirklich nach und nach alle Bereiche der Bühne überfliegt.

Wenn du willst, kannst du damit noch herumspielen – die Stiftfarbe ändern, die Dicke usw. … das geht alles mit den Malbefehlen. Wenn du fertig bist, klickst du einmal auf den obersten Befehl `Alles löschen` ❶ – und dann noch auf den vierten Befehl `schalte Stift aus` ❷.



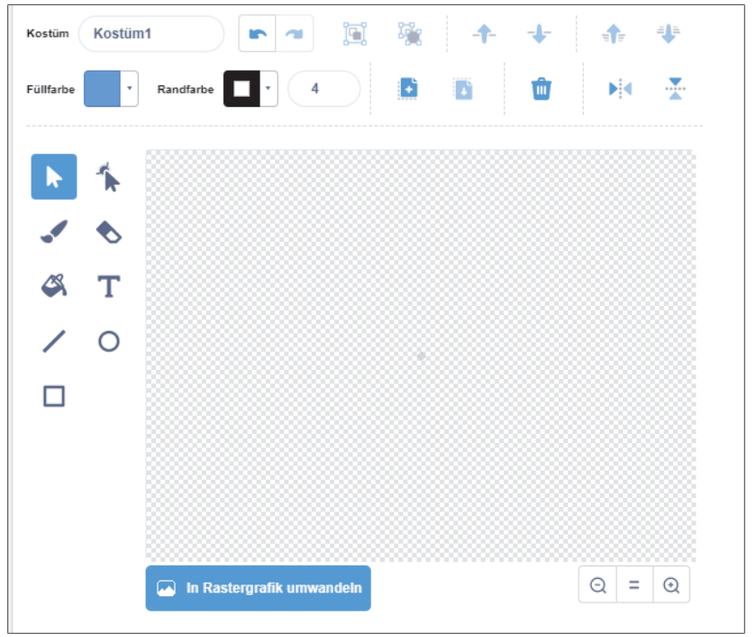
Jetzt ist alles wieder wie vorher, und wir kehren zu unserem Spiel zurück.

Das Schlagbrett

Wir gehen zum nächsten Schritt: Das Schlagbrett kommt dazu, mit dem wir den Ball unten abstoßen wollen. Das Brett soll sich ja nach links und rechts bewegen können, es sollte deshalb also eine neue Figur sein. In der Bibliothek findest du vielleicht eine irgendwie geeignete Figur, die man verwenden könnte, oder du könntest dir auch ein Bild aus dem Internet laden.

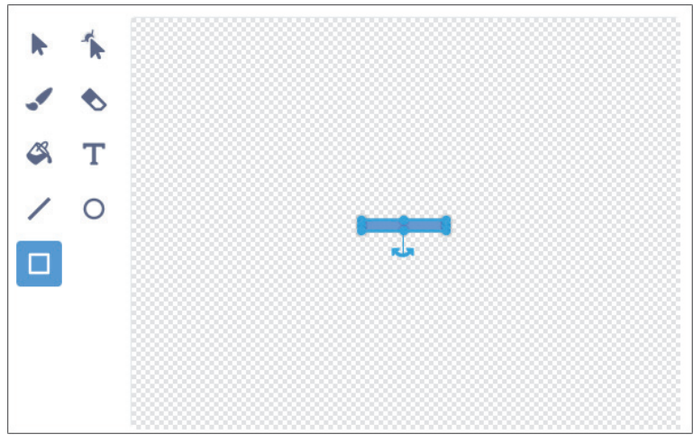
Aber es geht auch so: Mal dir einfach selbst eine Figur! So ein Brett ist wirklich schnell erstellt. Und dann sieht es wirklich genauso aus, wie du es haben möchtest. Dazu klickst du unter der Figurenbibliothek bei **Neue Figur** auf den Pinsel:

Nun bist du im grafischen Kostüm-Editor, wo du das Aussehen deiner neuen Figur selbst erstellen kannst.

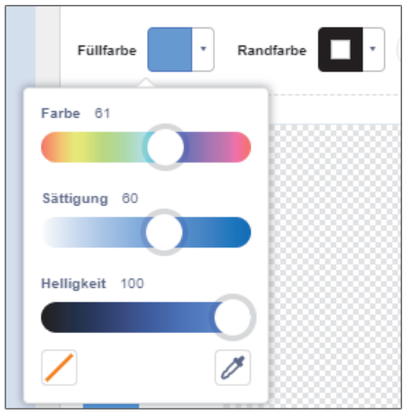


Was du damit alles anstellen kannst, steht ausführlich in Kapitel 5 – dort kannst du auch gern noch einmal nachschlagen.

