



WERKGYMNASIUM HEIDENHEIM

INFORMATIK 7

Scratch-Projekt

Name:

Autor:
Fabian DRECHSLER

5. Juni 2022

Bewertungsraster			
Text 1	Text 2	Text 3	
Summe Punkte:			
Projektnote:			

Inhaltsverzeichnis

1	Einleitung	4
2	Die ersten Programme	5
2.1	Scratch-Oberfläche	5
2.2	Neue Befehle	6
2.3	Programme mit Rechnungen	11
2.4	Teste dich selbst	14
3	Vorgänge wiederholen - Einfache Schleifen	16
4	Parameter und Variablen	21
4.1	Was ist ein Parameter?	21
4.2	Variablen	21
4.3	Neue Befehle	24
4.4	Zusammenfassung	24
4.5	Teste dich selbst	25
5	Verzweigungen und bedingte Schleifen	26
5.1	if-Verzweigung (Befehl ausführen, wenn ...)	26
5.2	EXKURS: Logische Operatoren	27
5.3	while-Schleifen	28
5.4	Neue Befehle	29
5.5	Teste dich selbst	30
6	Projekt	32
6.1	Steuerung der Spielfigur	32
6.2	Kostüme	33
6.3	Die Katze läuft über die Straße	35

1 Einleitung

Programmieren bezeichnet die Tätigkeit Programme für Computer zu erstellen. Computerprogramme werden mit Hilfe einer Programmiersprache, die der Computer versteht, entwickelt (codiert). Teilweise werden für die Programmierung Codegeneratoren verwendet, die Programmcodes aus vordefinierten Blöcken automatisch erzeugen.

In unserem Basiskurs verwenden wir die Programmiersprache **Scratch**. **Scratch** ist eine erstmals 2007 veröffentlichte bildungsorientierte visuelle Programmiersprache für Kinder und Jugendliche inklusive ihrer Entwicklungsumgebung. Sie wurde am MIT entwickelt und kann als Web-App in einem modernen Browser genutzt werden oder als Offline-Editor. Die mobilen Browser Chrome und Safari bieten eine Touch-Unterstützung an.

Web-App: <https://scratch.mit.edu/projects/editor>

Offline-Editor: <https://scratch.mit.edu/download>

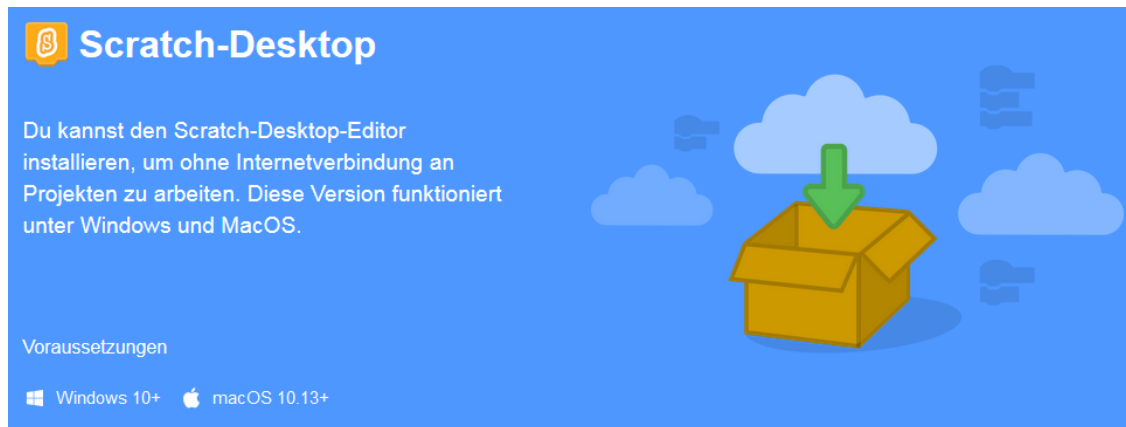


Abbildung 1: Downloadseite des Offline-Editors

Wie jede natürliche Sprache bestehen Programmiersprachen aus Wörtern, die eine bestimmte Bedeutung haben. Man verwendet eine Programmiersprache um einem Computer Anweisungen zu geben. Deshalb heißen die Wörter einer Programmiersprache Befehle oder auch Instruktionen. Ein Programm besteht aus einer Reihe von Befehlen.

