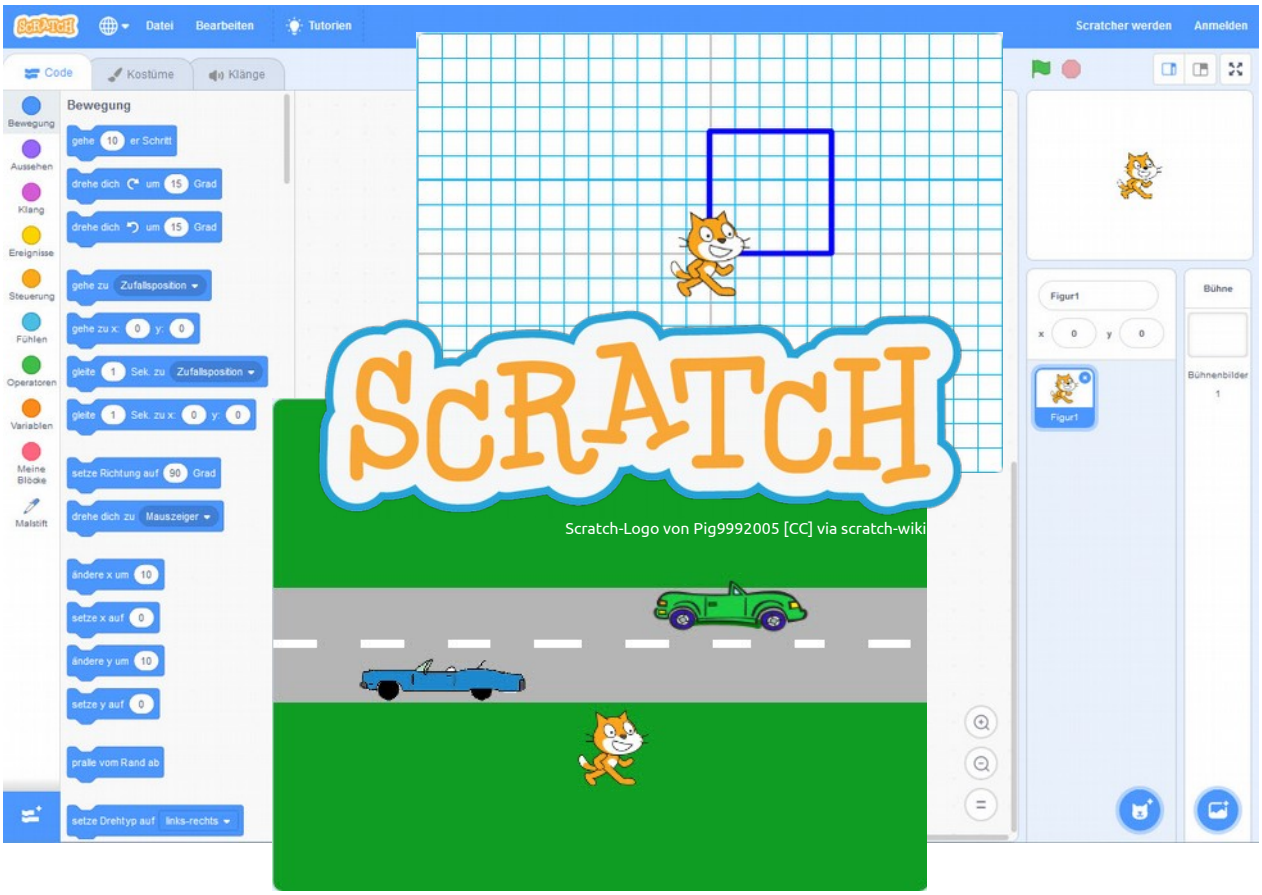
EINFÜHRUNG IN DIE PROGRAMMIERUNG MIT SCRATCH



Scratch ist eine Programmiersprache und Online-Gemeinschaft, in der man seine eigenen interaktiven Geschichten, Spiele und Animationen erstellen und seine Werke mit anderen überall auf der Welt teilen kann. Beim Entwickeln und Programmieren von Scratch-Projekten lernen junge Menschen kreativ zu denken, systematisch vorzugehen und kooperativ mit anderen zusammenzuarbeiten.

Scratch ist ein Projekt der Lifelong-Kindergarten-Gruppe am MIT-Media-Lab. Es ist kostenlos unter *http://scratch.mit.edu* verfügbar.

Monika Eisenmann – E-Mail: eisenmann.schule@email.de. – Januar 2017
Michael Sedding – E-Mail: sedding@csg-tuebingen.de - Januar 2019
Tobias Küchel – E-Mail: kuechel@humboldt-ka.de - November 2019

Inhaltsverzeichnis

Elternbrief

Liebe Eltern,

im Informatikunterricht arbeiten wir in den nächsten Wochen mit Scratch. Scratch ist eine Programmierumgebung, mit der die Schülerinnen und Schüler, ohne sich an komplizierte Syntax halten zu müssen, die Grundstrukturen der Programmierung kennen lernen und einüben.

Es wäre schön, wenn Sie Ihrem Kind zu Hause die Möglichkeit geben könnten regelmäßig mit Scratch zu arbeiten und zu experimentieren.

Sie müssen dafür nichts installieren. Scratch 3.0 läuft im Browser und kann mit dem Computer aber auch mit einem Tablets benutzt werden.

Sie finden den Entwicklungsumgebung unter: *https://scratch.mit.edu/*

Vielen Dank für Ihre Unterstützung!

Mit freundlichen Grüßen

Laufzettel von _____. Ich kann …

… ein Scratchprojekt abspeichern.	Teil 1, Seite 6	
… den Namen einer Figur ändern.	Teil 1, Seite 6	
… die Begriffe Anweisung und Sequenz erklären.	Teil 2, Seite 7	
… beschreiben, was die einzelnen Bewegungsanweisungen bewirken	Teil 2, Seite 7	
… die Katze mit dem Stift zeichnen lassen.	Teil 3, Seite 8	
… die Katze ein Rechteck, ein Quadrat und ein gleichseitiges Dreieck zeichnen lassen.	Teil 3, Seite 8	
… erklären, warum es sinnvoll ist, eine Startsequenz zu haben.	Teil 3, Seite 8	
… die Katze mit Hilfe einer Schleife zeichnen lassen.	Teil 4, Seite 9	
… erklären, was eine Schleife ist.	Teil 4, Seite 9	
… Tastaturereignisse programmieren, so dass die Katze über die Bühne gesteuert werden kann.	Teil 5, Seite 10	
… für eine Figur ein neues Kostüm erstellen.	Teil 5, Seite 10	
… erklären, was eine Verzweigung ist.	Teil 6, Seite 11	
… ein Bühnenbild und eine neue Figur erstellen und dessen Namen ändern.	Teil 7, Seite 12	
… das Auto mit einer Schleife so lange fahren lassen, bis es am Rand angekommen ist.	Teil 7, Seite 12	
… das Auto mit Schleife und Verzweigung ununterbrochen von links nach rechts fahren lassen	Teil 8, Seite 13	
… erklären, was eine Variable ist, sie deklarieren und initialisieren	Teil 9, Seite 14	
… Tastaturereignisse so programmieren, dass der Wert einer Variable erhöht und verringert wird.	Teil 9, Seite 14	
… mit Hilfe einer Variable das Auto mit unterschiedlicher Geschwindigkeit fahren lassen.	Teil 9, Seite 14	
… Bedingungen mit „und“ verknüpfen.	Teil 10, Seite 15	

Teil 0: Vorbereitungen

Die Scratch-Entwicklungsumgebung gibt es in einer online- und in einer offline-Variante. Um die online-Variante zu verwenden, musst Du kein Programm auf deinem PC installieren. Stattdessen rufst Du die Scratch-Internetseite in einem Browser auf. Dort läuft dann das Programm. In der Schule ist die offline-Variante installiert.

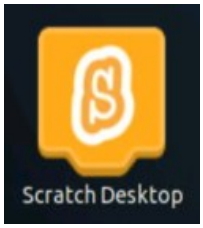
Mach dich arbeitsbereit: In der Schule

Bevor es losgeht, lege bitte einen Arbeitsordner in deinem Home-Verzeichnis an, in dem Du deine Programme / Zwischenstände abspeichern kannst. Starte dazu den Dateimanager und wechsle ins Verzeichnis `~/Home_auf_Server/Informatik/` und erstelle dort einen neuen Ordner mit dem Namen `Scratch`

Scratch starten

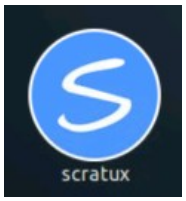
Starte die Offline-Variante. Falls das nicht möglich ist, öffne in einem Browser (z.B. Firefox) die Internetadresse: *https://scratch.mit.edu/* und klicke auf „Entwickeln“.