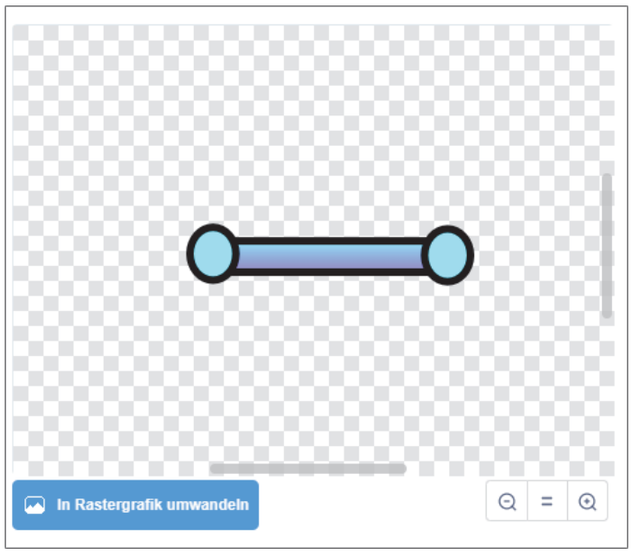
Unser Schlagbrett ist ganz einfach erstellt. Wähle das Rechteck aus, und ziehe mit der Maus ein Rechteck möglichst genau in der Mitte der Zeichenfläche auf.



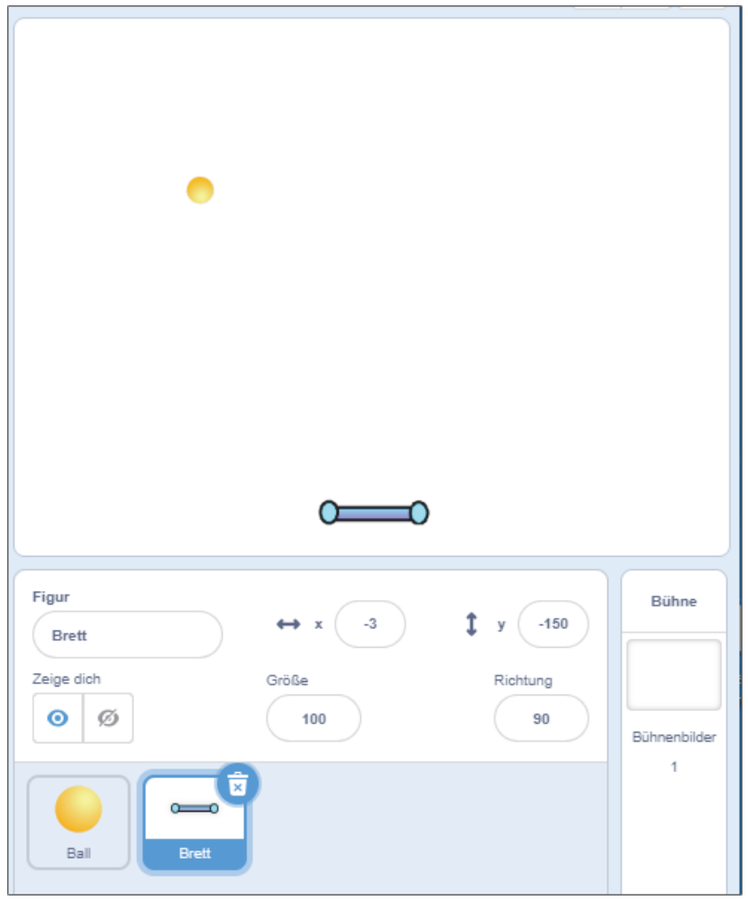
Mit den Anfassern kannst du die Größe noch verändern, bis sie optimal ist. Nun legst du eine schöne Farbe für das Rechteck fest, indem du auf **Füllfarbe** klickst.



Mit den drei Reglern kannst du die Grundfarbe, Sättigung und Helligkeit verändern, bis die Farbe dir gefällt. Du kannst auch einen Farbverlauf als Füllfarbe wählen, dann sieht es noch etwas raffinierter aus. Aber das ist ganz dir überlassen. Wenn du willst, kannst du auch noch Verzierungen an den Seiten anbringen, zum Beispiel zwei Kreise.