Ziel des Programmierens ist es Tätigkeiten eines Computers zu automatisieren. Computer können nur die Befehle ausführen mit denen sie programmiert wurden und können im Zweifelsfall nicht frei entscheiden. Damit Befehle eindeutig sind und muss die Tätigkeit, die automatisiert werden soll, vollständig verstanden werden. Erst dann können Programmierer die Tätigkeit in eine Programmiersprache übersetzen. Bei Scratch werden die Befehle mittels Codeblöcken an den Computer gegeben. Man kann mehrere Blöcke per Drag&Drop aneinander hängen und so zu komplexen Programmen gelangen.

Im nächsten Kapitel lernst du die ersten Befehle und kleine Programme kennen.

2 Die ersten Programme

2.1 Scratch-Oberfläche

Die Oberflächen des Online- und Offline-Editors sind identisch und sehr übersichtlich gestaltet.

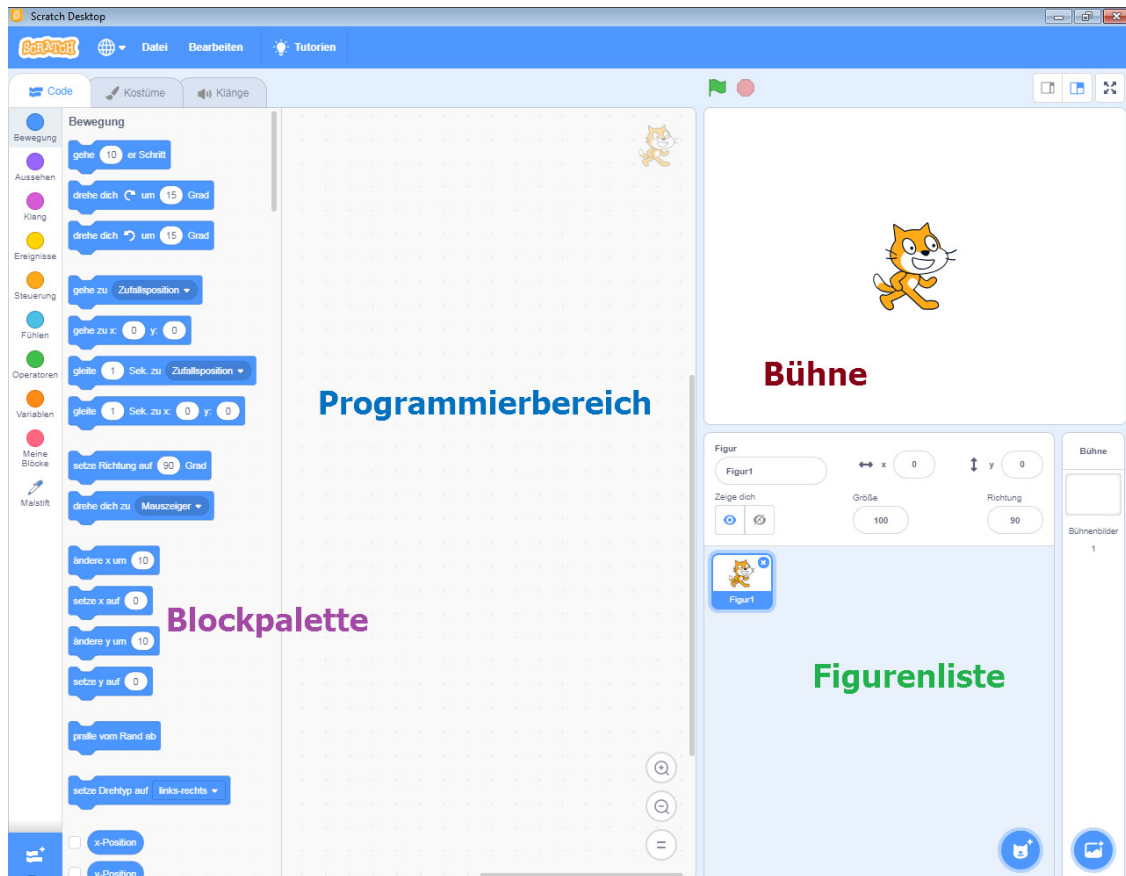


Abbildung 2: Scratch-Oberfläche

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 rechten Seite den Hintergrund deiner Bühne, die Bühnenbilder.