Ein Wort zum Abspeichern und Öffnen deiner Programme



Weil die Entwicklungsumgebung von Scratch in der Online-Variante „im Internet“ läuft, haben die Menüpunkte zum Speichern und Öffnen etwas eigenwillige Namen. Das gilt auch für die Offline-Variante.

Ein Programm *speicherst* Du *ab*, indem Du im Menü Datei den Menüpunkt: „Auf deinem Computer speichern“ klickst.

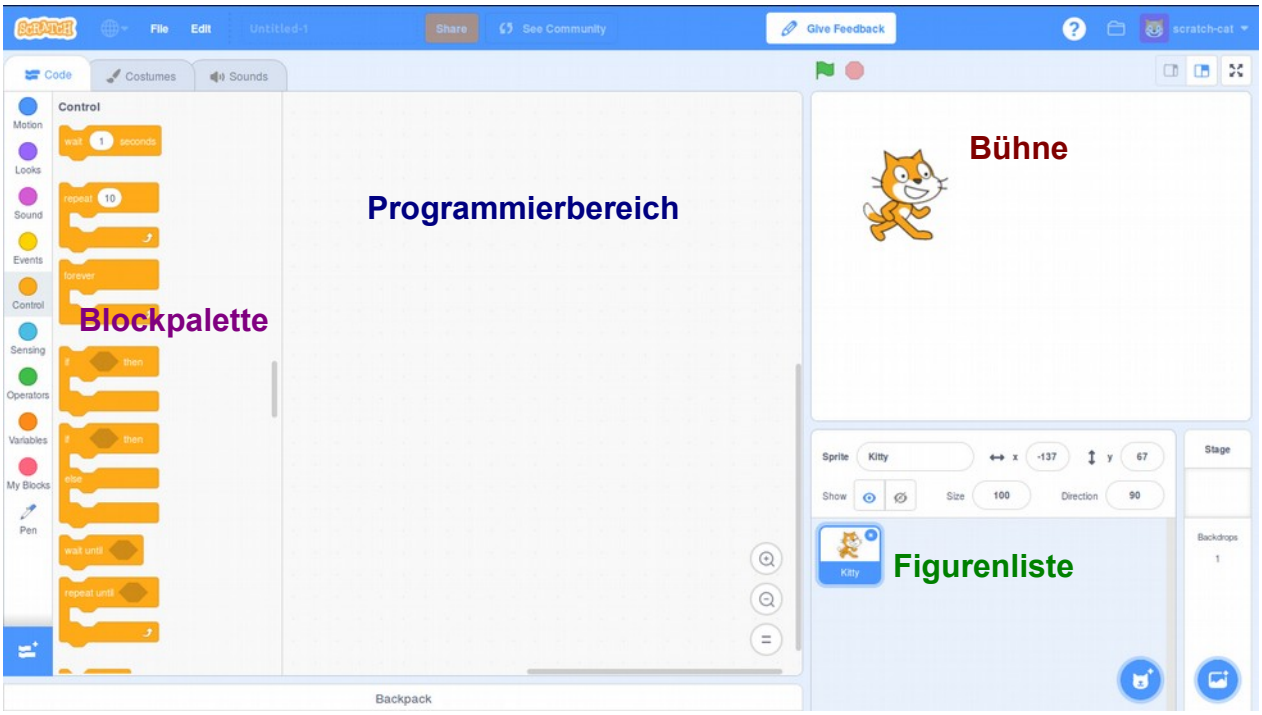


Ein gespeichertes Programm *öffnest* Du, indem Du im Menü Datei den Menüpunkt: „Hochladen von deinem Computer“ klickst. Diesen Schritt musst du zumindest zu Beginn jeder Unterrichtsstunde einmal machen.

Denke daran, Zwischenstände immer wieder unter anderem Namen abzuspeichern. So kannst Du leicht auf vorhergehende Entwicklungszustände zurückspringen.

Möchtest Du zu Hause weiterprogrammieren, lege dir die gespeicherte Datei in die Cloud unter Home_in_Cloud.

Teil 1: Erste Begegnung mit Scratch



Die **Scratch-Oberfläche** teilt sich in vier Bereiche auf.

Auf der **Bühne** läuft alles ab, was du programmierst. Dort machen deine Figuren das, was du ihnen im **Programmierbereich** aufträgst.

In der **Figurenliste** findest du alle Figuren, die in deinem Projekt dabei sind. Anfangs ist es nur eine Figur. Du kannst dort auch neue Figuren erzeugen und bestimmte Eigenschaften festlegen. Außerdem findest du dort auf der linken Seite auch die Bühne(n).

In der **Blockpalette** findest du die Blöcke, die du zum Programmieren brauchst. Du kannst sie entweder dort direkt durch einen Mausklick testen, oder mit der Maus in den **Programmierbereich** ziehen und an ein Programm anfügen.

Aufträge:

1. *Speichere dein Projekt erst einmal ab. Dazu wählst du Datei – Auf deinem Computer speichern , wählst den Ordner aus, den du für deine Programme angelegt hast und gibst den Dateinamen `01_katze` ein. Die Dateiendung .sb3 wird automatisch ergänzt.*
2. *Gib deiner Katze einen Namen. Das kannst Du im Einstellungsbereich oberhalb der Figurenliste tun. Schreibe statt „Figur 1“ den Namen deiner Katze, z.B. Kitty.. Hier kannst Du auch die Größe der Katze verändern.*

Teil 2: Die Katze lernt laufen – Anweisungen und Sequenzen

Lass die Figuren sich bewegen! Wie sie sich bewegen soll, müssen wir ihr sagen.

In der Palette findest du eine ganze Menge Blöcke, die für die Bewegung der Katze zuständig sind. Du kannst sie durch einen Mausklick direkt ausprobieren. Ziehe sie in den Programmierbereich, dort kannst du sie aneinander hängen und durch Anklicken werden die Anweisungen ausgeführt.

Löschen

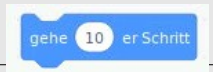
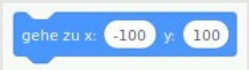
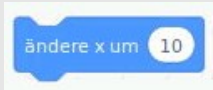
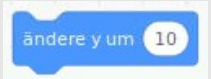
Du kannst auch Blöcke wieder löschen. Dazu ziehst du den Block einfach wieder zurück in die Blockpalette, oder du wählst nach einem Rechtsklick auf den entsprechenden Block „Lösche Block“ aus.

Sequenz

Jeder Auftrag, den du deiner Katze gibst, heißt **Anweisung** (z.B. „gehe 10er-Schritt“). Mehrere Anweisungen nacheinander nennt man **Sequenz**.

Aufträge:

1. *Teste folgende Anweisungen oder Sequenzen und schreibe auf, was passiert.*
2. *Probiere auch alle Blöcke in der Tabelle, nachdem du jeweils vorher die Anweisung „drehe dich um 30 Grad“ eingefügt hast. Bei welchen ändert sich das Ergebnis?*
3. *Schreibe in der Spalte 'Wirkung' auf, was passiert.*

Anweisung / Sequenz	**Wirkung**
Zusatzfrage: negativer  Was passiert bei Zahl?	
 Zusatzfrage: Wie heißen die Koordinaten der Ecken der Bühne? Tipp: Lade den Hintergrund „xy-Grid“.	
	
	
 Zusatzauftrag: Test alle vier Richtungen.	

4. *Bevor du weitermachst, speichere deine Datei unter `02katze_bewegen` ab. („Datei“ – „Auf deinem Computer speichern“)*

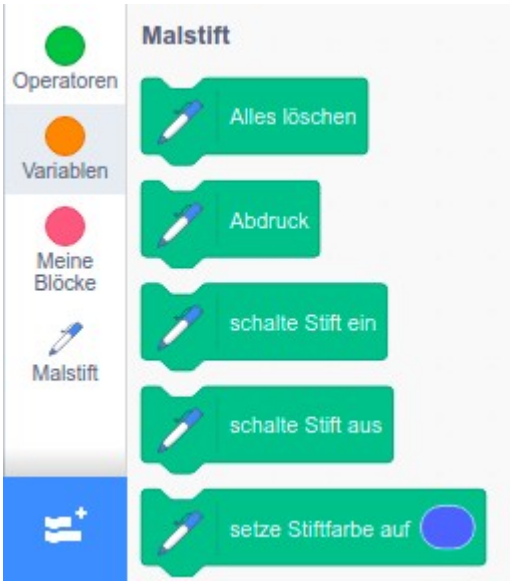
Teil 3: Die Katze kann zeichnen – Startsequenzen

Startblock

Wir können das, was wir im Programmierbereich zusammensetzen, auch mit der grünen Fahne links oberhalb der Bühne starten und mit dem Stoppschild wieder beenden. Den Startblock „Wenn Fahne angeklickt wird“ dazu findest du unter „Ereignisse“.