Mit dem Lupensymbol (+) kannst du heranzoomen, um die Grafik besser bearbeiten zu können. Wenn du zufrieden bist, kannst du zurück ins Skriptfenster gehen. Du hast jetzt eine neue Figur, die nennst du **Brett** und schiebst sie nah an den unteren Rand der Bühne.



Super! Jetzt haben wir die beiden wichtigsten Figuren, und das Spiel kann weiterprogrammiert werden.

Das Brett nach rechts und links steuern

Das Brett soll nach links und rechts bewegt werden können. Dafür müssen wir jetzt ein Programm erstellen. Es gibt hier zwei verschiedene Möglichkeiten. Wir werden sie beide ausprobieren, und du kannst dann entscheiden, welche du verwenden willst. Du kannst das Brett nämlich entweder mit der Tastatur (den Pfeiltasten) steuern oder mit der Maus.

Steuerung mit der Tastatur

Wie steuert man das Brett mit den Pfeiltasten? Nun, ganz einfach: Wenn du den Pfeil nach links drückst, soll das Brett nach links wandern, wenn du den Pfeil nach rechts drückst, soll das Brett nach rechts wandern. Nach oben oder unten soll das Brett nicht gehen können.

Damit die Steuerung in diesem Spiel schön flüssig wird, solltest du in deinem Brett-Programm die Pfeiltasten ständig (in einer fortlaufenden Wiederholung) abfragen und immer, wenn eine davon gedrückt ist, das Brett nach links oder rechts bewegen.

Profi-Frage: Warum in einer Wiederholungsschleife die Tastatur abfragen?

Warum kannst du nicht einfach das Ereignis `Taste gedrückt` benutzen wie in Kapitel 8?

Der Grund ist: Das Brett soll sich beim Drücken der Tasten ständig sanft und gleichmäßig hin und her bewegen. Du bekommst eine Spielsteuerung, wie sie hier gebraucht wird, so aber nicht schnell und gleichmäßig hin, denn das Tastaturereignis wird nur einmal beim Drücken ausgelöst, und wenn du die Taste gedrückt hältst, erst nach einer halben Sekunde wieder. Wenn du Spielfiguren durchgehend flüssig hin und her steuern willst, solltest du das immer in einer Dauerwiederholung mit Abfrage machen, nicht über Tastaturereignisse. Glaub mir, das ist hier der beste Weg.

Jetzt geht's ans Programmieren:

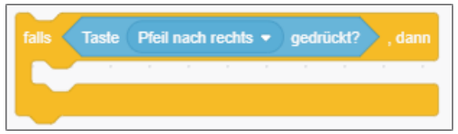
1. Achte darauf, dass du das Skriptfenster geöffnet hast und das Brett in der Bibliothek angewählt ist.

2. Hole dir einen `falls-dann`-Abfrageblock aus der Abteilung **Steuerung**.

3. Ziehe eine Tastaturabfrage aus der Abteilung **Fühlen** hinein. Das ist eine Abfrage, ob eine Taste gedrückt ist.

Wir wollen ja hier nicht die Leertaste prüfen, sondern den Pfeil nach rechts.

4. Ändere die abgefragte Taste also in `Pfeil nach rechts`.



Nun haben wir eine Abfrage, die prüft, ob der Pfeil nach rechts auf der Tastatur gedrückt wird.

Was soll in dem Fall passieren?

Klar – das Brett soll ein Stück nach rechts wandern. Wir könnten das Brett jetzt nach rechts drehen und dann einen Schritt vorwärtsgehen, wie wir es mit dem Käfer gemacht haben. Aber da das Brett sich eigentlich nie drehen soll, sondern immer genauso bleiben soll, wie es ist, und nur seine Position nach rechts verschoben werden soll, benutzen wir hier einen anderen Befehl, der für Bewegungen »rechts – links – hoch – runter« sehr praktisch ist – und vor allem schnell.

Mit diesem Befehl wird die x-Position einer Figur (also hier des Bretts) um 10 verschoben, also um 10 nach rechts gerückt.

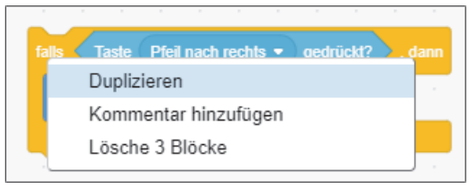
Position verschieben

Mit der x-Position verschiebt man Figuren nach links oder rechts, ohne sie zu drehen, mit der y-Position verschiebt man entsprechend Figuren nach oben oder unten.