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. Die Anweisungen werden in verschiedene Bereiche unterteilt, die sich farblich unterscheiden.

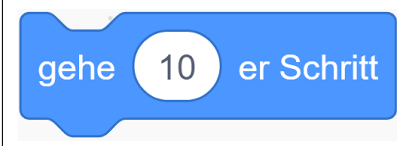
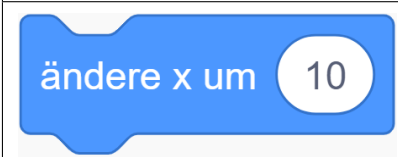
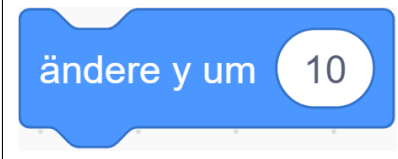
Ähnlich wie LEGO-Steine werden die Blöcke im **Programmierbereich** zu einem Programm zusammengefügt.

Hinweis: Falls Scratch nicht mit der Standardsprache Deutsch gestartet ist, kann die Sprache über den Globus links oben geändert werden.

2.2 Neue Befehle

Die Bühne in Scratch ist wie ein Koordinatensystem in der Mathematik aufgebaut. Die x-Achse verläuft waagrecht und die y-Achse senkrecht. Der Ursprung ($x = 0; y = 0$) liegt im Mittelpunkt der Bühne.

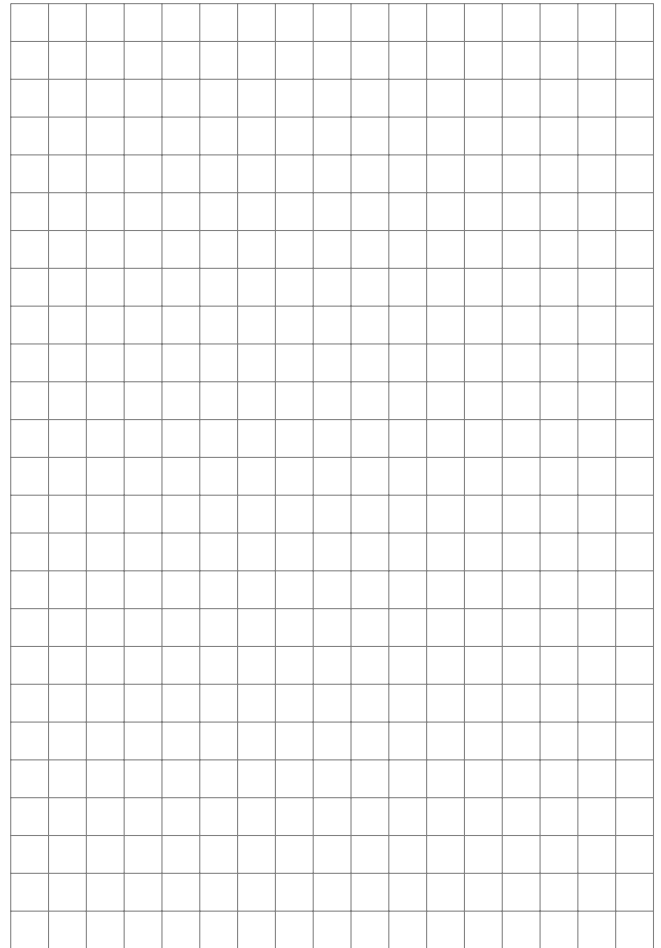
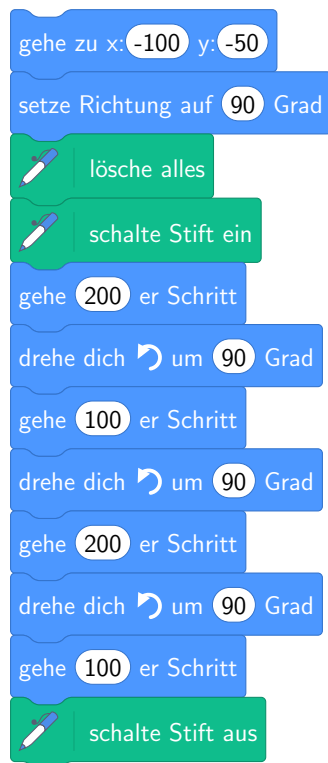
Zur Steuerung der Katze auf der Bühne gibt es folgende grundlegende Befehle:

	Die Figur bewegt sich 10 Schritte in die aktuelle Richtung. Setzt man eine negative Zahl für die Anzahl der Schritte, läuft die Figur rückwärts.
	Setzt die Richtung der Figur in Richtung des Pfeils. Die Gradzahl kann auch direkt eingegeben werden. 0° ist dabei senkrecht nach oben und nimmt im Uhrzeigersinn zu.
	Setzt die Figur auf die Koordinaten $x = 19; y = 5$.
	Setzt die x-Koordinate der Figur auf 50. Die y-Koordinate bleibt unverändert.
	Setzt die y-Koordinate der Figur auf 50. Die x-Koordinate bleibt unverändert.
	Ändert die x-Koordinate um die Zahl. Es sind sowohl positive als auch negative Änderungen möglich.
	Ändert die y-Koordinate um die Zahl. Es sind sowohl positive als auch negative Änderungen möglich.

Damit man die Befehle (und eventuelle Fehler im Programm) leichter nachvollziehen kann, lässt man seine Figur ihre Bewegungen auf der Bühne zeichnen. Dazu benötigt man die Erweiterung *Malstift* für Scratch. Die Übersicht der verfügbaren Erweiterungen kann man über einen Klick auf das blaue Symbol in der linken unteren Ecke (siehe Abb. 2) öffnen. Ein zweiter Klick auf den *Malstift* fügt diesen zur Blockpalette hinzu.

Beispiel 1:

Das folgende Programm zeichnet ein Rechteck der Größe 100×200 . Die Größe wird in der Anzahl der Schritte gemessen.



Überlege und skizziere neben dem Codeblock, welchen Weg die Katze bei der Ausführung des Programm läuft. Programmieren den Codeblock am Computer und überprüfe deine Hypothese, indem du ihn ausführen lässt (Doppelklick auf den Codeblock).

Speichere jede der folgenden Aufgaben in einer einzelnen Datei ab. Der Dateiname soll die Nummer der Aufgabe sein (Bsp: "Aufgabe 2.1").

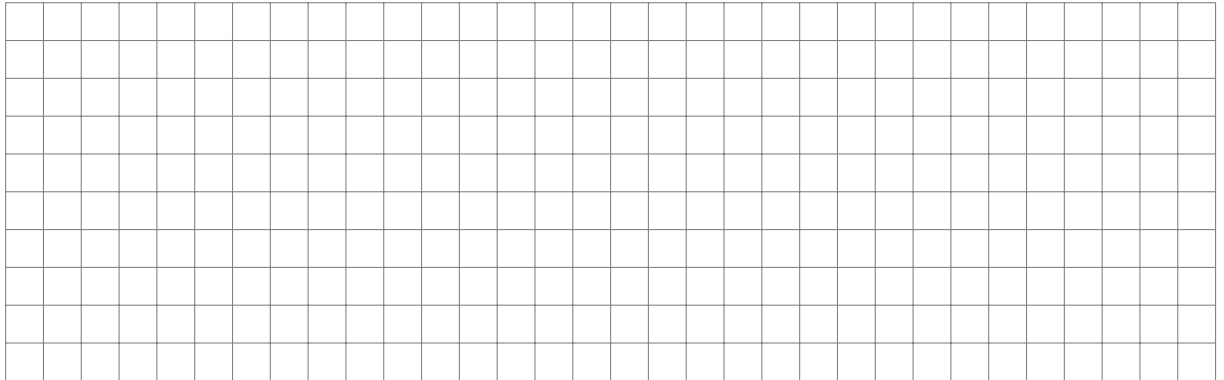
Aufgabe 2.1:

Wandle das Programm aus Beispiel 1 so ab, dass die Katze ein Quadrat der Größe 150×150 zeichnet.