Spuren zeichnen

Um besser sehen zu können, was die Katze gemacht hat, können wir ihre Spuren zeichnen lassen. Die nötigen Werkzeuge müssen wir erst aktivieren. Klicke dazu auf das blaue **„Erweiterung hinzufügen“-Icon** links unten (vgl. Abbildung) und wähle die Erweiterung „Malstift“. Es erscheinen nun weitere Programmierbausteine in einer eigenen Kategorie.



Füge nun jeweils zuerst den Block in deinem Programm ein, der die Anweisung gibt, alle Malspuren wegwischen zu lassen, bevor du etwas Neues probierst, den Block zum Einschalten (Absenken) des Stiftes setzt du vor deine Anweisungen für die Bewegungen, den Block zum Ausschalten (Anheben) danach.

Größe ändern

Außerdem machen wir unsere Katze etwas kleiner, dass sie nicht alle Spuren verdeckt. Wie das geht, weißt Du schon von Seite 6. Das wollen wir aber natürlich nicht jedes mal machen müssen, wenn wir das Programm benutzen – damit das automatisch am Beginn geschieht, füge am Anfang deiner Sequenz einen Baustein „setze Größe auf __%“ ein.

Es kann viele Voraussetzungen für dein Programm geben, wie „Größe festlegen“ und „Malspuren wegwischen“. Damit du diese nicht jedes mal von Hand eingeben musst, macht es Sinn, hier eine **Startsequenz** zu programmieren.

Aufträge:

*Schreibe **jeweils** ein kleines Programm mit Startsequenz, welches...*

1. *… zuerst die Katze ins linke obere Eck der Bühne setzt und sie von dort aus dann ein Rechteck um die Bühne zeichnen lässt.*
2. *… zuerst die Katze ins linke untere Eck der Bühne setzt und sie dann eine Treppe mit unterschiedlich großen Stufen nach oben steigen lässt.*
3. *... die Katze ein Quadrat zeichnen lässt.*
4. *… die Katze ein gleichseitiges Dreieck zeichnen lässt.*
5. *Speichere dein Projekt unter `03katze_zeichnen` ab.*

Teil 4: Die Katze zeichnet – Schleifen/Wiederholungen

Im letzten Kapitel hast du ein Quadrat und ein gleichseitiges Dreieck gezeichnet. Deine Lösung sieht dabei so oder so ähnlich aus (linkes Bild):

Abbildung 2: Programmierung mit vielen gleichen Anweisungen

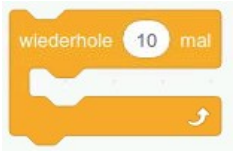
Hier tauchen dieselben Bausteine wiederholt auf – das geht auch schneller mit einer „**Schleife**“.Der Programmcode für das Quadrat wird dann gleich viel kürzer (rechtes Bild).

Abbildung 1: Programmierung mit Schleifen

Schleifen oder **Wiederholungen** sind ein ganz zentrales Konzept in der Programmierung – und es gibt sie in jeder Programmiersprache. Sie erleichtern uns vieles – und machen den Code übersichtlicher. Eine gute Orientierung ist:

> Wenn du überlegst, Blöcke zu kopieren, oder dir auffällt, dass du etwas doppelt machst, dann schaue zunächst, ob du dasselbe nicht durch den Einsatz einer Schleife lösen kannst.
>
> Wenn die Beschreibung deines Programmteils mit „**Solange**…“ beginnt, mit „**bis**...“ endet oder „**___ mal**“ enthält, ist das ein ziemlich sicherer Hinweis darauf, dass du hier eine **Schleife** einsetzen solltest.

Es gibt verschiedene Arten von Schleifen, die du sicher in den weiteren Aufgaben nutzen kannst:

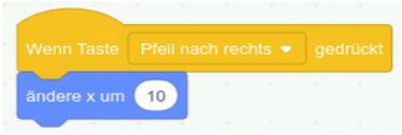
Zählschleife	Endlosschleife	Bedingte Schleife
		
Hier kannst du angeben, **wie oft** die Anweisungen in der Schleife wiederholt werden.	Diese Schleife wiederholt **endlos** die Anweisungen.	Diese Schleife hat ein sogenanntes **Abbruchkriterium**.

Aufgaben

1. *Ändere deine Programme für Dreieck und Quadrat so, dass sie mithilfe von Schleifen funktionieren – speichere unter `04katze_schleife` ab.*
2. *Schreibe ein Programm, das ein Sechseck zeichnet.*
3. *Programmiere auch die Treppe mithilfe einer Schleife. Schreibe beide Programmversionen in dein Heft – einmal mit, einmal ohne Schleife.*
4. *(*) Zeichne das Haus vom Nikolaus. Du sollst damit links unten beginnen und rechts unten aufhören. Erweitere Dein Programm dann so, dass es drei Häuser nebeneinander zeichnet – und verwende nur einen einzigen zusätzlichen Baustein.*

Teil 5: Ich bewege die Katze – Tastensteuerung

Öffne dein Projekt `01_katze` oder starte ein neues Projekt und speichere es unter `05_katze` wieder ab.



Tastendruckereignisse - Pfeiltasten

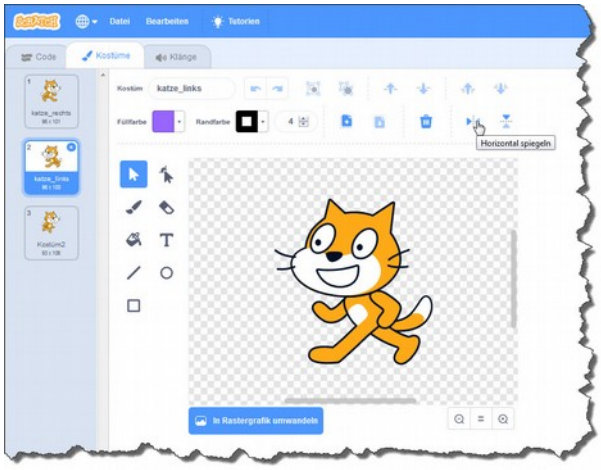
Unser Ziel ist es jetzt, die Katze mit den Pfeiltasten der Tastatur steuern zu können. Du hast schon das erste **Ereignis** kennen gelernt: „Wenn Fahne angeklickt“. Direkt darunter findest du das Ereignis „Wenn Taste … gedrückt“. Welche Taste hier abgefragt werden soll, kannst du einstellen. Das nennt man in der Informatik „**Parameter**“. Ziehe den Block in den Programmierbereich und wähle als Parameter „Pfeil nach rechts“ aus. Wie weit die Katze bei jedem Tastendruck gehen soll, kannst du selbst bestimmen.

Aufgaben

1. *Programmiere das Ereignis „Wenn Fahne angeklickt“ so, dass eine um 50% verkleinerte Katze dabei an eine Startposition ganz unten auf der Bühne in der Mitte gesetzt wird.*
2. *Programmiere jetzt die Tastaturereignisse für die restlichen Pfeiltasten und probiere alles aus, indem du die Katze ein bisschen über die Bühne spazieren lässt.*

Du kannst einzelne Blöcke oder auch ganze Blocksequenzen vervielfältigen und dann verändern. Klicke dazu mit der rechten Maustaste auf den ersten zu kopierenden Block und wähle „**Duplizieren**“ aus. **Aber**: **Kopieren** solltest du nur dann, wenn du **Ähnliches** wiederverwenden willst. Bei **Gleichem** hilft meist eine **Schleife**, und dann solltest du diese auch einsetzen. Genauso kannst du auch einzelne Teile löschen, indem du nach dem Klick mit der rechten Maustaste „**Löschen**“ wählst.

Kostüme



Die Katze läuft jetzt zwar fröhlich über die Bühne, schaut aber immer nach rechts, selbst wenn sie nach links läuft. Um das zu ändern, nutzen wir für unsere Katze ein neues Kostüm auf dem sie nach links schaut.

