2. Auflage

Al Sweigart

Coole Spiele mit Scratch 3

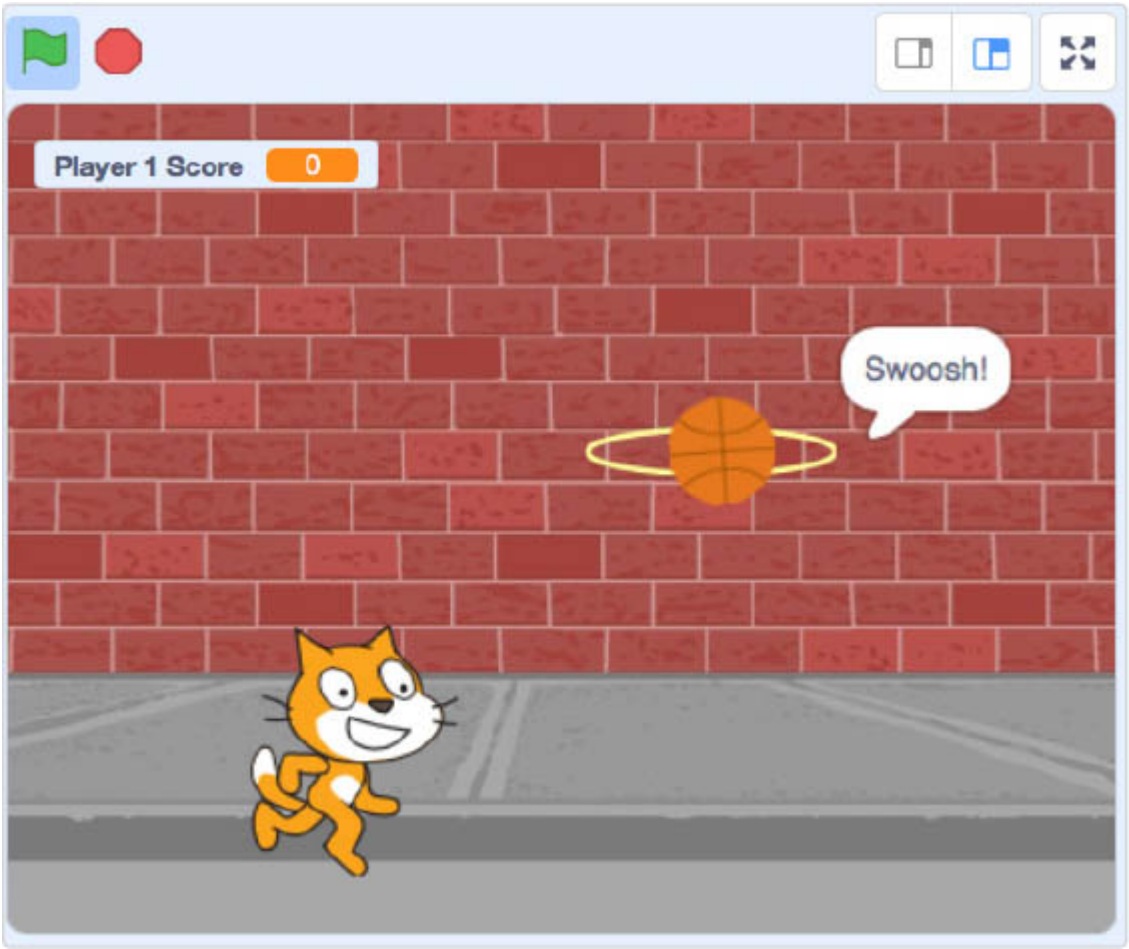
Lerne programmieren und baue deine eigenen Spiele

no starch press

dpunkt.verlag

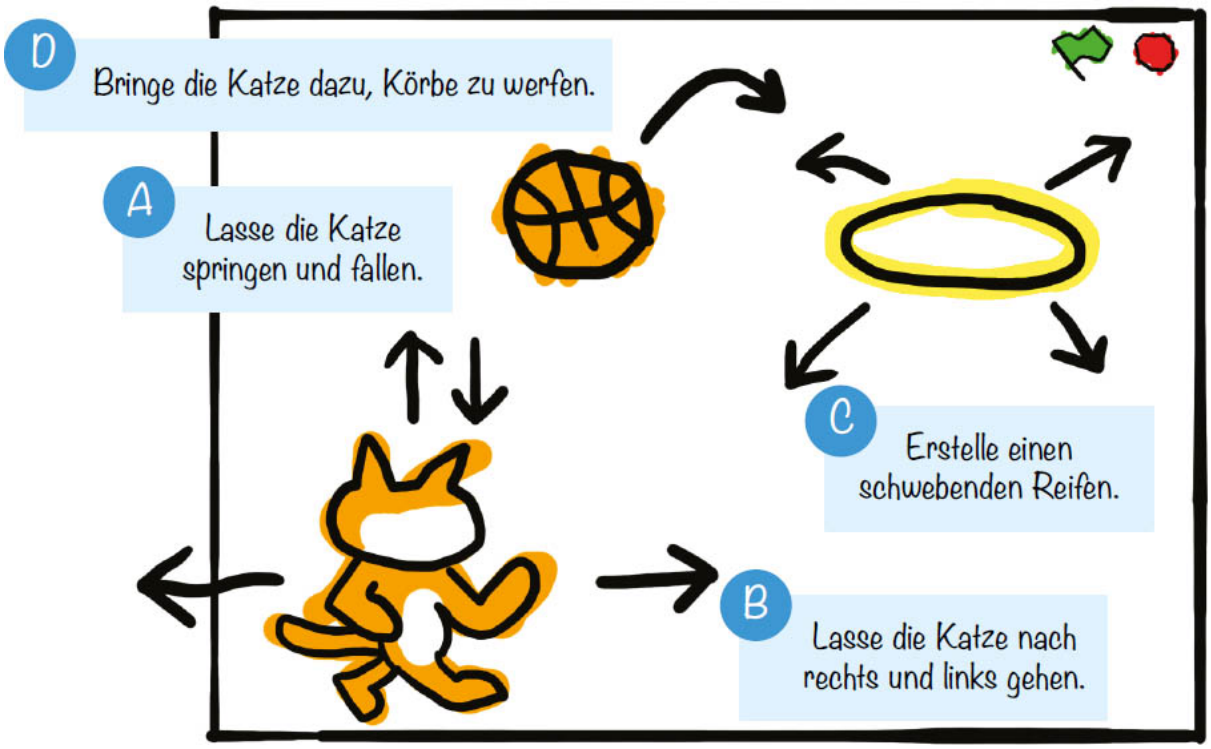
Inhalt

Bevor du mit dem Programmieren beginnst, solltest du dir das fertige Spiel auf *https://www.nostarch.com/scratch3playground/* anschauen.



Das Projekt skizzieren

Als Erstes wollen wir skizzieren, was in dem Spiel passieren soll. Der Spieler steuert die Katze, die sich nach rechts und links bewegen und springen kann. Das Ziel besteht darin, Bälle in einen Ring zu werfen, der sich zufällig auf der Bühne bewegt.



Um Zeit zu sparen, kannst du die Skelettdatei für das Projekt als Ausgangspunkt verwenden. Sie befindet sich als *basketball-skeleton.sb3* in der ZIP-Datei, die du von *https://www.nostarch.com/scratch3playground/* herunterladen kannst, indem du auf den Link *Download Scratch 3 Programming Playground's resources* rechtsklickst und **Ziel speichern als** auswählst. Entpacke anschließend die ZIP-Datei. In der Skelettdatei sind bereits alle Figuren geladen und tragen die im folgenden Abschnitt verwendeten Namen, sodass du nur noch die Codeblöcke zusammenstellen musst.

A Die Katze springen und fallen lassen

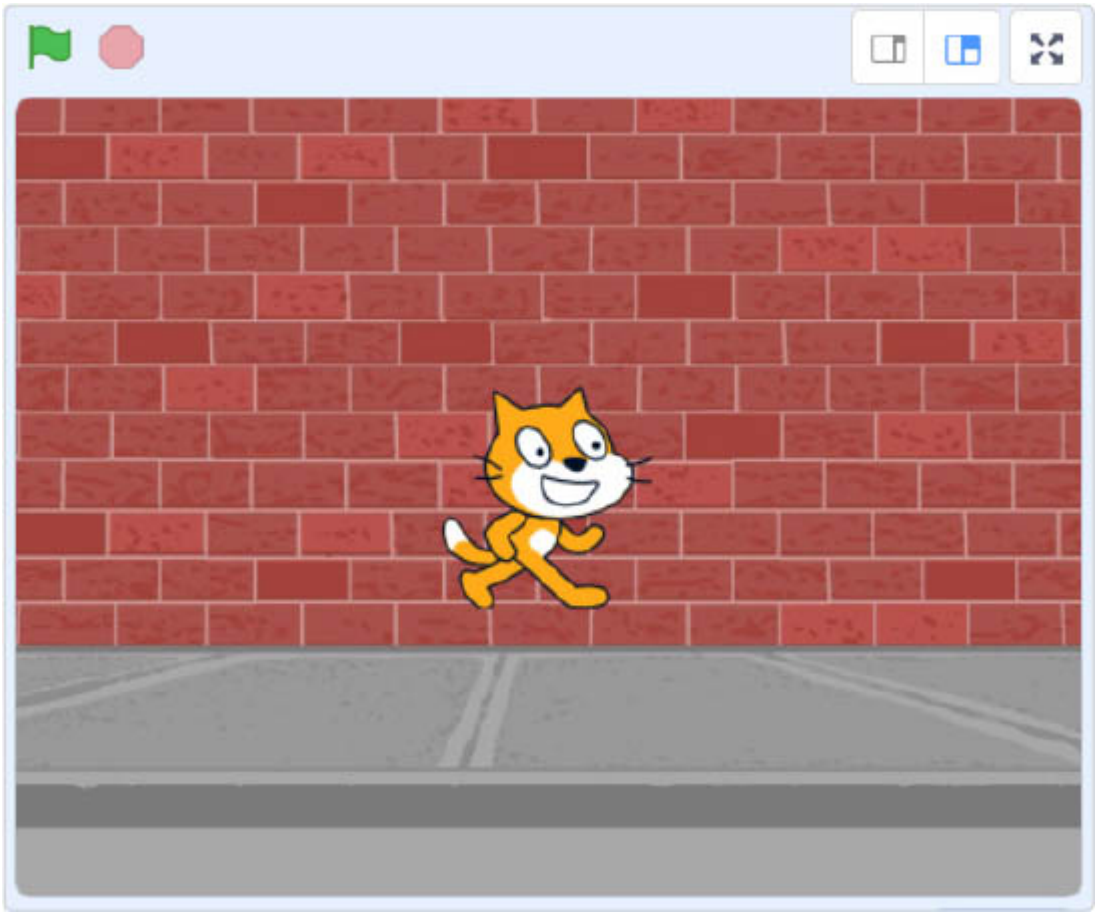
Als Erstes wollen wir für Schwerkraft sorgen, damit die Katze springen und herunterfallen kann.