Aufgabe 2.2:

Programmiere ein Skript für deine Figur so, dass sie ein gleichseitiges Dreieck mit der Seitenlänge 250 zeichnet. Gehe folgendermaßen vor:

1. Skizziere hier den Weg, den deine Katze laufen soll.

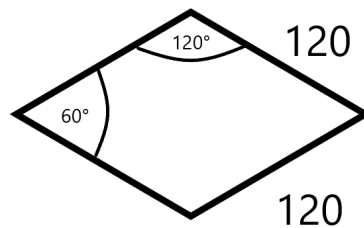


2. Überlege, wie groß die Innenwinkel eines gleichseitigen Dreiecks sind. Die Figur dreht jeweils um 180° minus den Innenwinkel des Dreiecks.
3. Programmiere dein Skript in Scratch. (Tipp: Setze deine Figur zu Beginn deines Programms in das linke untere Viertel der Bühne.)

Aufgabe 2.3:

Schreibe ein Programm, dass die abgebildete Skizze zeichnet.

Lege als erstes einen geeigneten Startpunkt fest und die Blickrichtung fest.



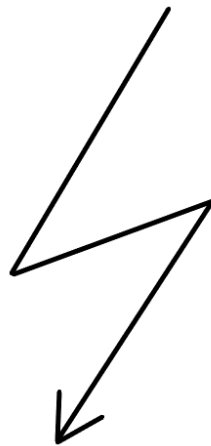
Farben

Die Figur kann in verschiedenen Farben und Strichstärken zeichnen. Hier eine kleine Übersicht:

 schalte Stift ein	Schaltet den Malstift ein.
 schalte Stift aus	Schaltet den Malstift aus. Es ist sinnvoll am Ende eines Programms den Stift auszuschalten um keine ungewollten Linien zu zeichnen, wenn die Figur danach bewegt wird.
 lösche alles	Löscht alle bisherigen Malspuren, z.B. nachdem die Figur an ihrer Startposition angekommen ist.
 setze Stiftfarbe auf 	Legt die Farbe des Stifts fest. Ein Klick in das Farbfeld öffnet den Farbeditor.
 setze Stiftstärke auf 	Setzt die Strichstärke auf die gewünschte Größe.

Aufgabe 2.4:

Schreibe ein Programm, das einen Blitz (siehe Skizze) zeichnet. Wähle selbst die Länge der Linien und der Winkel. Bonus: Ändere die Farbe der Linie so, dass es tatsächlich wie ein Blitz aussieht.



Beispiel 2:

Im nächsten Schritt soll die Katze ein regelmäßiges Sechseck wie in Abbildung 3 zeichnen.