Öffne den Reiter Kostüme. Du siehst dort, dass die Katze schon zwei Kostüme hat. Um ein weiteres Kostüm zu erstellen, haben wir mehrere Möglichkeiten. Im Moment möchten wir aber unsere Katze nur duplizieren und dann spiegeln. Klicke mit der rechten Maustaste auf Kostüm 1 und wähle „Duplizieren“ aus. Klicke auf das Icon für „Horizontal spiegeln“ und gib dann dem Kostüm einen sinnvollen Namen. Wechsle wieder in den Programmierbereich (Reiter „Code“).

Aufgaben

3. *Erweitere deine Tastatursteuerung, sodass der Katze jeweils vor dem Loslaufen das richtige Kostüm angezogen wird. (Baustein „wechsle zu Kostüm ...“)*
4. *Teste dein Programm und lasse die Katze in alle Richtungen über die Bühne laufen.*
5. *Speichere alles ab.*
6. *Denkaufgabe: Warum nutzen wir den Baustein „ändere x um __“ und nicht „gehe __ Schritte“?*

Teil 6: Die Katze entscheidet – Bedingungen mit Sensoren und Verzweigungen

Ein weiteres zentrales Konzept bei der Programmierung sind **Verzweigungen**. Da diese immer eine **Bedingung** beinhalten, lernst du hier gleich beides kennen.

Zum Beispiel kann eine Figur in Scratch erkennen, ob es ein anderes Objekt oder eine bestimmte Farbe berührt, ob es am Bühnenrand angekommen ist oder auch eine bestimmte Koordinate überschritten hat.

Diese **Bedingungen**, die mit Hilfe von **Sensoren** zwei Dinge vergleichen findest du unter der Kategorie „Fühlen“ und in der Kategorie „Operatoren“.

Strukturen, bei der ein Teil ausgeführt wird, wenn eine Bedingung erfüllt ist, und der andere, wenn die Bedingung nicht erfüllt ist, heißen **Verzweigungen**. Eine Verzweigung ist immer dann vorhanden, wenn man die Formulierung „**Wenn**“ oder „**Falls**“ nutzt. Bedingungen entscheiden dabei ob der **erste Zweig** verwendet wird oder **nicht** (oder der „**sonst**“ Zweig verwendet wird, bevor es weiter geht). Mit Hilfe der **Sensoren** wird die Bedingung entschieden.

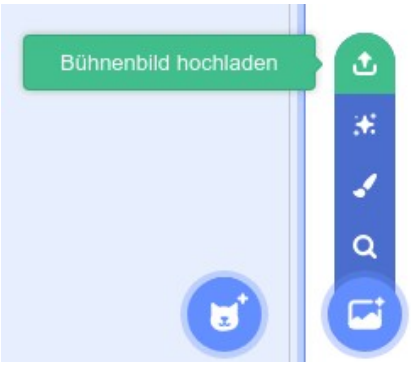
Aufgaben

1. *Lass deine Katze endlos laufen. Wenn sie den Rand berührt, soll sie sich um 30° drehen, ca.10 Schritte gehen (so dass sie vom Rand weg ist). Speichere als* `06_katze_verzweigung`
2. *Lass deine Katze (ohne Endloslauf) bei „Leertaste“ etwas zeichnen: Wenn Deine Katze sich in der rechten Hälfte des Bildes befindet, soll sie ein Dreieck zeichnen. Befindet sie sich in der linken Hälfte, soll sie ein Quadrat zeichnen. Verwende deinen Code aus den vorigen Programmen hierfür wieder.*

3. *(*) Du kannst deine Katze zufällig positionieren. Erweitere deinen Code so, dass die Katze an einem zufälligen Punkt erscheint und abhängig von ihrer Position dann ein Quadrat oder ein Dreieck zeichnet.*
4. *(*) Auch andere Parameter kannst du beeinflussen – experimentiere mit Farbwechseln bei Tastendruck und eigenen weiteren Ideen.*

Teil 7: Ein Auto, eine Straße – Ein Spielesetting entwerfen

Neue Bühnenbilder



Nur über eine weiße Fläche zu laufen, ist für die Katze mit der Zeit langweilig. Deshalb verändern wir den Hintergrund – die so genannte Bühne.

Aufgabe

1. *Lade die Bühne aus der Datei „`straße01.png`“, siehe Bild rechts. Gib dem Bühnenbild einen neuen Namen.*

Neue Figuren

Als nächstes brauchen wir eine neue Figur (solche Figuren nennt ein Programmierer auch Objekte): das erste Auto, das dann auf der Straße fahren soll. Findest du heraus, wie man ein neues Objekt anlegen kann? Wenn nicht, bekommst du hier die Anleitung:

Klicke mal auf den blauen Katzenkopf mit dem „+“ am rechten Ohr, den Du unten rechts in der Figurenleiste siehst: Du hast dann – von oben nach unten – vier Möglichkeiten:

Eine Figur aus einer Datei laden.

Eine zufällig gewählte Figur einfügen.

Eine neue Figur selbst zeichnen.

Eine Figur aus der Scratch-Bibliothek wählen.

Aufgabe

2. *Wähle zunächst (mit der 4. Möglichkeit) ein Auto aus der Bibliothek. Wenn du später magst, kannst du eigene Autos entwerfen. Ändere in den Kostümen die Farbe des Autos.*

Auto bewegen lassen

Das Auto hat einen eigenen Programmierbereich. Wir programmieren das Auto so, dass es beim Klick auf die grüne Fahne losfährt. Wie sich ein Objekt nach rechts bewegen kann, haben wir schon kennen gelernt. Wir verändern einfach die x-Koordinate um einen bestimmten Wert.

Aufgaben

3. *Teste zunächst die Schleife, die eine bestimmte Anzahl an Durchläufen hat. Den Block mit der Anweisung zum Ändern der x-Koordinate ziehst du einfach in den Schleifenblock. Lass das Auto 10-mal nacheinander ein Stückchen nach rechts fahren.*

4. *Teste dann die Schleife, die so lange Anweisungen wiederholt, bis eine Abbruchbedingung erfüllt ist. Das Auto soll bei uns jetzt so lange fahren, bis es am Rand angekommen ist. (Was passiert, wenn das Auto den Rand links berührt?)*

Teil 8: Verschachtelung: Das Auto soll ständig fahren

Für unser geplantes Spiel soll das Auto natürlich nicht am Rand stehen bleiben, sondern möglichst ganz aus dem Bild verschwinden und dann irgendwann wieder auf der linken Seite auftauchen. Und das ununterbrochen.

Um sich im Kopf eine Struktur für einen Programmierschritt zu überlegen, versucht man das, was man erreichen möchte, zuerst in Worte zu fassen.

Hier z.B.: „Wiederhole ständig: bis zum Rand zu fahren, dann wieder links im Bild auftauchen." Das Auto kann obige Schleife verwenden, um zum Rand zu fahren, muss dann nach links „getragen" werden und das ganze soll fortlaufend wiederholt werden. Man muss also zwei Schleifen ineinander bringen (verschachteln).

Eine alternative Denkweise: „Wiederhole ständig, nach rechts zu fahren, falls du am Rand ankommst, setze das Auto nach links." Diese Alternative erfordert *keine* Schleife mit Bedingung. Stattdessen wird allerdings eine Bedingung *innerhalb* der Schleife gebraucht.

Aufgaben

1. *Setze die erste Idee in Scratch um (Bedingte Schleife innerhalb einer Endlosschleife)*
2. *Überlege dir, was es heißt, dass das Auto rechts aus dem Bild gefahren ist. Es genügt uns nicht mehr, dass es nur den Rand berührt. Dafür musst du die alternative Denkweise als Programm bauen (Verzweigung innnerhalb einer Endlosschleife)*

   Ein kleiner Tipp: Schau auch mal in der Anweisungspalette bei den „Operatoren" nach. Vielleicht kannst du etwas davon brauchen.

Hilfekarten

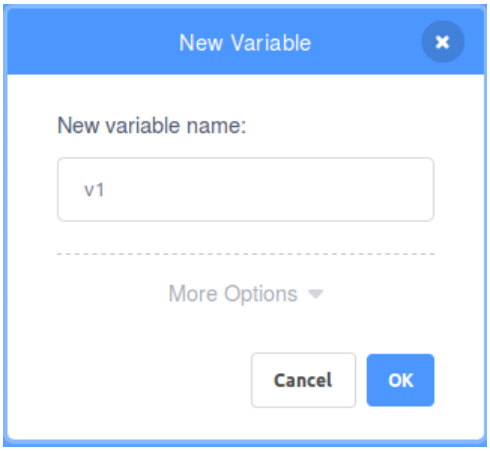
Wenn du Hilfe brauchst, hol dir die passende Hilfekarte.