1. Den Schwerkraftcode zur Katze hinzufügen

Benenne im Figuren-Eingabefeld *Figur1* in *Cat* um. Sofern du als Scratcher angemeldet bist, kannst du im Textfeld oben im Scratch-Editor anstelle von *Untitled* den neuen Namen *Basketball* für das Programm eingeben..

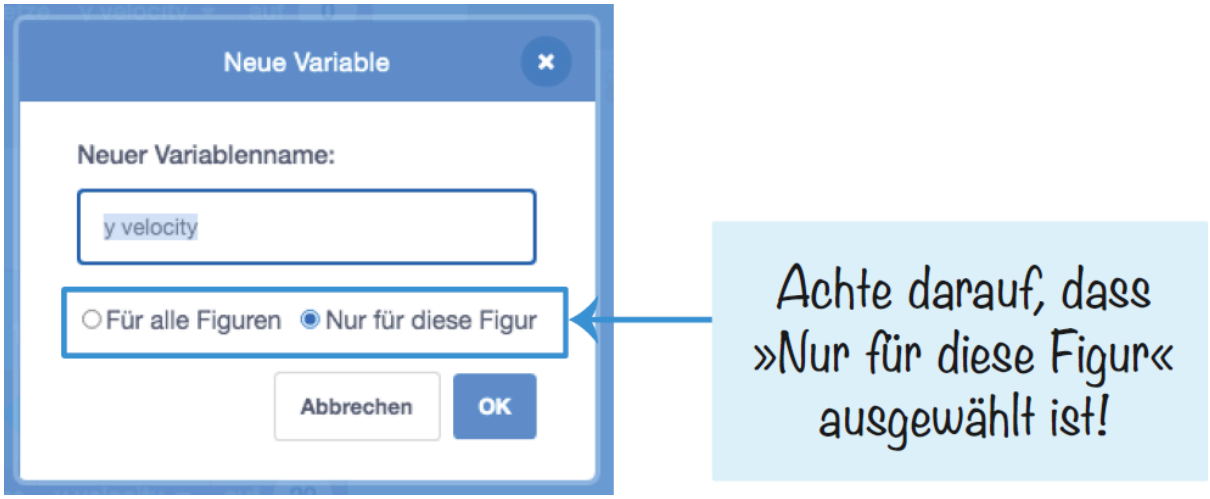
Klicke unter *Bühne* auf **Bühnenbild wählen**, um die Bühnenbild-Bibliothek zu öffnen. Wähle darin **Wall 1** aus und klicke auf **OK**. Die Bühne sieht jetzt wie

folgt aus:



Um in dem Programm für Schwerkraft zu sorgen, brauchst du *Variablen*. Eine Variable kannst du dir als einen Behälter zum Speichern von Zahlen oder Text vorstellen, die du später in dem Programm verwenden kannst. Für unser Spiel erstellen wir eine Variable mit einer Zahl, die angibt, wie schnell die Katze fällt.

Wähle dazu als Erstes die Figur *Cat* in der Liste aus und öffne die Registerkarte *Skripte*. Klicke dann in der orangefarbenen Gruppe *Variablen* auf die Schaltfläche **Neue Variable**. Dadurch wird das gleichnamige Fenster eingeblendet. Gib als Variablenname *y velocity* ein (»y-Geschwindigkeit«). Diese Variable besagt, wie schnell sich etwas nach oben oder unten bewegt. Ist *y velocity* eine positive Zahl, bewegt sich die Katze nach oben, ist es eine negative Zahl, sinkt sie nach unten. Vergewissere dich, dass **Nur für diese Figur** ausgewählt ist. (Wenn *Für alle Figuren* ausgewählt ist, hast du die Bühne markiert, nicht die Katze!) Klicke anschließend auf **OK**.



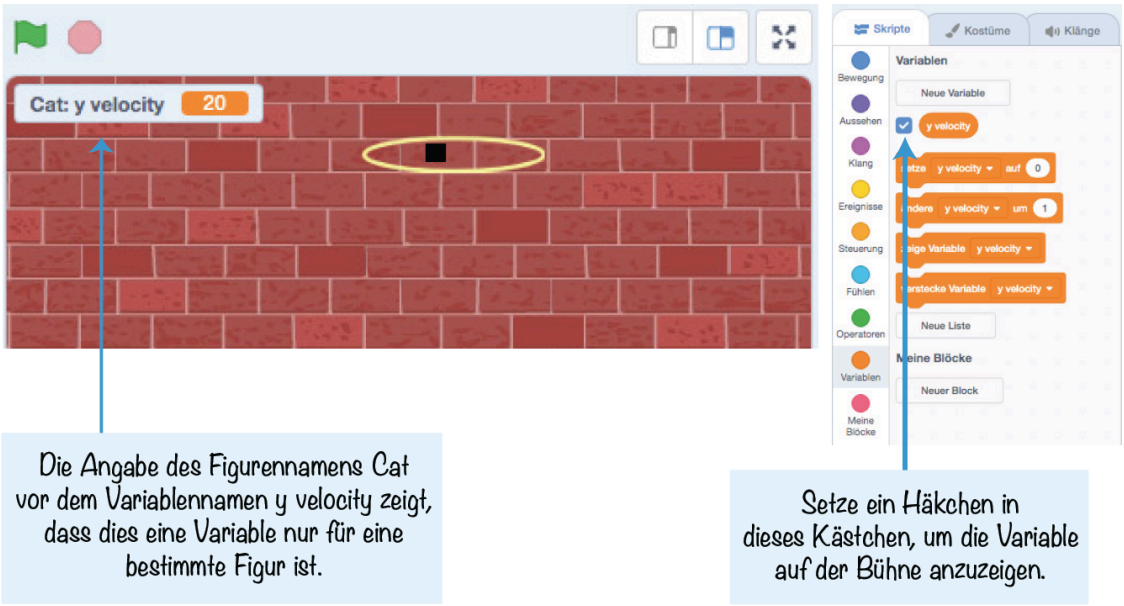
Jetzt erscheinen in der Gruppe *Variablen* mehrere neue Blöcke. Einer davon ist der abgerundete Variablenblock *y velocity*.

»Für alle Figuren« und »Nur für diese Figur«

Als du die Variable *y velocity* erstellt hast, musstest du **Nur für diese Figur** auswählen. Diese Einstellung sorgt dafür, dass die Variable, die du anlegst, nur von der Katze verwendet werden kann. Mit *Für alle Figuren* dagegen erstellst du Variablen, die alle Figuren nutzen können.

Um zu prüfen, was für eine Art von Variable du angelegt hast, setzt du ein Häkchen in das Kästchen neben dem abgerundeten Variablenblock in

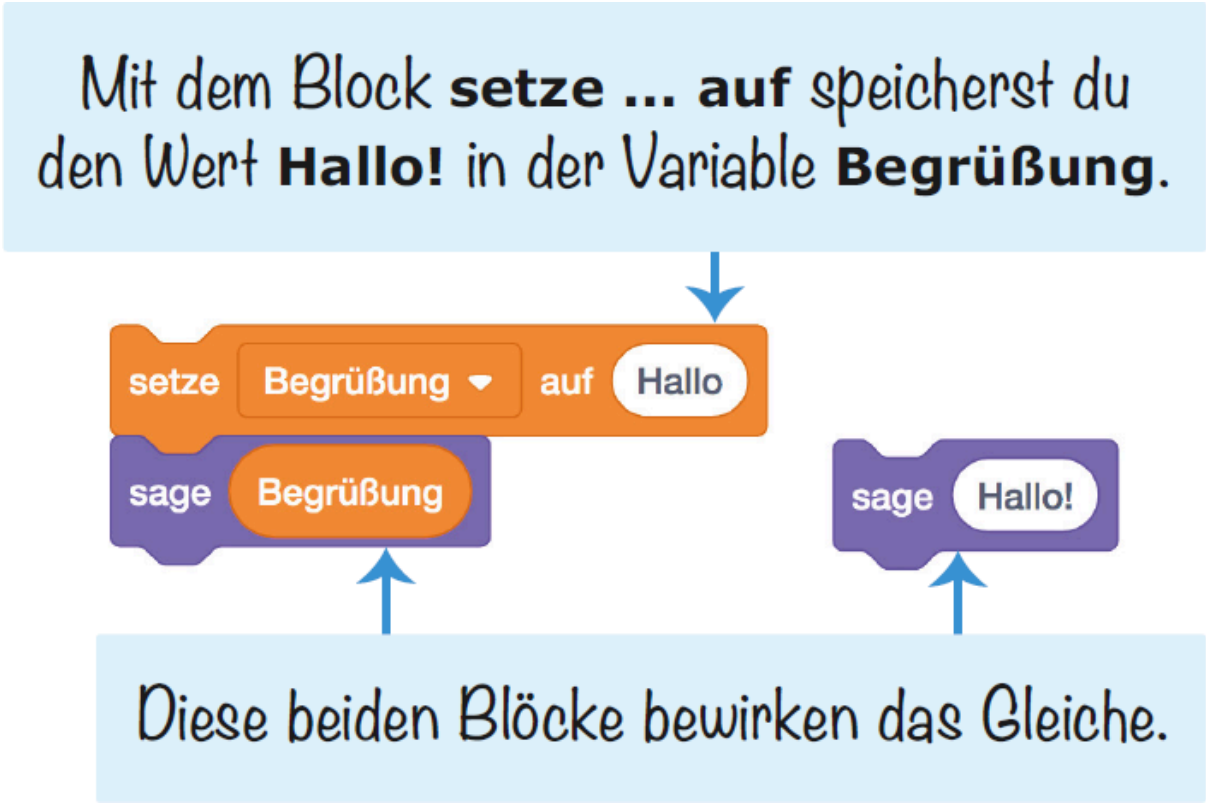
der Palette, sodass die Variable auf der Bühne angezeigt wird. Hast du *Nur für diese Figur* ausgewählt, erscheint vor dem Variablennamen der Name der Figur. Bei *Für alle Figuren* dagegen wird nur der Variablenname angegeben.