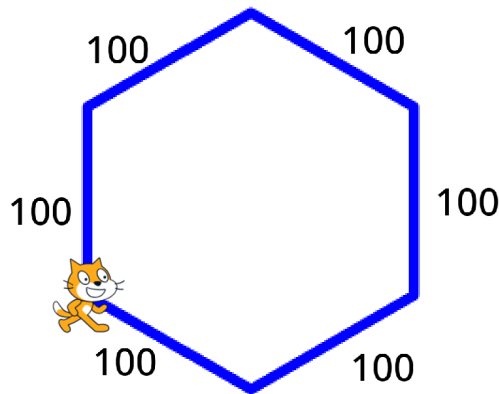


Abbildung 3: Sechseck

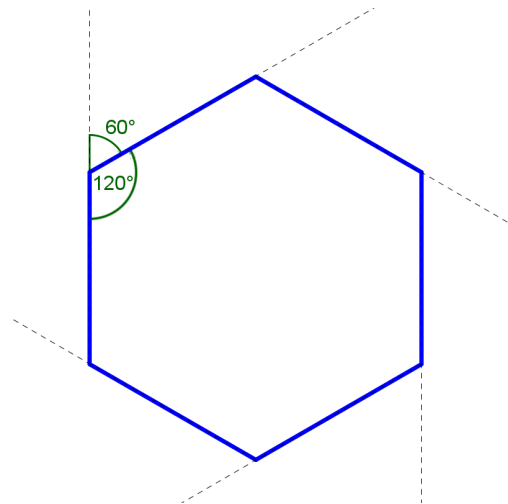
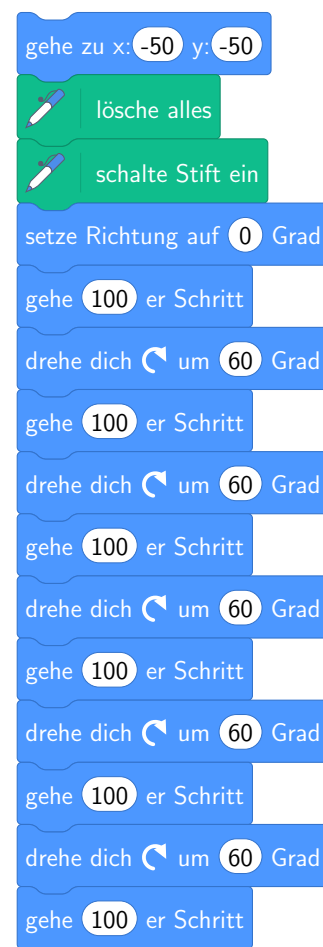


Abbildung 4: Sechseck mit Winkeln

Vor der Programmierung des Skripts müssen wir festlegen, an welcher Stelle unsere Figur mit dem Zeichnen beginnt. In Abb.3 sitzt die Katze bereits an entsprechender Stelle.

Als nächstes überlegen wir, um wie viel Grad sich die Katze nach dem Zeichnen einer jeden Seite drehen muss (siehe Abb. 4). Damit zum Schluss ein regelmäßiges Sechseck entsteht müssen alle sechs Winkel gleich groß sein. Die Katze startet mit Blickrichtung nach oben (In Scratch: Richtung 0°) und befindet sich am Schluss ihrer Bewegung wieder an ihrer Startposition. Sie hat nach dem 6-maligen eine vollständige Drehung von 360° ausgeführt. Der Drehwinkel an jeder Ecke ist somit $\frac{360^\circ}{6} = 60^\circ$. Die ist exakt der Nebenwinkel zum Innenwinkel des regelmäßigen Sechsecks.

Nachdem wir jetzt den Ablauf des Programms kennen, können wir es in Scratch programmieren. Schreibe das Programm ab und führe es aus.



Aufgabe 2.5:

Schreibe ein Programm, das ein regelmäßiges Fünfeck mit Seitenlänge 110 zeichnet. Gehe wie in Beispiel 2 vor (Skizze auf Papier, Winkel einzeichnen ...).

2.3 Programme mit Rechnungen

Computer können mathematische Berechnungen sehr schnell durchführen, weshalb sie auch "Rechner" genannt werden. Rechnungen sind in Scratch im Block *Operatoren* untergebracht.

Beispiel 3:

Wir wollen den Term $(163 \cdot 3) - (77 \cdot 4)$ berechnen lassen. Dafür schreiben wir in Scratch folgendes Programm:



Abbildung 5: Beispiel für die Berechnung eines mathematischen Terms.

Ein Klick auf den Codeblock führt die Berechnung aus und zeigt das Ergebnis in einer Sprechblase darunter an. Damit lässt sich Scratch nicht nur als (umständlicher) Taschenrechner nutzen, sondern man kann diese Berechnungen direkt in seine Programme einbauen.

Beispiel 4:

Erinnern wir uns an Beispiel 2, in dem wir das regelmäßige Sechseck gezeichnet haben. Die Figur dreht sich in 6 Schritten insgesamt um 360° . Sie dreht sich an jeder Ecke um $\frac{360^\circ}{6}$. Wir müssen dies nicht selbst berechnen sondern können es das Programm überlassen. Unser Beispiel sieht jetzt aus wie nebenan.



Aufgabe 2.6:

Schreibe ein Programm, das ein regelmäßiges Siebeneck zeichnet. Die Seitenlänge kannst du selbst wählen.