Wenn du fertig bist, kontrolliere auch mit der Hilfekarte und vergleiche deinen Weg mit dem vorgeschlagenen.

Verändere jetzt den letzten Schleifenblock noch so, dass er beim Start des Programms (beim Klick auf die grüne Fahne) ausgeführt wird und speichere alles ab.

Teil 9: Das Auto ändert die Geschwindigkeit – Variablen

Im Moment bewegt sich das Auto bei jedem Schleifendurchlauf um dieselbe Schrittweite vorwärts. Wir werden unser Programm jetzt so abändern, dass wir mit zwei weiteren Tasten das Auto schneller bzw. langsamer machen können.



Dazu geben wir der Schrittweite einen Namen. Da es sich in gewissem Sinne um die Geschwindigkeit des Autos handelt und diese in der Physik die Abkürzung v erhält, nutzen wir hier v1 für die „Geschwindigkeit“ von Auto1. Du könntest auch einen längeren Namen vergeben, z.B. geschwindigkeit_auto1, aber beim Programmieren verändert man solche Variablen häufig und dann sind lange Bezeichner umständlich.

In der Informatik bezeichnet Wertespeicher als **Variable**. Man benutzt sie, um sich etwas zu merken.

Legt man eine neue Variable an, nennt man das **Deklarieren** der Variable.

Eine Variable kann einen Wert zugewiesen bekommen. Diesen Wert kann man jederzeit verändern. Bekommt eine Variable das erste Mal einen Wert zugewiesen, nennt man das **Initialisierung** der Variable.

Wähle in der Blockpalette „Daten“ und lege eine neue Variable v1 an.

Es werden neue Blöcke erscheinen, die du vermutlich sofort erklären kannst.

Der Wert der Variable wird im linken oberen Eck angezeigt. Im Moment ist das ganz gut, da wir verfolgen können, ob alles so funktioniert, wie wir uns das vorstellen.

Später beim Spiel können wir das Häkchen vor v1 in der Blockpalette entfernen um den Variablenwert nicht mehr auf der Bühne anzuzeigen.

Aufträge:

1. *Initialisiere die Variable v1 beim Start des Programms mit dem bisherigen Wert 10. Überlege dir noch einmal kurz, was die 10 aussagt.*
2. *Wähle dann zwei neue Tastaturereignisse. Die Tasten kannst du selbst wählen, wie du es geschickt findest. Bei der einen Taste erhöhst du den Wert von v1 um einen gewissen Wert, bei der anderen soll etwas subtrahiert werden. (Bei negativer Änderung gibst du der Zahl ein negatives Vorzeichen.)*
3. *Ersetze jetzt an der Stelle, an der x bisher um 10 geändert wurde, die Zahl durch das Symbol der Variable v1.*
4. *Teste dein Programm und speichere es ab.*

Teil 10: Auf Ereignisse exakt reagieren - Boolesche Operatoren

Das Auto fährt im Moment noch irgendwann rückwärts oder wird viel zu schnell. Hier kannst du dein Programm noch verfeinern.

Aufgaben

1. *Baue Grenzen für die Variable v1 ein. Ändere nur, wenn die Grenzen noch nicht über- bzw. unterschritten wurden. Nutze dazu Operatoren (> und <).*
2. *Teste dein Programm, vergleiche es mit der Hilfekarte und speichere wieder ab.*

Wenn mehrere Tastaturereignisse nebeneinander programmiert werden, kann es bei größeren Programmen zu Schwierigkeiten bei der Ausführung kommen.

Wir ändern deshalb den Programmcode für unser Auto. In der Endlosschleife „wiederhole fortlaufend“ befindet sich gerade die Verzweigung, in der die Position des Autos abgefragt und seine Position verändert wird.

Wir wollen in die gleiche Endlosschleife zwei weitere Verzweigungen anhängen.

Falls Taste x gedrückt wird und v1 < 20, dann soll die Geschwindigkeit erhöht werden.

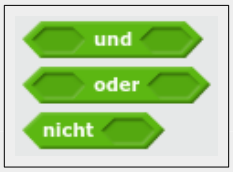
Falls Taste c gedrückt wird und v1 > 5, dann soll die Geschwindigkeit verringert werden.

Aufgaben

3. *Probiere, das Gewünschte in deinem Programmcode zu ergänzen.*
   (Mache dich für die gewünschten Blöcke auf die Suche in „Steuerung“, „Fühlen“ und „Operatoren“.)
4. *Speichere das veränderte Programm unter einem neuen Namen 06katzeV2 ab und teste hinterher beide Varianten. Merkst du einen Unterschied?*

Du hast für die Bedingung in der Verzweigung ein „und“ verwendet, um zwei Bedingungen zu verknüpfen.

Information: logische Operatoren



Neben „und“ findest du noch „oder“ und „nicht“ bei den Operatoren. Diese drei nennt man logische Verknüpfungen oder logische Operatoren oder Boolesche Operatoren.

„und“ bedeutet, dass beide Bedingungen gleichzeitig erfüllt sein müssen,

„oder“ bedeutet, dass mindestens eine der beiden Bedingungen erfüllt sein muss.

Teil 11: Die Katze muss aufpassen – Reaktion auf andere Objekte

Jetzt kennst du dich schon so gut aus, dass du erst einmal probieren kannst, wie es weitergeht.

Die Katze möchte ja heil und sicher über die Straße kommen. Wenn das Auto die Katze berührt, soll zunächst einmal das Auto stoppen.

Speichere dein Projekt ab und probiere es aus.

Hast du eine Lösung? Wenn ja, vergleiche sie mit der zugehörigen Hilfekarte.

Jetzt bist du bereit, dein Projekt zu erweitern. Die nächsten Aufgaben sind unabhängig voneinander. Du kannst selbst entscheiden, in welcher Reihenfolge du sie machst. Natürlich kannst du auch eigene Ideen umsetzen, um das Spiel spannender zu machen.

Aufgaben

1. *Überlege dir, was passieren soll, wenn die Katze angefahren wird. Setze deine Ideen um.*
2. *Lasse auf der Straße ein weiteres Auto fahren und programmiere alles Nötige für die Kollision mit der Katze.*
3. *Baue einen Zähler ein, der mitzählt, wie oft die Katze heil am oberen Rand angekommen ist.*
4. *Gib der Katze eine bestimmte Anzahl an Leben. (Vielleicht sieben?) Wenn sie diese alle verloren hat, endet das Spiel.*
5. *Erweitere dein Spiel um weitere Level. Diese können immer schwieriger werden. Lege dazu neue Bühnenbilder an und lasse die Katze, wenn sie am oberen Rand angekommen ist, auf die neue Bühne wechseln. Überlege dir auch, was passieren soll, wenn sie erwischt wird. (Start in dem Level? Start in Level1?)*
6. *Füge ein Level mit Fluss ein, auf dem Baumstämme schwimmen, auf die die Katze springen muss, um über den Fluss zu kommen.*

Ein kleiner **Tipp**:

Objekte können miteinander in Verbindung treten, indem sie sich „**Nachrichten**“ schicken und auf solche reagieren. Die nötigen Blöcke dazu findest du in der Anweisungspalette unter „Ereignisse“.

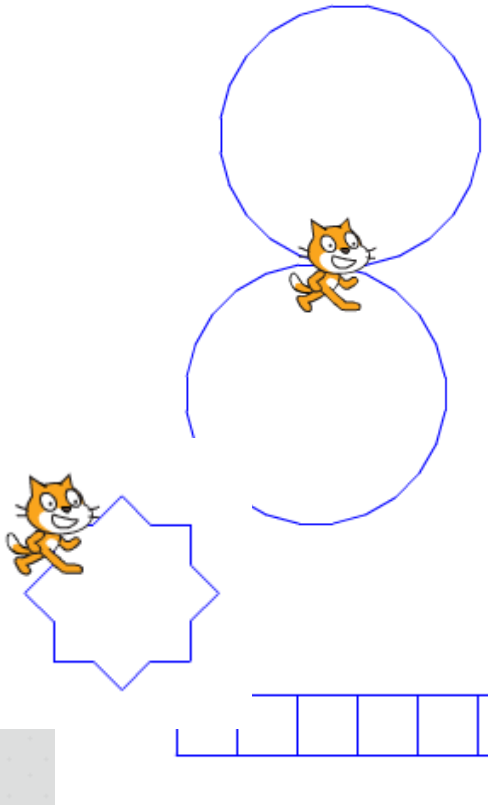
Gib deinen Nachrichten sinnvolle Namen, damit du später noch weißt, wozu sie gut sind.