Sollte der Figurenname *Cat* nicht vor dem Variablennamen angezeigt werden, klickst du mit der rechten Maustaste in der orangefarbenen Gruppe *Variablen* auf den Block **y velocity** und wählst **Variable löschen**. Erstelle die Variable *y velocity* dann neu, aber achte diesmal darauf, dass **Nur für diese Figur** ausgewählt ist.

Du kannst den Block *y velocity* und auch jeden anderen Variablenblock überall dort platzieren, wo du eine Zahl oder Text eingeben kannst. Der Codeblock, in den du die Variable einfügst, verwendet dann die in der Variable gespeicherte Zahl bzw. den Text. Wenn du Variablen verwendest, kannst du die darin gespeicherten Zahlen oder Texte *im laufenden Programm* ändern.

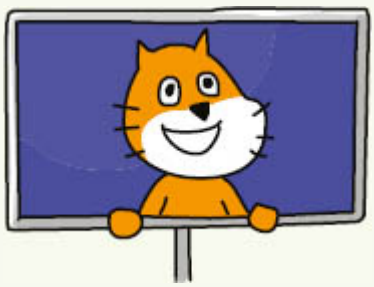
Um einen Wert in deiner Variable zu speichern, verwendest du den orangefarbenen Block **setze … auf**. Wenn du beispielsweise die Variable *Begrüßung* erstellt hast, kannst du mit **setze … auf** den Wert *Hallo!* darin aufnehmen. Dann kannst du *Begrüßung* in einen **sage**-Block einfügen, was das Gleiche bewirkt, als würdest du *Hallo!* direkt in diesen Block schreiben. (Füge diese Blöcke nicht zu dem Programm *Basketball* hinzu, und erstelle hier auch nicht die Variable *Begrüßung* – dies ist nur ein Beispiel!)



Um den Text der Begrüßung im Programmverlauf zu ändern, fügst du einen weiteren **setze ... auf**-Block hinzu. Bei Variablen, die Zahlen enthalten, kannst du auch den Block **ändere ... um** verwenden, um etwas von der Zahl abzuziehen oder zu ihr zu addieren.

Die Schwerkraft sorgt dafür, dass ein Objekt nach unten *beschleunigt*. Wenn die Katzenfigur im Spiel nach unten sinkt, muss sich diese Sinkgeschwindigkeit also im Programmverlauf ändern. Füge den folgenden Code zur Figur *Cat* hinzu, um in deinem Scratch-Programm die Schwerkraft wirken zu lassen. Das ist die Mindestmenge an Code, die du brauchst, damit eine Figur unter Schwerkrafteinfluss fällt. Du kannst ihn zu jeder Figur hinzufügen, die fallen muss.

Wenn du auf die grüne Fahne klickst, wird die Variable *y velocity* auf 0 gesetzt. Dann tritt das Skript in eine Endlosschleife ein. Darin wird die y-Position (die vertikale Position) der Katze um den Wert von *y velocity* geändert und *y velocity* um −2. Mit jedem Schleifendurchlauf ändert sich die y-Position schneller, sodass die Katze immer schneller fällt.



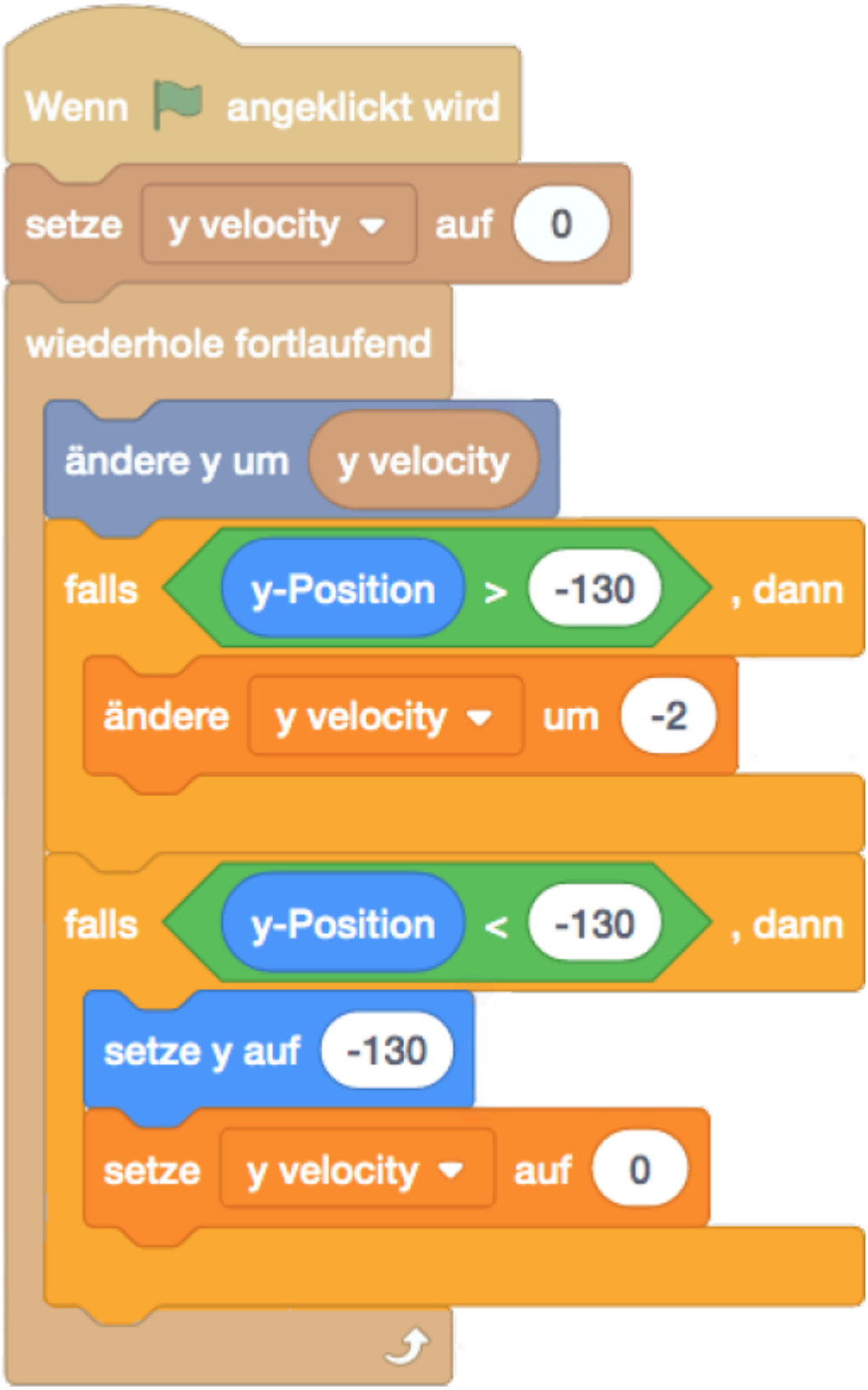
Zeit zu speichern!

Ziehe als Erstes die Katzenfigur zum oberen Bildschirmrand. Klicke dann auf die grüne Fahne und beobachte, wie die Katze fällt. Um sie abermals fallen zu lassen, musst du auf das rote Stoppschild klicken, die Katze

wieder zum oberen Bühnenrand ziehen und erneut auf die grüne Fahne klicken. Speichere dein Programm.

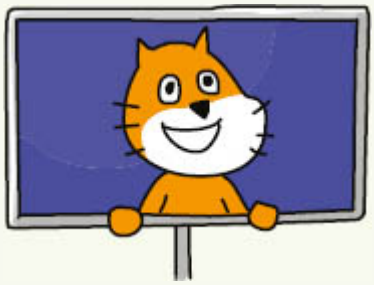
2. Code für die Landung auf dem Boden hinzufügen

Die Katze fällt jetzt zwar, aber wir wollen auch, dass sie damit aufhört, wenn sie den Boden erreicht. Dazu ergänzen wir den Code für die Figur *Cat* wie folgt:



In diesem Code verwenden wir als **y-Position** des Bodens den Wert −130. Ist die y-Position der Katze größer als −130 (befindet sie sich also über dem Boden), so wird *y velocity* um −2 geändert, und die Katze fällt weiter. Schließlich fällt die Katze aber unter die Bodenebene, sodass ihre y-Position kleiner als −130 ist.

Wenn das geschieht, wird die Figur auf die y-Position −130 zurückgesetzt und *y velocity* auf 0 gesetzt, damit die Katze nicht weiter fällt.