5. Füge diesen Befehl in die Klammer des Abfragebefehls ein.

Alles klar? Es wird also geprüft, ob gerade die [→]-Taste gedrückt wird – wenn ja, dann wird das Brett um 10 Punkte nach rechts verschoben. `Ändere x um 10` heißt: 10 zur x-Position hinzuzählen. Jetzt brauchen wir das Gleiche noch einmal für die Bewegung nach links.

6. Dupliziere den ganzen Programmblock einmal, indem du ihn mit der rechten Maustaste oben anklickst und **Duplizieren** wählst.



7. Ändere den neuen Block so, dass die Taste [←] abgefragt wird und `ändere x um` auf `-10` steht.

Wenn jetzt also die linke Pfeiltaste gedrückt wird, ändert sich die x-Position des Bretts um –10. Das heißt, die x-Position wird um 10 verringert, und das Brett wird dadurch 10 Punkte nach links verschoben.

Nun haben wir die beiden Abfragen. Damit sie auch wirksam werden, müssen sie ununterbrochen durchlaufen werden, denn sonst würde sich das Programm nach dem ersten Prüfen sofort wieder beenden. Also packen wir die beiden Blöcke in eine fortlaufende Wiederholung.

8. Hole dir aus der Abteilung **Steuerung** einen Block `wiederhole fortlaufend`, und setze die beiden Abfragen dazwischen. Setze oben noch einen Flaggen-Ereignisbefehl drauf, damit das Programm mit der Flagge automatisch startet.

Fertig: deine perfekte Tastatur-Steuerung für das Brett. Wenn du jetzt die grüne Flagge klickst, starten die Programme für das Brett und für den Ball gleichzeitig, denn beide werden ja durch die Flagge in Gang gesetzt.

Das ist doch schon ein richtig guter Anfang unseres Spiels! Der Ball fliegt über die Bühne, und das Brett ist flüssig steuerbar.

Steuerung mit der Maus

Es gibt, wie gesagt, noch eine zweite Möglichkeit, das Brett zu steuern. Und zwar mit der Maus. Das ist sogar noch einfacher zu programmieren und hat

den Vorteil, dass man das Brett beim Spielen fast beliebig schnell hin und her bewegen kann. Es ist ganz allein deine Entscheidung, wie du die Steuerung machen möchtest – beides ist gut möglich. Hier erfährst du, wie man es mit der Maus macht:

Alles, was dabei passieren muss, ist, dass in einer Endlosschleife ständig und immer wieder die x-Position des Bretts auf die *aktuelle x-Position der Maus* gesetzt wird. Das Programm setzt das Brett also immer genau auf die Position des Mauszeigers, egal wo der Mauszeiger hinwandert. Wenn das geschieht, ist das Brett dadurch immer auf derselben waagerechten Höhe wie die Maus.

Probieren wir's:

1. Hole dir in das Skriptfenster des Bretts einen fortlaufenden Wiederholungsblock.

In dieser Schleife soll jetzt ständig die x-Position des Bretts neu gesetzt werden.

2. Hole dir aus dem Bereich **Bewegung** den Block `setze x auf`, und füge ihn in den Wiederholungsblock ein.

Nun soll die x-Position ja nicht ständig auf –3 gesetzt werden, auch nicht auf einen anderen festen Wert, sondern die x-Position soll ständig auf die aktuelle *x-Position des Mauszeigers* gesetzt werden.

Wie lässt du eine Figur mit der Maus mitwandern?

3. Hole dir den Block `Maus x-Position` aus der Abteilung **Fühlen**, und schiebe ihn in das Zahlenfeld des `setze x auf`-Blocks.

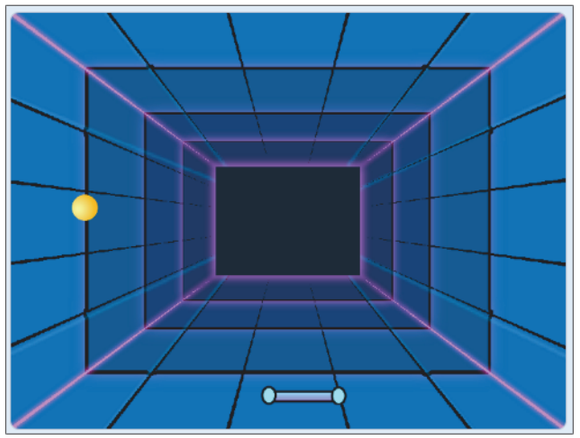
Statt einer festen Zahl wird jetzt hier immer die aktuelle Maus-x-Position verwendet.