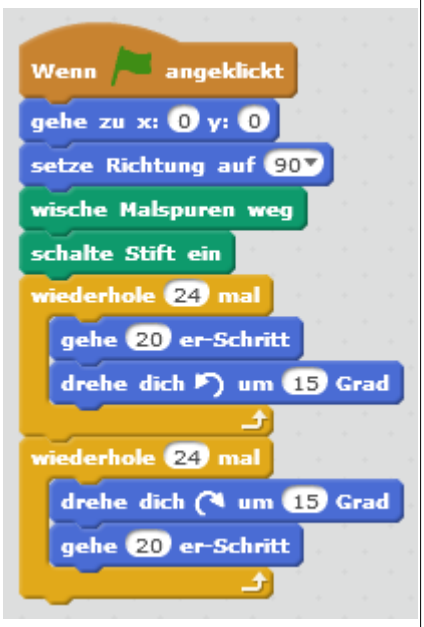
Anhang A: Katzengeometrie – Übungen zu Schleifen und Verzweigungen

Speichere die folgenden Aufgaben in einer eigenen Datei `A_katze_zeichnet`.

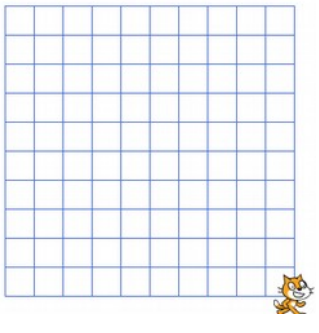
Aufgaben

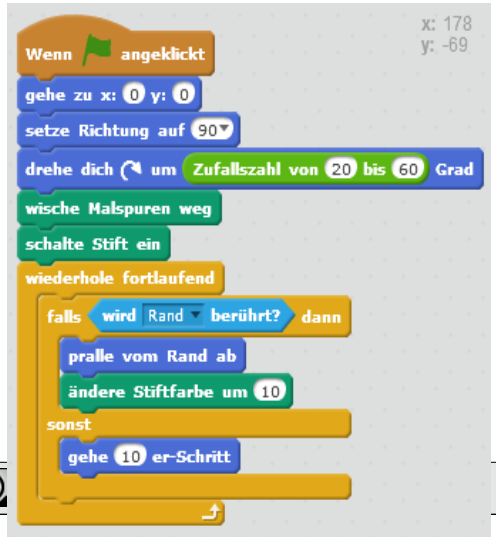
1. *Was passt zusammen? Verbinde und begründe kurz.*





2. *In der ersten Aufgabe gibt es ein Programm, in dem die Katze zehn Quadrate nebeneinander zeichnet. Erweitere das Programm so, dass die Katze zehnmal zehn Quadrate zeichnet.*

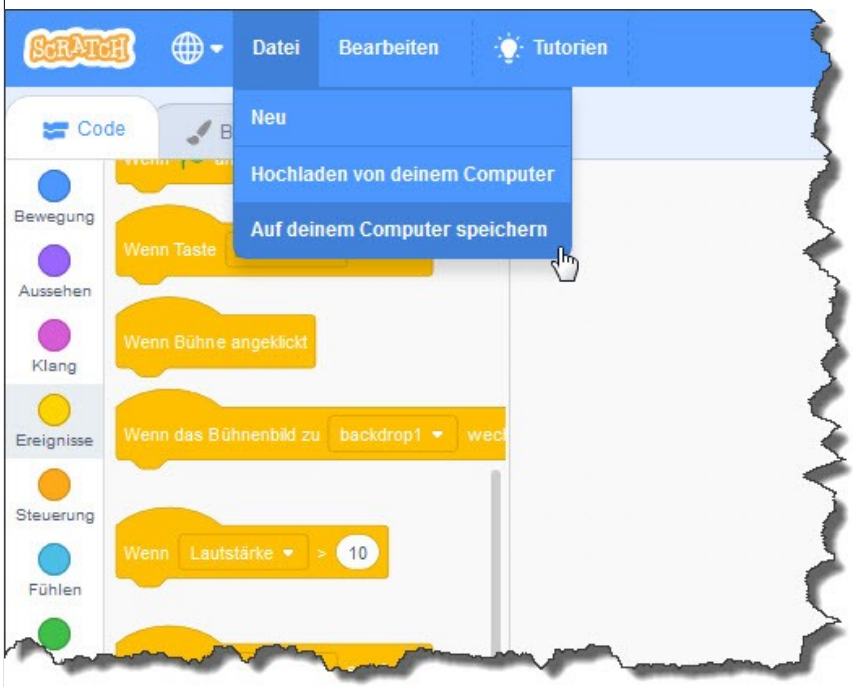




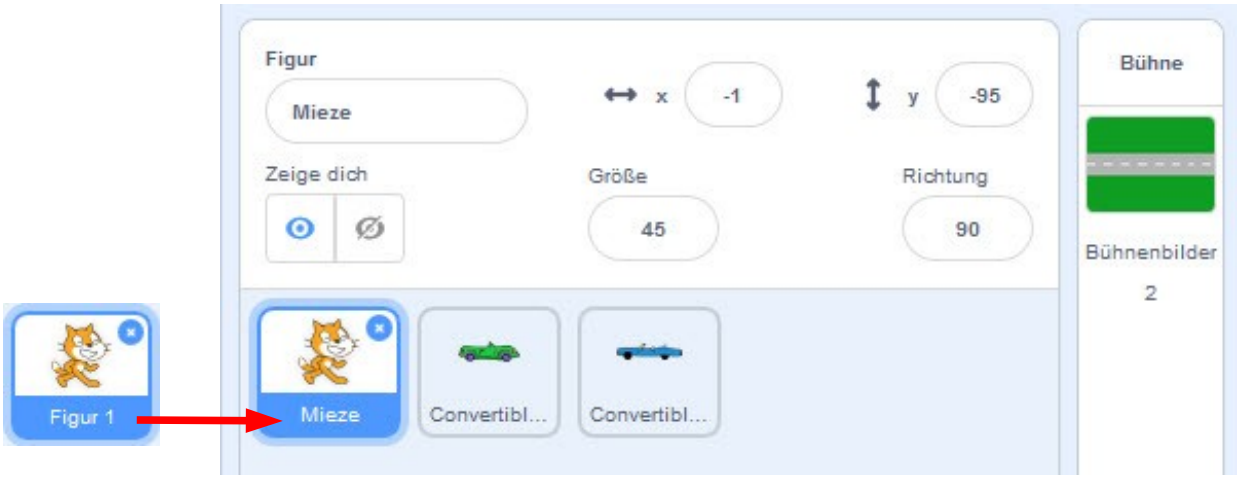
3. *Beschreibe, was der folgende Programmcode auslöst.*
4. *(*) Schreibe ein Programm, das die Katze das „Haus vom Nikolaus“ zeichnen lässt. Verändere es dann so, dass sie fünf Häuser direkt nebeneinander zeichnet.*

Anhang B1: Hilfekarten zu den Aufgaben – Teil 1

Datei speichern:

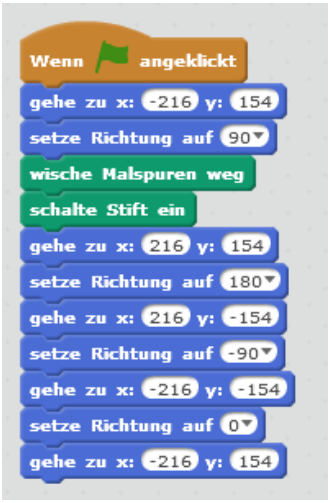


Name des Objektes ändern:

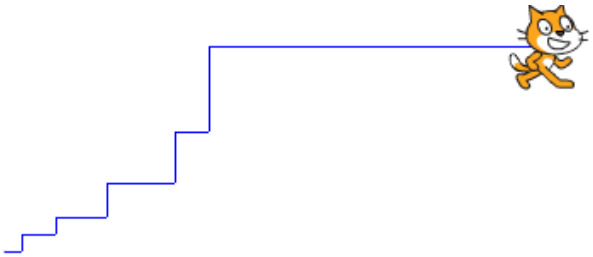


Anhang B3: Mögliche Lösungen zu den Aufgaben – Teil 3

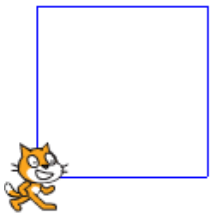
Aufgabe 1:

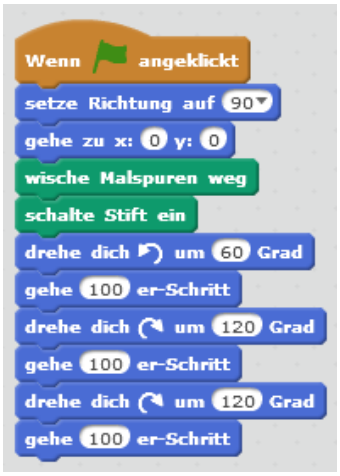


Aufgabe 2:

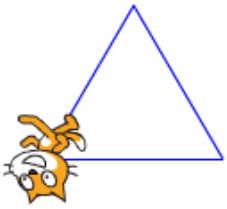


Aufgabe 3:



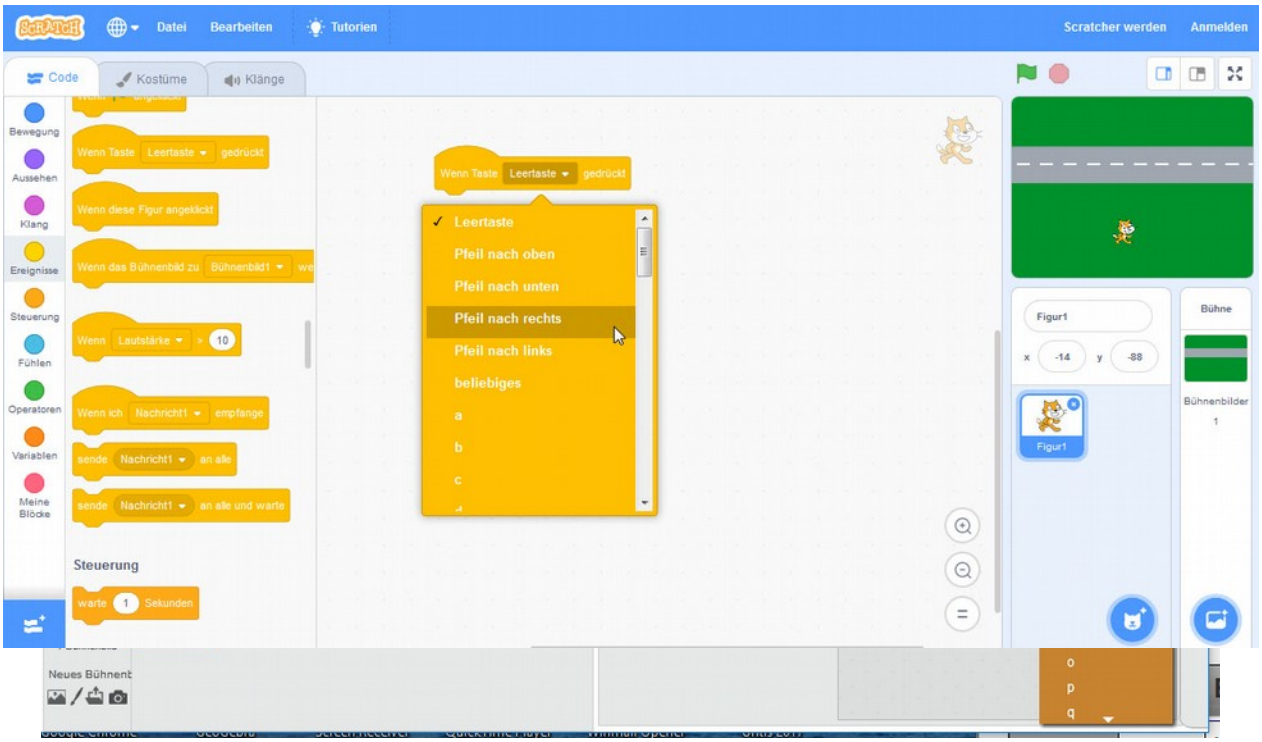


Aufgabe 4:



Anhang B5: Hilfekarten zu den Aufgaben – Teil 5

Einstellung der Taste beim Tastaturereignis:



Aufgabe 2:

Aufgabe 3:

Läuft die Katze nach oben oder unten, behält sie das Kostüm bei, das sie gerade anhat.

Anhang B8: Hilfekarte zu den Aufgaben – Teil 8

Ergebnis:

Schleifenblock:

Durch „wiederhole fortlaufend“ wird der Inhalt des Schleifenblockes so lange ausgeführt, bis das Programm beendet wird.

Diese Schleife nutzen wir, weil das Auto ja zunächst einmal dauernd fahren soll.

Würden wir nur das „ändere x um 10“ in die Schleife ziehen, würde Scratch uns da keinen Strich durch die Rechnung machen – wenn das Auto am Rand hängen) – das Auto immer weiter nach rechts aus der Bühne fahren.

Verzweigungsblock:

Weil wir wollen, dass das Auto nur fährt, falls es sich noch nicht am rechten Rand befindet, ziehen wir einen Verzweigungsblock in die Schleife.

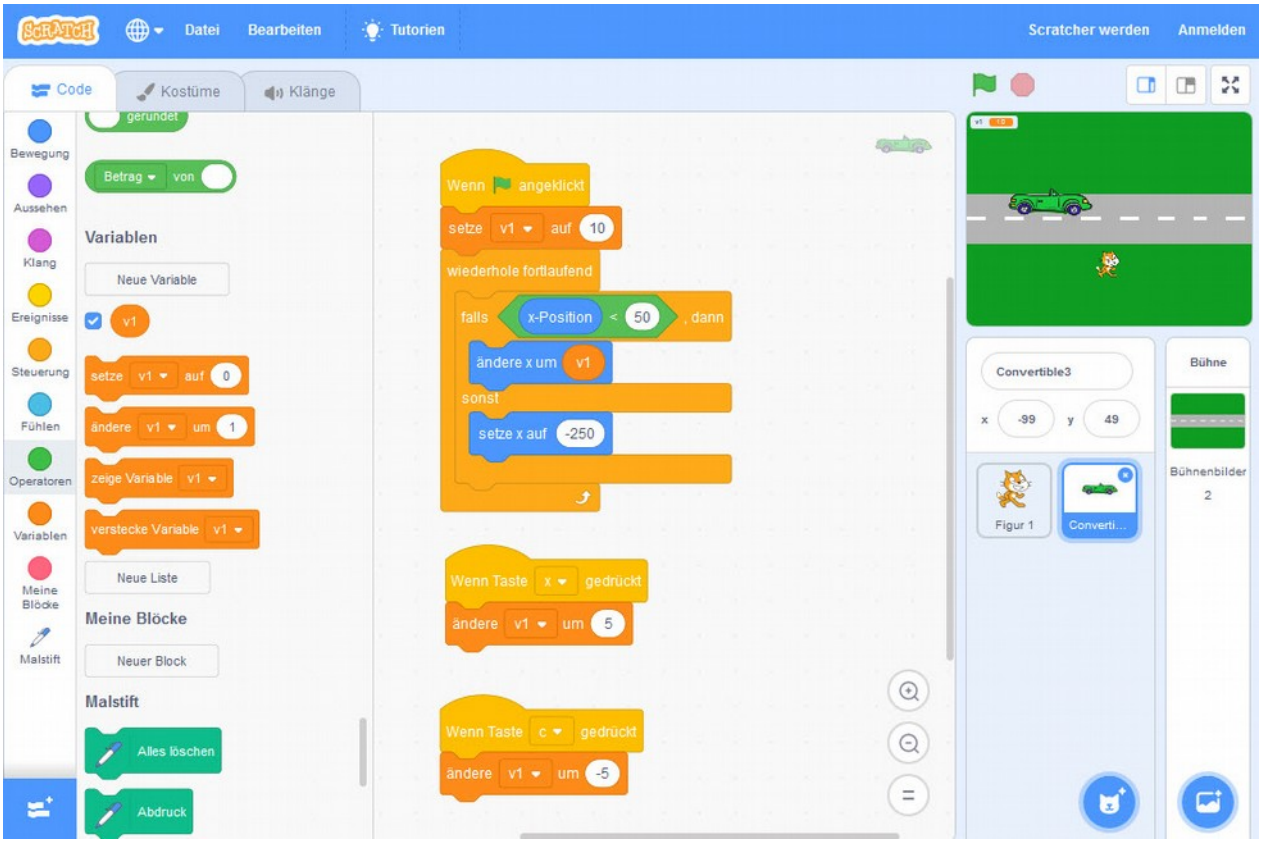
Falls die Bedingung erfüllt ist, wird die Anweisung (oder Sequenz) ausgeführt, die im oberen Teil nach dem „falls“ steht, und ist die Bedingung nicht erfüllt, das was im unteren Teil nach dem „sonst“ steht.