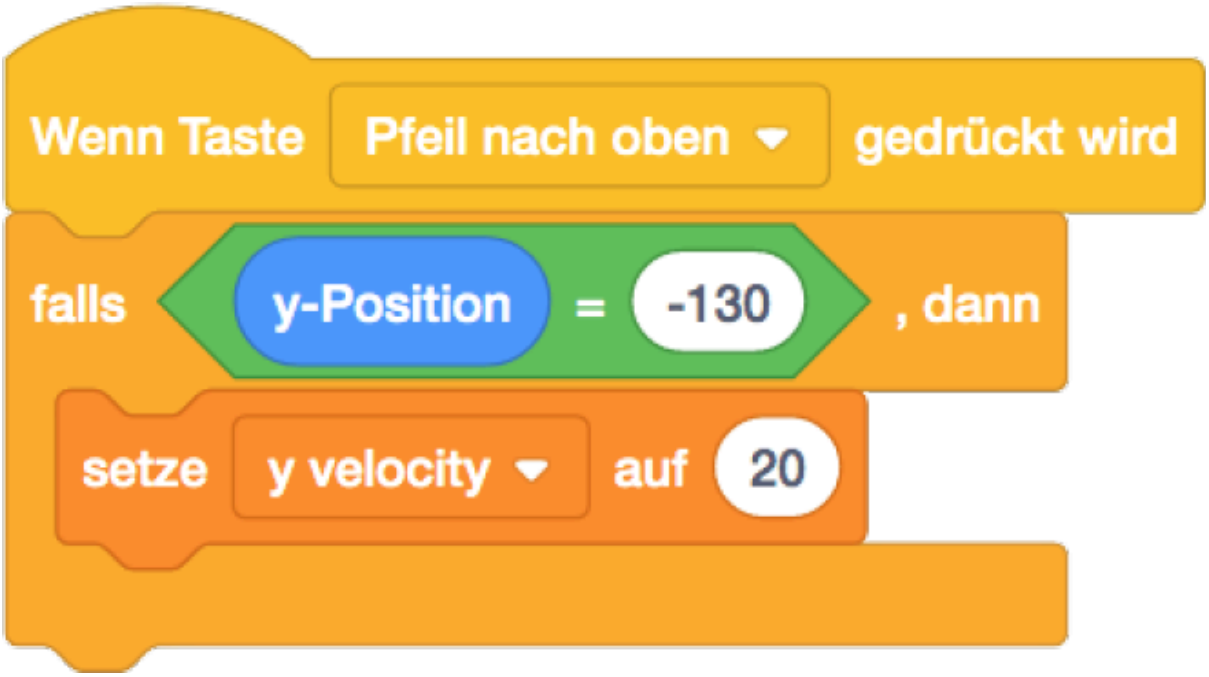
Zeit zu speichern!

Klicke auf die grüne Fahne, um den bisherigen Code zu testen. Ziehe die Katze mit der Maus nach oben und lass sie fallen. Vergewissere dich, dass sie bis zum Boden fällt, aber nicht über den unteren Bühnenrand hinaus sinkt. Du kannst auch verschiedene andere Bodenhöhen ausprobieren, indem du statt −130 eine andere Zahl verwendest. Klicke auf das rote Stoppschild und speichere dein Programm.

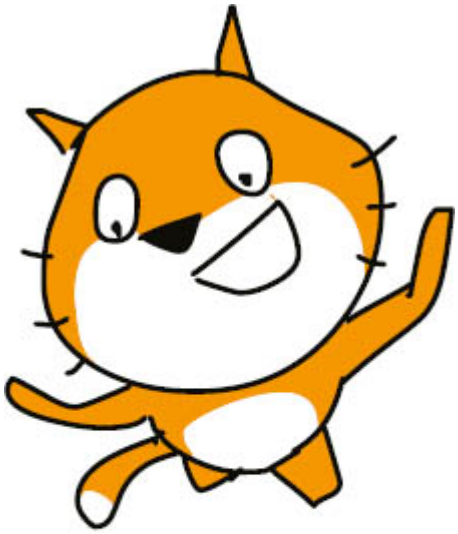
Wenn du mit dem Testen fertig bist, solltest du die Variable *y velocity* auf der Bühne unsichtbar machen, indem du in der orangefarbenen Gruppe *Variablen* das Häkchen aus dem Kästchen neben dem Variablenblock nimmst.

3. Den Springcode zur Katze hinzufügen

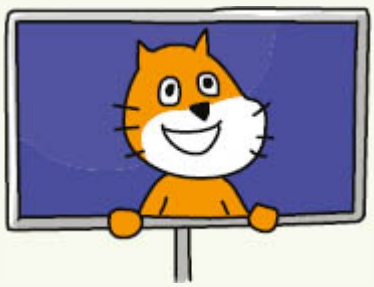
Da die Katze jetzt über Schwerkraftcode verfügt, ist es einfach, sie springen zu lassen. Füge der Figur *Cat* folgenden Code hinzu:



Wenn du jetzt die Taste mit dem nach oben weisenden Pfeil drückst, wird *y velocity* auf die positive Zahl 20 gesetzt, sodass die Figur nach oben springt. Allerdings wird die Variable *y velocity* immer noch bei jedem Durchlauf der Endlosschleife um 2 verringert. Die Katze steigt also zunächst 20 Einheiten in die Höhe, beim nächsten Durchlauf nur noch 18, dann 16 usw. Der Block **falls … dann** prüft, ob sich die Katze überhaupt am Boden befindet, denn es sollte nicht möglich sein, die Katze zum Springen zu veranlassen, wenn sie bereits in der Luft schwebt.



Wenn *y velocity* auf 0 gesetzt wird, hat die Katze den höchsten Punkt ihrer Flugbahn erreicht. Da die Variable bei jedem weiteren Durchlauf um −2 geändert wird, beginnt die Katze anschließend wieder zu fallen, bis sie den Boden berührt. Probiere verschiedene Zahlenwerte für die Blöcke **setze y velocity auf** und **ändere y velocity um** aus. Finde heraus, wie du die Katze höher oder weniger hoch springen lassen kannst (aber immer oberhalb des Bodens) oder wie du die Schwerkraft stärker und schwächer machst.

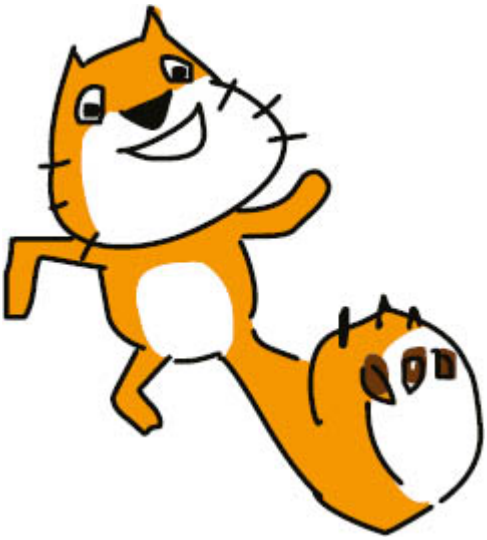


Zeit zu speichern!

Klicke auf die grüne Fahne, um den bisherigen Code zu testen. Drücke die Taste mit dem Pfeil nach oben und vergewissere dich, dass die Katze nach oben springt und wieder nach unten fällt. Klicke auf das rote Stoppschild und speichere dein Programm.

B Die Katze nach rechts und links gehen lassen

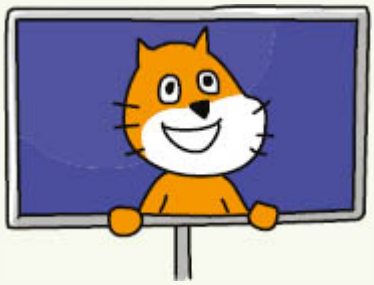
Als Nächstes fügen wir der Figur *Cat* Code hinzu, sodass der Spieler sie mithilfe der Tastatur nach rechts und links gehen lassen kann.



4. Gehcode zur Katze hinzufügen

Füge unterhalb des Codes für die Figur *Code* folgenden weiteren Code hinzu:

In der Schleife **wiederhole fortlaufend** prüft das Programm, ob die Tasten mit dem Pfeil nach links oder rechts gedrückt wurden. Wenn ja, wechselt die Katzenfigur zum nächsten Kostüm und ändert ihre x-Position um −10 (nach links) bzw. 10 (nach rechts). Die Figur *Cat* hat zwei Kostüme, die du dir anschauen kannst, indem du auf die Registerkarte *Kostüme* über der Blockpalette klickst. Dadurch, dass die Katze im Block **nächstes Kostüm** ständig das Kostüm wechselt, sieht es so aus, als würde sie gehen.



Zeit zu speichern!

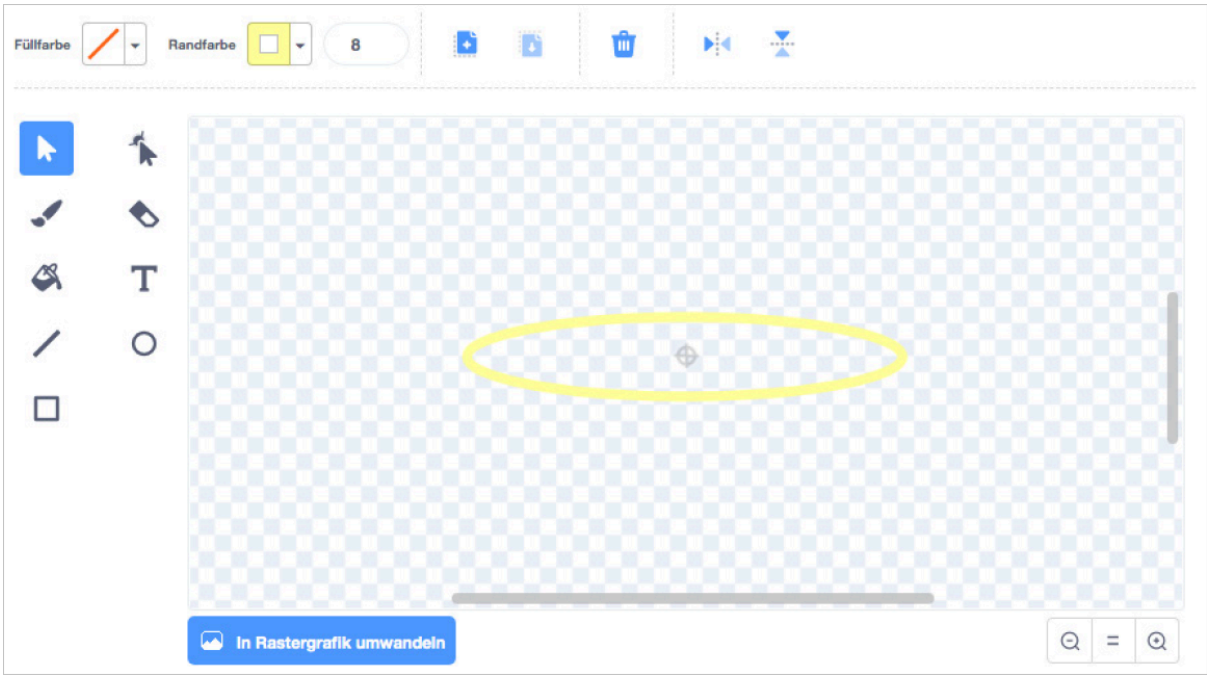
Klicke auf die grüne Fahne, um den bisherigen Code zu testen. Drücke die Tasten mit dem Pfeil nach links und rechts und vergewissere dich, dass die Katze in die richtige Richtung geht. Wenn die Katze sich nach links bewegt, sieht es so aus, als ginge sie rückwärts; das ist aber so in Ordnung. Klicke auf das rote Stoppschild und speichere dein Programm.