Das war's schon! Damit haben wir uns so etwas wie einen eigenen Befehl zusammengebaut. *Setze die x-Position unserer Figur auf die x-Position des Mauszeigers*. Und das ist genau das, was wir jetzt brauchen, denn so können wir das Brett mit der Maus nach links und rechts verschieben.

Teste es mal, indem du den Programmblock anklickst und startest und dann mit der Maus über die Bühne fährst! Wenn du es verwenden willst, solltest du oben noch das Startereignis mit der Flagge draufsetzen.

Du musst dich jetzt nur noch entscheiden, welche Steuerung dir lieber ist, denn beide gleichzeitig funktionieren nicht. Lass ein Steuerungsprogramm da, und das andere kannst du löschen oder deaktivieren, indem du einfach den Flaggen-Ereignisbefehl entfernst. Damit wird es dann auch nicht mehr gestartet.

Nun noch ein schönes Bühnenbild – ich empfehle das Bild **Neon tunnel** aus der Bibliothek – und schon hast du eine tolle Basis für ein echtes Spiel.



Speichere das Spiel jetzt unbedingt ab, damit dir deine Arbeit nicht verloren geht.

Eine ganz wichtige Grundfunktion fehlt jetzt aber natürlich noch. Der Ball soll an dem Brett abprallen, sonst macht das Brett ja überhaupt keinen Sinn.

Pralle vom Brett ab

Da wird es ein bisschen raffinierter, denn es gibt zwar den tollen und praktischen Befehl `pralle vom Rand ab` – aber es gibt leider keinen vorgegebenen Befehl `pralle von einer Figur ab`. Diese Funktion müssen wir uns also selbst programmieren.

Falls der Ball das Brett berührt, soll er seine Richtung ändern, und zwar soll er genau wie beim Abprallen an der Wand diagonal in die entgegengesetzte Richtung fliegen, aus der er gekommen war. Es muss also ständig geprüft werden, ob der Ball das Brett berührt (wie das geht, weißt du ja schon), und wenn ja, dann muss seine Richtung *um 90 Grad verschoben werden*, damit er in die »gespiegelte Gegenrichtung« fliegt. Wie das geht, zeige ich dir jetzt: Wir gehen Schritt für Schritt vor.

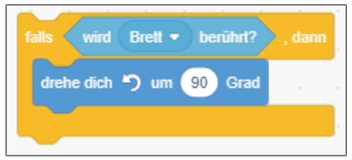
1. Wähle den Ball aus, denn wir arbeiten jetzt am Skript des Balls.

2. Hole dir eine `falls-dann`-Abfrage (Steuerung), und füge `wird Brett berührt` aus **Fühlen** hinein.

   Nun hast du eine Abfrage, die testet, ob der Ball das Brett berührt. Wenn nicht, dann passiert nichts, und der Ball fliegt weiter in seine Richtung. *Wenn doch – was passiert dann?*

   Dann kriegt der Ball eine neue Richtung. Damit er genau in die gespiegelt entgegengesetzte Richtung fliegt, muss er sich einfach um 90 Grad drehen. Für diese Drehung gibt es einen einfachen Befehl.

3. Wähle den Befehl `drehe dich um … Grad`, setze die Richtungsänderung auf `90`, und setze den Befehl in die Abfrage.



Sobald der Ball das Brett berührt, dreht er sich um 90 Grad und prallt somit »gespiegelt« ab.

Das war schon die ganze Abfrage. Aber diese Abfrage steht jetzt noch im leeren Raum. Sie muss natürlich noch aufgerufen werden, wenn sie aktiv sein soll, und zwar immer wieder. Wir müssen sie also in eine fortlaufende Wiederholung stecken. Da wir beim Ball schon eine haben, fügen wir die Abfrage dort einfach mit ein.

4. Füge die Abfrage in die laufende Wiederholung des Balls ein.

Und ausprobieren! Großartig! Jetzt läuft die Spielmechanik. Das Brett ist nach links und rechts steuerbar, der Ball fliegt, er prallt an den Wänden ab und ebenso am Brett, wenn er es berührt.

Jetzt ist es wieder an der Zeit, das Spiel zu speichern.

Im nächsten Kapitel werden wir *Gefahren und Ziele* hinzufügen – genau das, was noch fehlt, um dieses Programm zu einem schwierigen, aber auch spannenden Breakout-Spiel zu machen!

Auf einen Blick

Inhalt