Für solche Verzweigungen müssen wir uns geschickte Bedingungen überlegen. Hier ist es sinnvoll, danach zu schauen, ob die x-Position des Autos kleiner ist als der x-Wert am rechten Rand.

Falls ja, fährt das Auto (ändert seine x-Position um 10); falls nein, wird es links wieder abgesetzt.

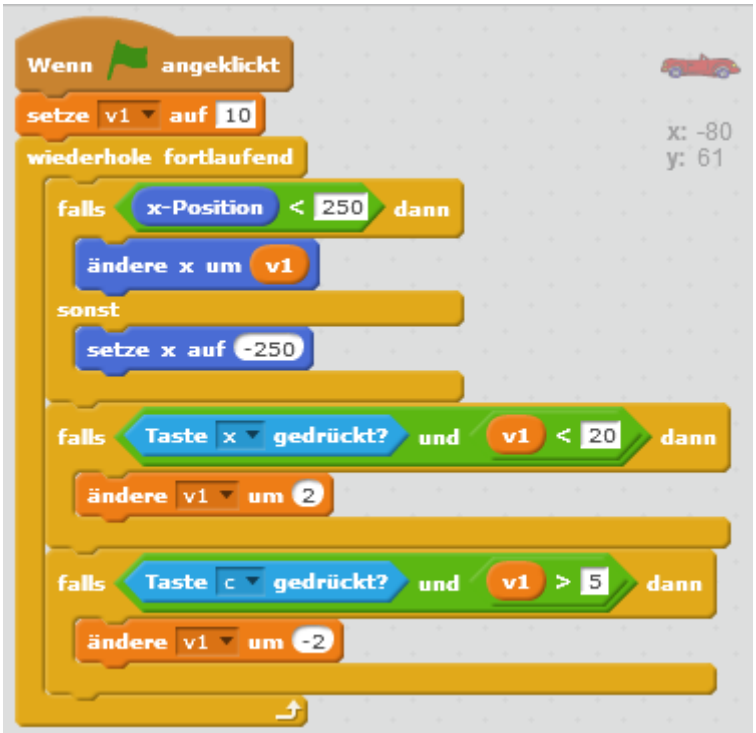
Anhang B10: Hilfekarte zu den Aufgaben – Teil 10

Aufgaben 1 – 4:



Aufgabe 5:

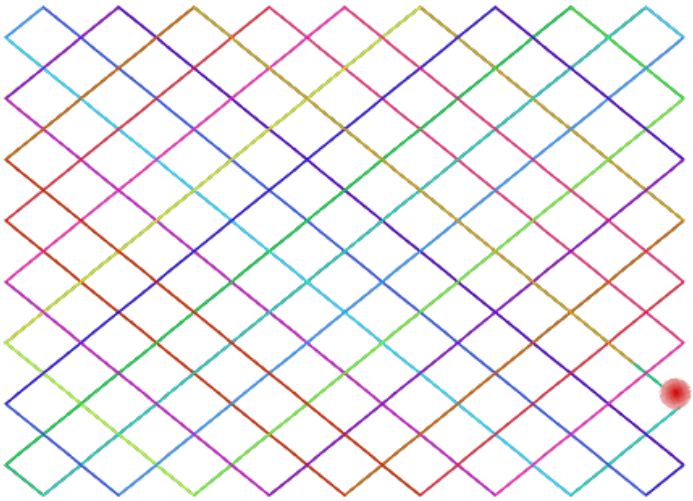
Aufgabe 7:


Wenn angeklickt
setze v1 auf 10
wiederhole fortlaufend
falls x-Position < 250 dann
ändere x um v1
sonst
setze x auf -250
falls Taste x gedrückt? und v1 < 20 dann
ändere v1 um 2
falls Taste c gedrückt? und v1 > 5 dann
ändere v1 um -2
x: -80
y: 61

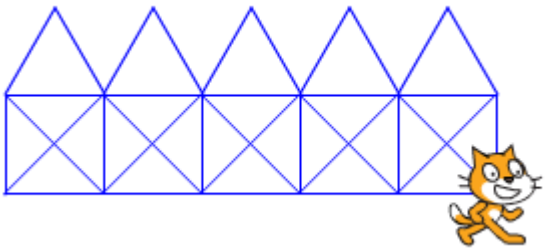
Anhang B12: Hilfekarte zu den Aufgaben – Teil 12

Auto stoppt, wenn die Katze berührt wird:

Anhang BA: Die Katze zeichnet – Lösungsvorschläge:




Wenn angeklickt
gehe zu x: -217 y: 0
setze Richtung auf 90
wische Malspuren weg
schalte Stift ein
wiederhole 5 mal
ändere y um 50
drehe dich um 60 Grad
warte 0.5 Sek.
gehe 50 er-Schritt
drehe dich um 120 Grad
warte 0.5 Sek.
gehe 50 er-Schritt
warte 0.5 Sek.
ändere x um -50
setze Richtung auf 90
drehe dich um 45 Grad
warte 0.5 Sek.
gehe 70 er-Schritt
warte 0.5 Sek.
ändere x um -50
drehe dich um 90 Grad
warte 0.5 Sek.
gehe 70 er-Schritt
warte 0.5 Sek.
ändere y um -50
warte 0.3 Sek.
setze Richtung auf 90
Das
ist
das
Haus
vom
Ni...
...ko...
...laus



Anhang C: Vorbemerkung für Lehrer: Scratch-Versionen und Alternativen

Die aktuelle Scratch-Version 3.6.0 (Stand 10/2019) basiert auf HTML5. Es gibt die Online-Version und eine Offline-Version, die auf Windows, Mac und Linux läuft.

Neben dem offiziellen Download der Offline-Versionen für Windows und Mac, findet sich hier unter *https://scratux.org* eine von der Community beigesteuerte Offline-Version für Linux.

Scratch 2

Die bisherige (beliebte) Scratch-Version 2.0 basiert auf Adobe Flash. Aufgrund gravierender Sicherheitslücken ist Adobe Flash in den letzten Jahren mehr und mehr in Verruf geraten. Seit etwa 2015 rät die Fachpresse[1] den Flash-Player nicht mehr zu verwenden, Browserhersteller gingen dazu über, das Flash-Plugin standardmäßig zu deaktivieren. Schließlich hat Adobe letztes Jahr angekündigt, die Entwicklung von Flash 2020 einzustellen, keine Updates mehr bereitzustellen und die Software nicht mehr zu verbreiten[2]. Es gibt eine Offline-Version, die unter wine funktioniert und mit dem sicherheitslückenbehafteten Adobe Air auch nativ unter Linux funktioniert.

Die Alternative: Snap!

Dennoch lohnt ein Blick auf die Alternative. An der Uni Berkeley wird mit Snap! eine weitere Sprache / Entwicklungsumgebung für visuelles Programmieren entwickelt. Die Benutzeroberlfäche von Snap! ähnelt Scratch weitestgehend, was einfach daher rührt, dass Snap! anfangs nur ein modifiziertes Scratch 1.x war. Auch wenn für die aktuelle Version der Code komplett neu geschrieben wurde, merkt man doch die gemeinsamen Wurzeln.

Insgesamt ist Snap! mächtiger als Scratch. So kann man mit Snap! beispielsweise eigene Prozeduren mit Parametern und Rückgabewerten definieren und auf diese Weise rekursive Algorithmen implementieren.

Auch Snap! läuft im Browser (*https://snap.berkeley.edu/run/*), eine offline-Version gibt es nicht.

Abwägung

Die Entscheidung, ob man in Zukunft auf Snap! oder Scratch setzen will, ist eine Abwägung: Einerseits funktioniert unter Snap! derzeit schon einiges, was unter Scratch noch nicht implementiert ist. Zudem läuft es schon jetzt stabil und ohne Flash. Durch den erweiterten Funktionsumfang kann man mit Snap! ggf. auch noch nach Klasse 7 weiterarbeiten. Andererseits hat Scratch im Moment die größere Community und wird, wenn der Umstieg gut gelingt, vermutlich auch das häufiger verwendete Programm bleiben. Auch der offline-Editor und die Verwendbarkeit auf Tablets spricht aus technischer Sicht für Scratch.

1 z.B. *https://www.heise.de/ct/ausgabe/2015-5-Editorial-Goodbye-Flash-2536897.html* oder *https://www.heise.de/security/meldung/Flash-Player-deaktivieren-Schon-wieder-Angriffe-auf-ungepatchte-Luecke-2535100.html*

2 *https://www.heise.de/newsticker/meldung/Adobe-verabschiedet-sich-von-Flash-2020-ist-Schluss-3783264.html*