C Einen schwebenden Reifen erstellen

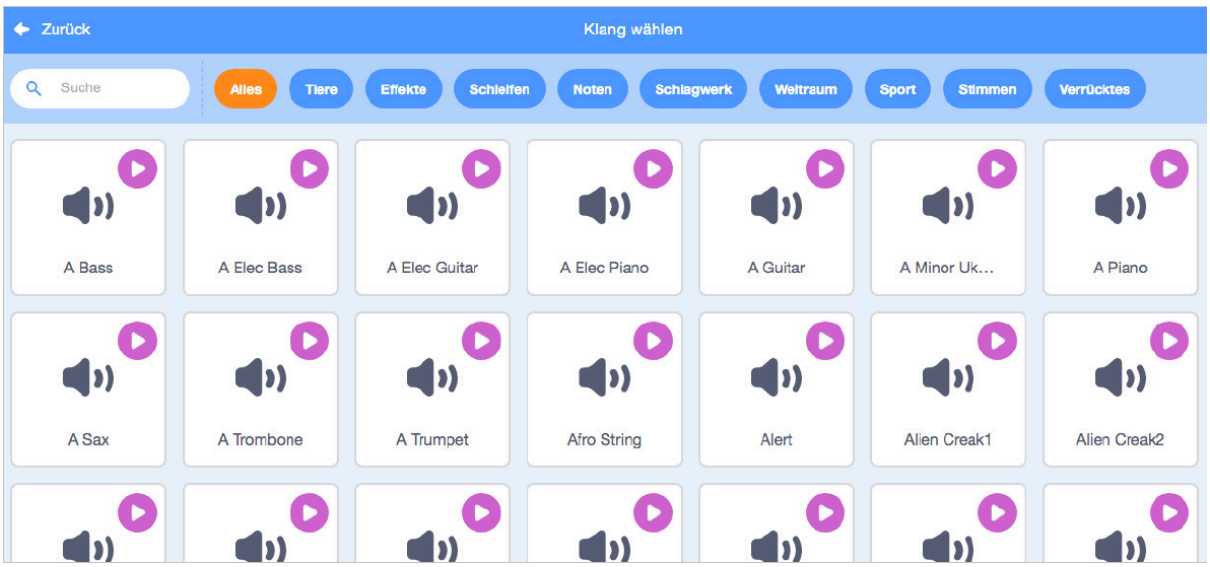
Nachdem wir nun die Figur *Cat* fertiggestellt haben, gehen wir zur nächsten Figur über, nämlich dem Reifen, durch den wir die Bälle werfen wollen.

5. Die Figur für den Reifen erstellen

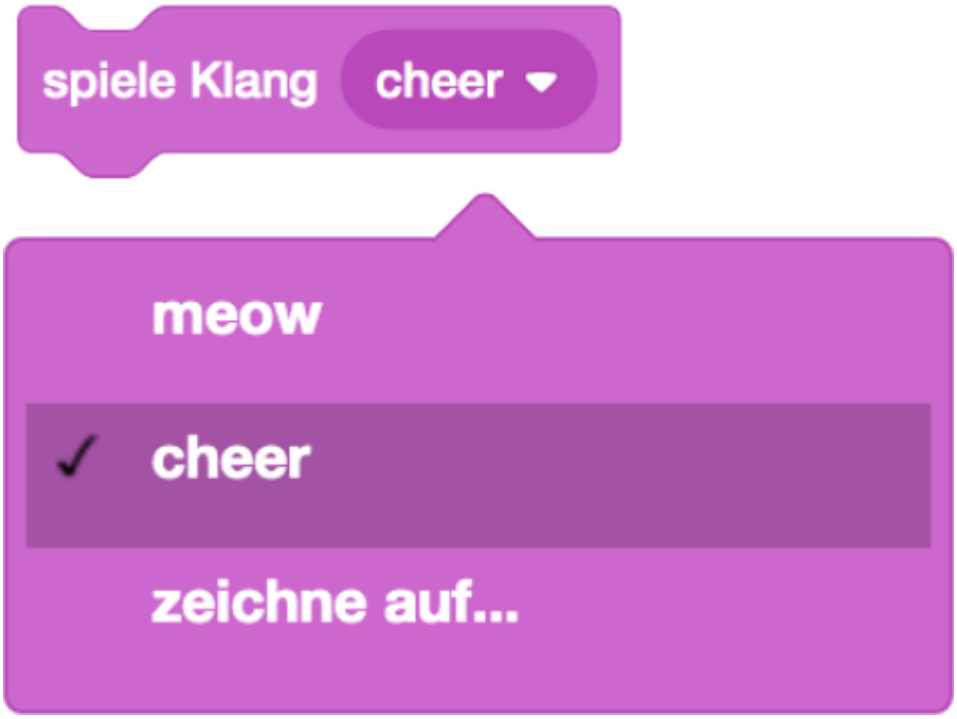
Klicke auf die Schaltfläche **Malen**, die erscheint, sobald du den Mauszeiger auf den Button **Figur wählen** bewegst oder darauf tippst. Die Schaltflächen der Zeichenwerkzeuge erscheinen auf der linken Seite des Zeichenbereichs. Wähle eine gelbe Farbe und verwende das Ellipse-Werkzeug, um einen Kreis zu zeichnen. Um die Linie des Kreises dicker zu machen, erhöhe die Zahl im Eingabefeld neben den Feldern Füll- und Randfarbe. Achte darauf, dass sich das Fadenkreuz des Zeichenbereichs im Mittelpunkt der Ellipse befindet.



Gib der Figur im Figuren-Eingabefenster den Namen *Hoop* (»Reifen«). Diese Figur soll einen Jubelklang abspielen, wenn der Spieler einen Korb wirft. Daher laden wir als Nächstes den Klang *cheer*. Klicke oberhalb der Blockpalette auf die Registerkarte **Klänge** und dann, über den Button mit dem Lautsprechersymbol *Klang wählen*, auf die gleichnamige Schaltfläche. In dem Auswahlfenster **Klang wählen** wählst du dann *cheer* aus und klickst auf **OK**.



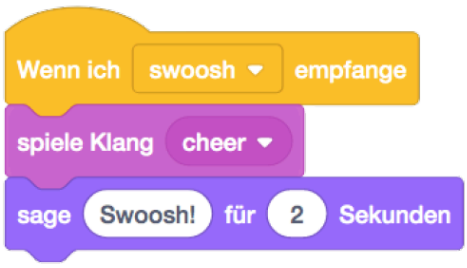
Der Klang *cheer* erscheint jetzt als Auswahlmöglichkeit im Block **spiele Klang**, den wir im Folgenden zu der Figur *Hoop* hinzufügen.



Füge *Hoop* den folgenden Code hinzu, damit die Figur zufällig über die obere Hälfte der Bühne gleitet und bei einem Treffer den Jubelklang abspielt. In dem zweiten Skript musst du dafür sorgen, dass eine Nachricht empfangen werden kann. Klicke dazu auf das schwarze Dreieck des Blocks **Wenn ich … empfange** und wähle **Neue Nachricht**. Gibt als Namen für die Nachricht *swoosh* ein.

1

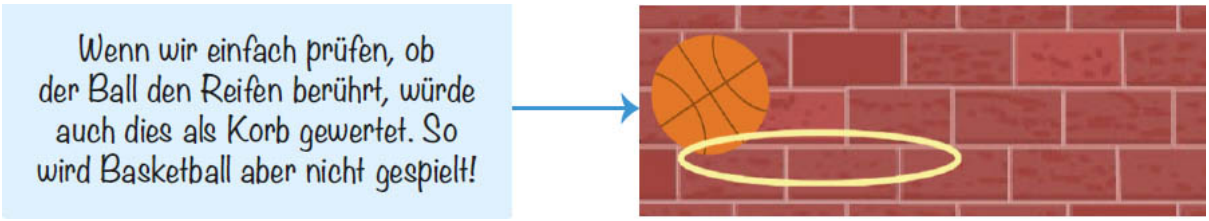
2



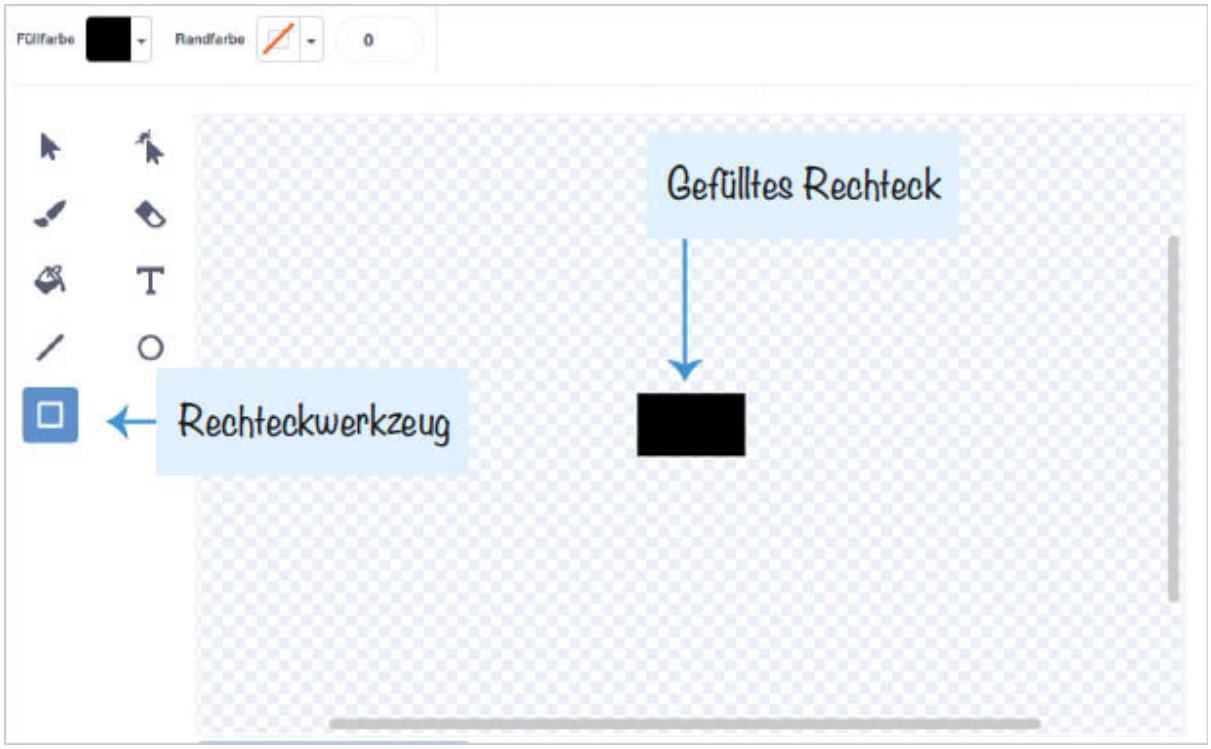
Skript 1 sorgt dafür, dass der Reifen in jeder Sekunde an eine neue Position schwebt. Ein beweglicher Reifen macht das Spiel viel kniffliger und spannender! Skript 2 spielt den Klang *cheer* ab und zeigt *»Swoosh!«* an, wenn es die Nachricht *swoosh* empfängt. Wir werden später dafür sorgen, dass die Figur für den Ball diese Nachricht sendet, wenn sie im Korb landet.