Beispiel 5:

Wir möchten ein rechtwinkliges Dreieck wie in Abb. 6 zeichnen. Das Dreieck hat einen 90° und zwei 45° Winkel. Die Katheten (kurzen Seiten) haben die Länge x . Die Länge der Hypotenuse y (lange Seite) lässt sich mit dem Satz des Pythagoras ("Die Summe der Kathetenquadrate ist gleich dem Quadrat der Hypotenuse") berechnen. Du lernst ihn in Klasse 9 im Fach Mathematik kennen. Für jetzt reicht das Ergebnis: $y \approx x \cdot 1,414214$.

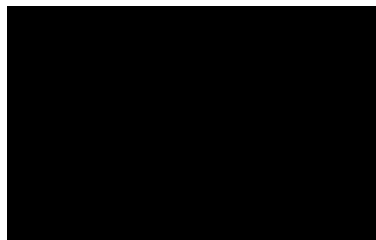
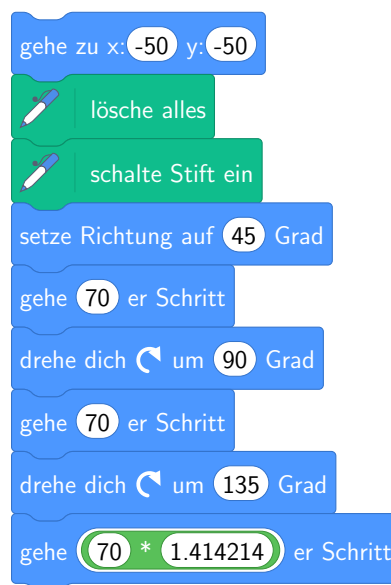


Abbildung 6: Rechtwinkliges Dreieck

Anstatt die Länge umständlich vorher zu Berechnen, übernimmt dies jetzt unser Programm. Achte darauf, statt eines Dezimalkommamas einen Dezimalpunkt zu verwenden.



Aufgabe 2.7:

Schreibe ein Programm, welches mit Scratch das Haus des Nikolaus (siehe Abb. 7) zeichnet. Es besteht aus mehreren rechtwinkligen Dreiecken wie in Beispiel 5. Damit kannst du überlegen, wie viel länger/kürzer die einzelnen Strecken sind. Das Grundquadrat hat eine Seitenlänge von 185.

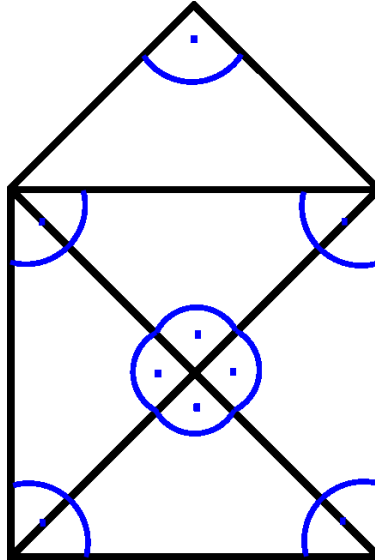
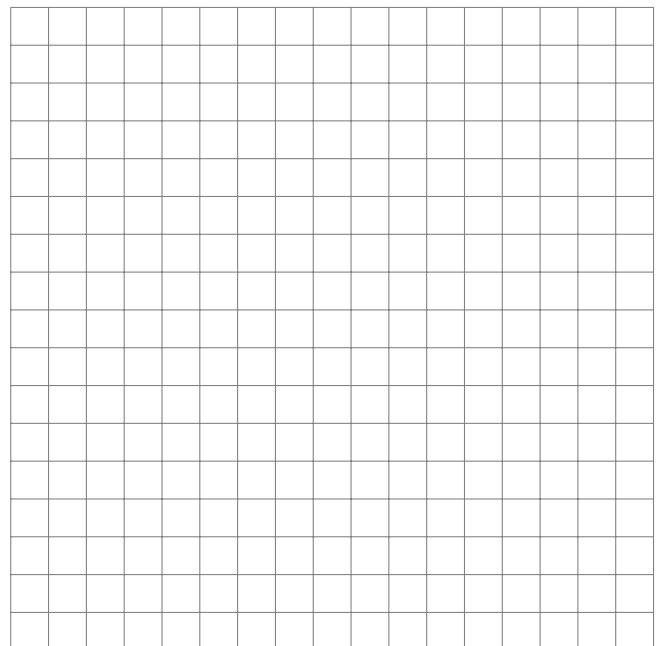