6. Die Hitbox-Figur erstellen

Als Nächstes müssen wir überlegen, wie der Code erkennen kann, ob der Spieler einen Korb geworfen hat. Wir könnten zwar ein Programm schreiben, das prüft, ob der Ball den Reifen *berührt*, aber dabei würde auch schon ein einfaches Anstoßen an den Ring als Korb zählen. In Wirklichkeit wollen wir aber nur dann einen Treffer anerkennen, wenn der Ball durch die Mitte des Reifens geht. Aber wie machen wir das?



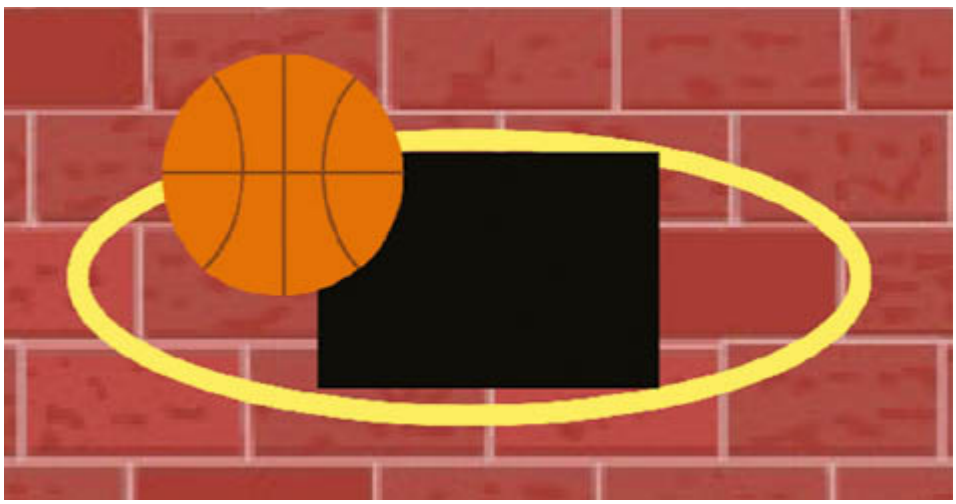
Eine mögliche Lösung besteht darin, eine *Hitbox* zu verwenden. Das ist ein Begriff aus der Spieleentwicklung für einen rechteckigen Bereich, der herangezogen wird, um zu bestimmen, ob zwei Spielobjekte miteinander zusammengestoßen sind. Zeichnen wir also eine Hitbox-Figur! Klicke auf die Schaltfläche **Malen**, die erscheint, sobald du auf den Button **Figur wählen** gehst. Male mit dem Rechteckwerkzeug und der Auswahl für ein gefülltes Rechteck ein kleines schwarzes Quadrat auf das Fadenkreuz. Nenne die Figur *Hitbox*. Sie sieht wie folgt aus:



Füge der Figur *Hitbox* den folgenden Code hinzu:

Dadurch folgt die Hitbox dem Reifen, wohin auch immer dieser sich bewegt.

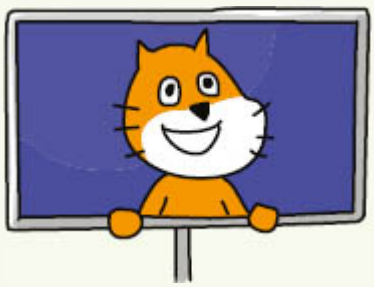
Das Skript, das wir in Schritt 9 schreiben werden, sorgt dafür, dass nur dann ein Treffer gewertet wird, wenn der Ball die Figur *Hitbox* trifft, nicht *Hoop*. Der Ball muss also viel näher an die Mitte des Reifens herankommen, damit ein Korbwurf gewertet wird.



Es würde allerdings ziemlich merkwürdig aussehen, wenn wir das schwarze Viereck in der Mitte des Reifens beließen. Daher wollen wir die Hitbox unsichtbar machen. Füge den Block **setze Effekt Durchsichtigkeit auf**zu der Figur *Hitbox* hinzu und gib als Wert 100 an (in der Palette erscheint dieser Block zunächst als

setze Effekt Farbe auf, anstatt »Farbe« kann dann »Durchsichtigkeit« eingestellt werden).

Es gibt einen Unterschied zwischen den Blöcken **verstecke dich** und **setze Durchsichtigkeit-Effekt auf**. Würdest du die Hitbox mit **verstecke dich** unsichtbar machen, dann könnten die *wird … berührt*-Blöcke, die wir später verwenden, niemals eine Berührung zwischen dem Ball und der Hitbox erkennen und der Spieler könnte keinen Treffer erzielen. Dagegen macht **setze Durchsichtkeit-Effekt auf** die Figur zwar ebenfalls unsichtbar, lässt es aber weiterhin zu, das die *wird … berührt*-Blöcke ihre Anwesenheit wahrnehmen.



Zeit zu speichern!

Klicke auf die grüne Fahne, um den bisherigen Code zu testen. Vergewissere dich, dass der Reifen über die Bühne wandert und dass sich

die Hitbox immer in seiner Mitte befindet. Klicke auf das rote Stoppschild und speichere dein Programm.

D Die Katze Körbe werfen lassen

Als Nächstes fügen wir einen Ball hinzu, den die Katze werfen kann. Ebenso wie die Katze braucht auch der Ball Schwerkraftcode, damit er zu Boden fallen kann.

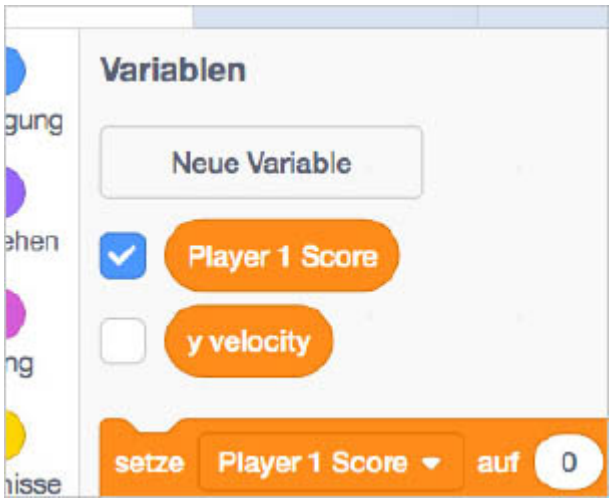
7. Die Figur für den Ball erstellen

Klicke im Bereich unten rechts auf die Schaltfläche **Figur wählen**, um die Figurenbibliothek zu öffnen. Wähle dort **Basketball** aus.

Anschließend klickst du oberhalb der Blockpalette auf die Registerkarte **Klänge** und dort im Bereich links unten auf **Klang wählen**, um die Klangbibliothek zu öffnen. Wähle den Klang *pop* aus. Öffne dann wieder den Skriptbereich, indem du über der Blockpalette auf **Skripte** klickst.

Als Nächstes erstellst du in der orangefarbenen Gruppe *Variablen* zwei neue Variablen. Klicke auf die Schaltfläche **Neue Variable** und nenne die Variable *y velocity*. Vergewissere dich, dass **Nur für diese Figur** ausgewählt ist, bevor du auf **OK** klickst. Da die Variablen jeweils nur für die vorliegende Figur gelten, handelt es sich bei *y velocity* für die Katze und für den Ball um zwei verschiedene Variablen, auch wenn sie denselben Namen haben.

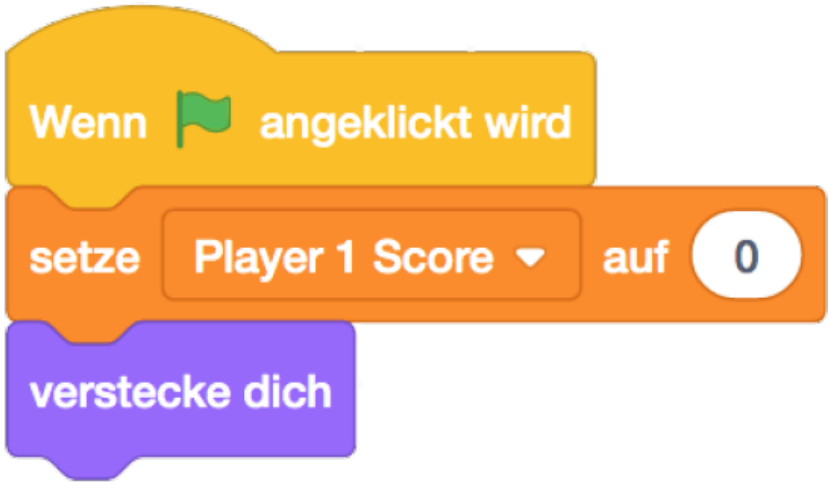
Klicke erneut auf **Neue Variable**, um eine weitere Variable namens *Player 1 Score* (»Punktestand Spieler 1«) zu erstellen. Diesmal aber wählen wir die Einstellung **Für alle Figuren**. (Wir schreiben *Player 1 Score* groß, da wir die Variable auf der Bühne anzeigen wollen.) Nimm das Häkchen aus dem Kästchen neben *y velocity*, um diese Variable auf der Bühne zu verbergen. Jetzt erscheinen die neuen Variablenblöcke in der Gruppe *Variablen*.



8. Den Code für den Ball hinzufügen

Mit dem Klang *pop* und den beiden Variablen kannst du nun den folgenden Code zur Figur *Basketball* hinzufügen:

1


Wenn angeklickt wird
setze Player 1 Score auf 0
verstecke dich

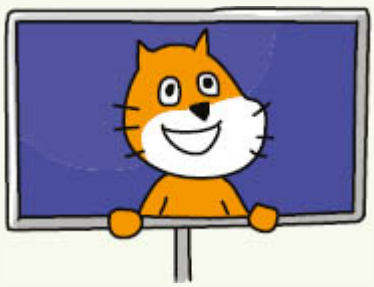
2


Wenn Taste Leertaste gedrückt wird
spiele Klang pop
gehe zu Cat
setze y velocity auf 24
zeige dich
wiederhole bis y-Position < -130
ändere x um 8
ändere y um 10
ändere y velocity um -2
drehe dich um 6 Grad
verstecke dich

Skript 1 sorgt dafür, dass der Spieler zu Anfang einen Punktestand von 0 hat, und versteckt den Basketball.

Der Code in Skript 2 ähnelt dem für die Katze. Wenn der Spieler die Leertaste drückt, erscheint der Ball vor der Katze und beginnt sich weiter nach vorn und nach oben zu bewegen. Ebenso wie die Variable *y velocity* der Katze bei einem Sprung wird hier die Variable *y velocity* des Balls auf eine positive Zahl gesetzt, damit es so aussieht, als ob die Katze den Ball nach oben wirft.

Der Block **wiederhole bis y-position < −130** sorgt dafür, dass der Ball fällt, bis er den Boden berührt. Danach wird der Ball wieder versteckt, bis der Spieler erneut die Leertaste drückt.



Zeit zu speichern!

Klicke auf die grüne Fahne, um den bisherigen Code zu testen. Drücke die Leertaste, damit die Katze den Ball wirft. Vergewissere dich, dass der Ball wieder verschwindet, wenn er den Boden berührt. Klicke auf das rote Stoppschild und speichere dein Programm.

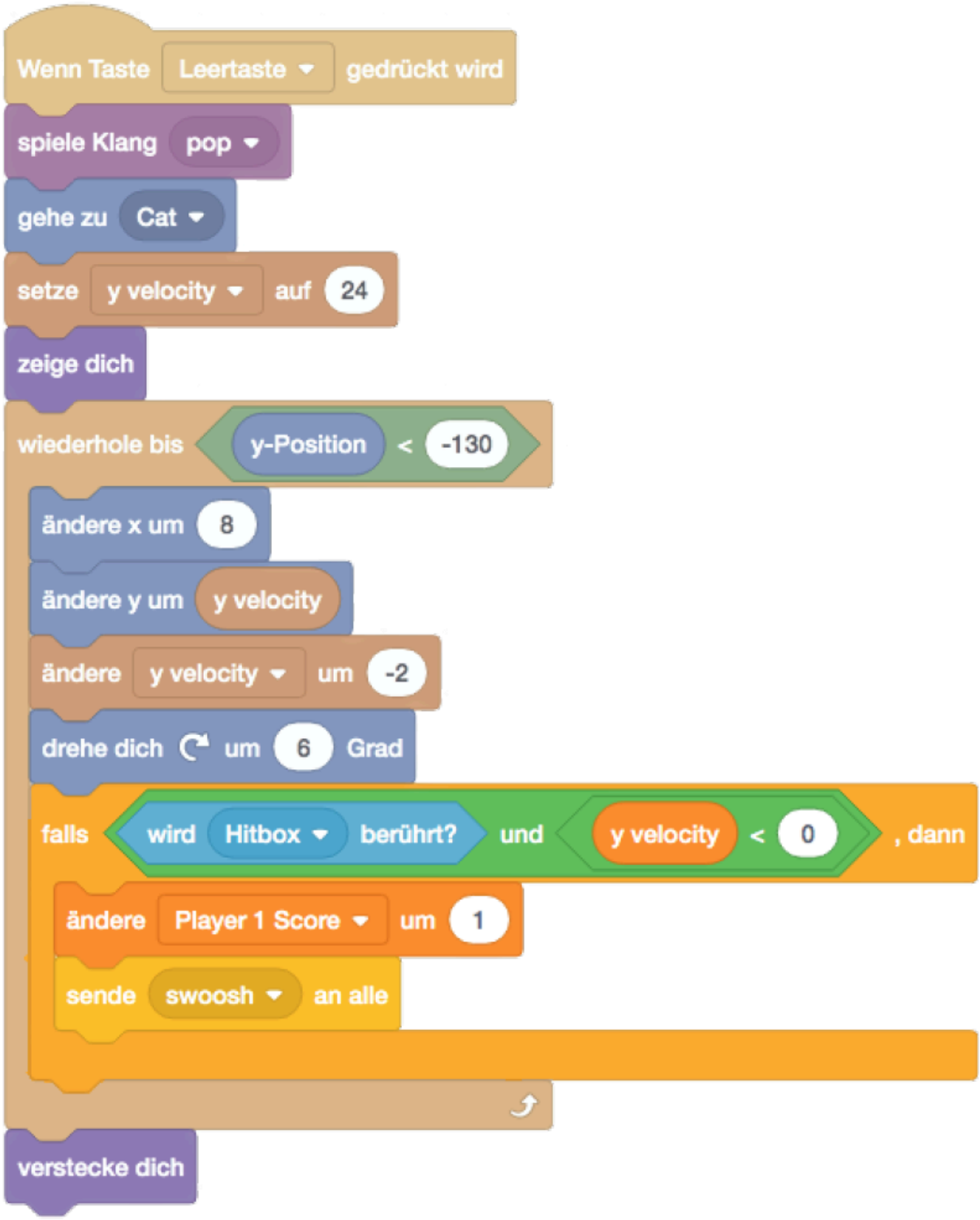
9. Einen Treffer erkennen

Als Nächstes fügst du den Code hinzu, um zu prüfen, ob der Ball die Hitbox getroffen hat. Dadurch kannst du erkennen, ob der Spieler einen Korb geworfen hat. Wenn ja, erhöhst du den Wert der Variable *Player 1 Score*. Es gibt jedoch einen Haken an der Sache: Es wird kein Korb gezählt, wenn der Ball den Reifen *von unten* durchdringt.

Denke daran, dass der Block **ändere y um y velocity** den Ball nach oben bewegt, wenn die Variable *y velocity* positiv ist. Ist *y velocity* gleich 0, bewegt sich der Ball gar nicht, und hat die Variable einen negativen Wert, fällt der Ball nach unten.

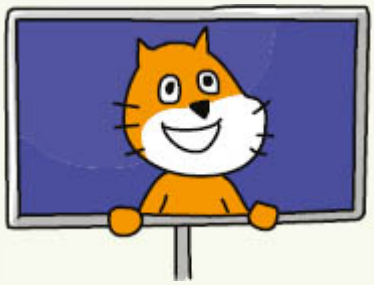
Daher musst du dem Code für die Figur *Basketball* einen weiteren **falls … dann**-Vergleich hinzufügen. Der Punktestand soll nur dann erhöht werden, wenn

der Ball die Hitbox trifft (**wird Hitbox berührt?**) und nach unten fällt (**y velocity < 0**).



Der **und**-Block kombiniert zwei Bedingungen. Damit Scratch die Codeblöcke innerhalb des **falls … dann**-Blocks ausführt, müssen *beide* Bedingungen erfüllt sein. Es reicht nicht, dass die Figur die Hitbox berührt oder dass *y velocity* kleiner 0 ist. Sowohl **wird Hitbox berührt?** als auch **y velocity < 0** müssen wahr sein.

Nur wenn das der Fall ist, wird die Variable *Player 1 Score* um 1 erhöht und die Nachricht *swoosh* gesendet.



Zeit zu speichern!

Klicke auf die grüne Fahne, um den bisherigen Code zu testen. Wirf einige Bälle. Die Variable *Player 1 Score* sollte nur dann erhöht werden, wenn der Ball die Reifenmitte berührt und sich dabei abwärts bewegt. Außerdem sollte dann die Sprachblase »Swoosh!« angezeigt und der Jubelklang abgespielt werden. Klicke auf das rote Stoppschild und speichere dein Programm.

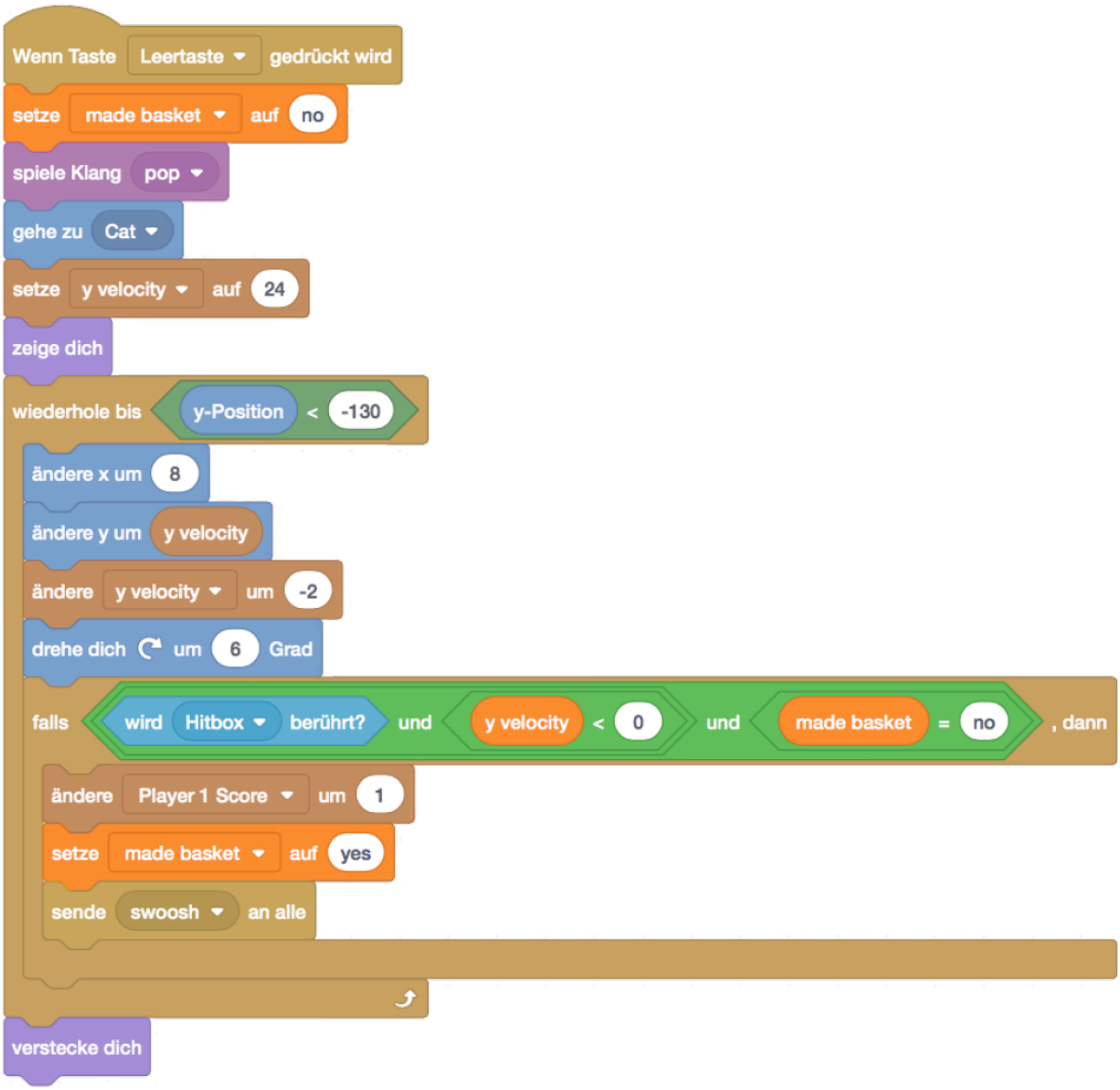
10. Den Fehler bei der Punktwertung beheben

Ist dir aufgefallen, dass *Player 1 Score* bei einem einzigen Korb um mehrere Punkte erhöht wird? Das ist ein *Bug*, also ein Problem, das dafür sorgt, dass sich das Programm auf unerwartete Weise verhält. Um herauszufinden, warum das passiert, müssen wir uns den Code noch einmal genau anschauen.

Die Schleife **wiederhole bis** wird ständig durchlaufen, bis der Ball den Grund berührt. Der gesamte Code darin gilt also für einen einzigen Wurf. Dabei prüft der Block aber mehrmals, ob der Ball die Hitbox berührt und dabei nach unten fällt. *Player 1 Score* sollte aber nur beim ersten Eintreten beider Bedingungen erhöht werden.

Um diesen Bug zu beheben, erstellst du eine neue Variable, die sich merkt, ob der Ball während eines Wurfs bereits einmal den Reifen berührt hat. Dadurch kannst du sicherstellen, dass der Spieler immer nur einen Punkt pro Wurf erzielen kann.

Klicke in der Blockpalette auf die orangefarbene Gruppe *Variablen* und dann auf **Neue Variable**. Nenne die Variable *made basket* (»Korb geworfen«) und wähle **Nur für diese Figur**. Ändere dann den Code für den Ball wie im Folgenden gezeigt.

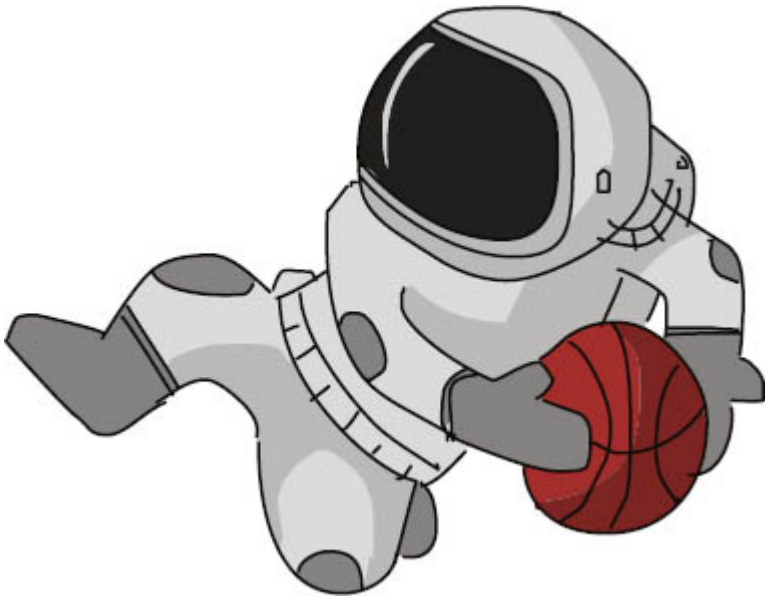


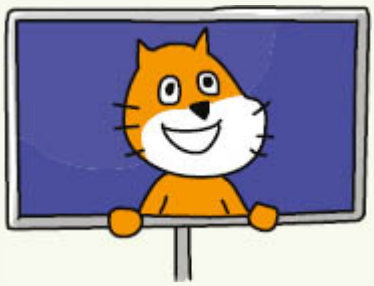
Wenn der Spieler die Leertaste drückt, wird die Variable *made basket* auf *no* gesetzt. Das ist sinnvoll, denn schließlich kann noch gar kein Korb geworfen worden sein, wenn der Spieler gerade erst wirft. Später verwenden wir einen **und**-Block, um eine weitere Bindung hinzuzufügen, die der Code überprüfen muss, um festzustellen, ob ein Korb erzielt wurde. Jetzt wird ein Korb nur dann erkannt und der Code in dem **falls … dann**-Block nur dann ausgeführt, wenn die folgenden drei Bedingungen alle wahr sind:

1. Die Figur *Basketball* berührt die Figur *Hitbox*.
2. Die Variable *y velocity* ist negativ (der Basketball bewegt sich also abwärts).
3. Die Variable *made basket* hat den Wert *no*.

Wenn der Code zum ersten Mal feststellt, dass ein Korb erzielt wurde, erhöht er *Player 1 Score* um 1 und setzt *made basket* auf *yes*. Bei allen weiteren Prüfungen

während dieses Wurfs ist *made basket* nicht mehr gleich *no*, weshalb kein weiterer Korb mehr erkannt wird. Wenn der Spieler wieder die Leertaste drückt, um den Ball erneut zu werfen, wird *made basket* auf *no* zurückgesetzt.





Zeit zu speichern!

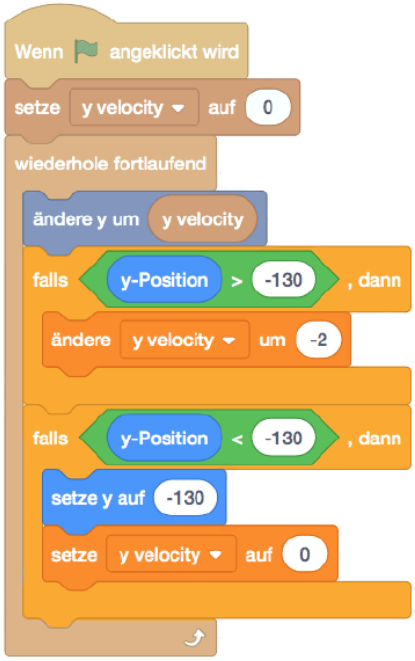
Klicke auf die grüne Fahne, um den bisherigen Code zu testen. Wirf einige Bälle und vergewissere dich, dass *Player 1 Score* für jeden Korb immer nur um 1 erhöht wird. Klicke auf das rote Stoppschild und speichere dein Programm.

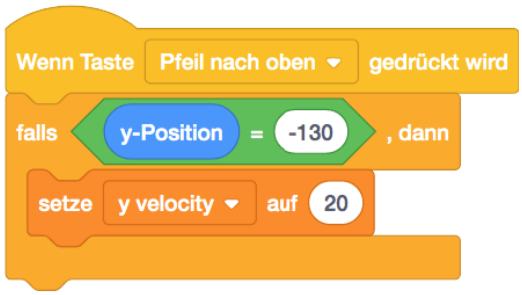
Das vollständige Programm

Die folgende Abbildung zeigt den fertigen Code. Wenn dein Programm nicht richtig funktioniert, vergleiche deinen Code mit dem in der Abbildung.

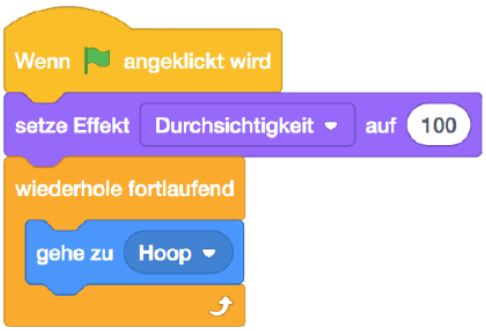
Cat


Wenn angeklickt wird
wiederhole fortlaufend
falls Taste Pfeil nach links gedrückt? , dann
wechsle zum nächsten Kostüm
ändere x um -10
falls Taste Pfeil nach rechts gedrückt? , dann
wechsle zum nächsten Kostüm
ändere x um 10


Wenn angeklickt wird
setze y velocity auf 0
wiederhole fortlaufend
ändere y um y velocity
falls y-Position > -130 , dann
ändere y velocity um -2
falls y-Position < -130 , dann
setze y auf -130
setze y velocity auf 0


Wenn Taste Pfeil nach oben gedrückt wird
falls y-Position = -130 , dann
setze y velocity auf 20

Hitbox


Wenn angeklickt wird
setze Effekt Durchsichtigkeit auf 100
wiederhole fortlaufend
gehe zu Hoop

Index

Symbole

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

R

S

T

U

V

W

X

Y

Z