Abbildung 7: Schema Haus des Nikolaus

Wissensfragen

-

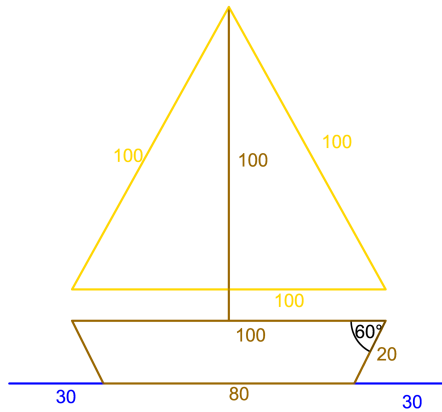
- [illegible]



Programmieraufgaben

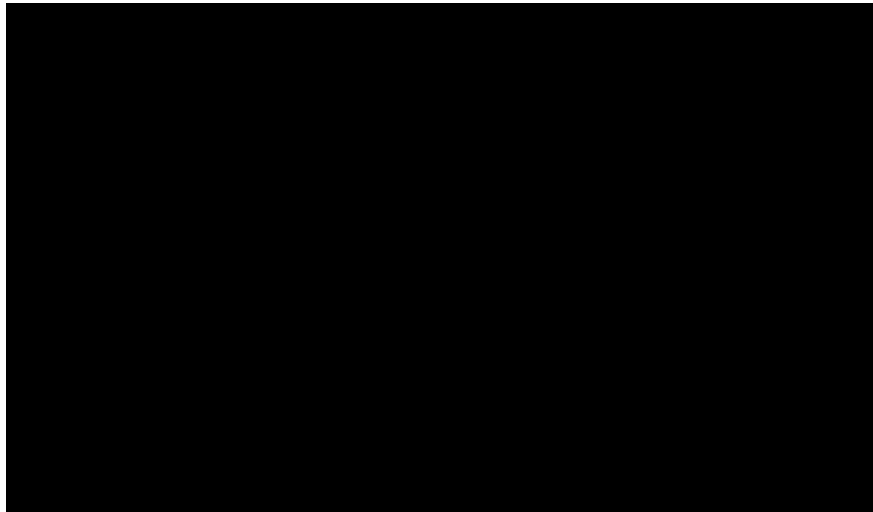
Aufgabe 2.8:

Schreibe ein Programm, das mit der Katze ein Segelschiff wie in der Skizze zeichnet. Verwende die Maßangaben aus der Skizze.



Aufgabe 2.9:

Die folgende Abbildung zeigt ein kleines Quadrat, das um 45° gedreht in einem großen Quadrat liegt. Die Seiten des großen Quadrats sind 1,414214 Mal so lang wie die Seiten des kleinen Quadrats. Schreibe ein Programm, das diese Figur mit den zwei Quadraten zeichnet. Das große Quadrat soll eine Seitenlänge von 218 haben und die Katze soll sich am Ende verstecken.



3 Vorgänge wiederholen - Einfache Schleifen

In unserem Alltag kommen immer wieder Routinetätigkeiten vor die einfach und langweilig sind. Mit Hilfe der Informatik kann man solche langweiligen Tätigkeiten automatisieren. Computer können immer mehr und komplexere Aufgaben übernehmen und führen zu einer Entlastung der Menschen und bieten die Möglichkeit sich mit kreativen Aufgaben zu beschäftigen.

Dafür automatisieren wir Vorgänge und lassen Computer die Tätigkeiten übernehmen.

Du hast beim Programmieren sicher schon bemerkt, dass es manchmal eintönig ist, immer die gleichen Befehle hintereinander zu schreiben, um z.B. ein Quadrat oder ein Sechseck zu zeichnen. Eintöniger wird es, wenn man gar ein 100-Eck zeichnen möchte. Man müsste 100-mal die Befehle

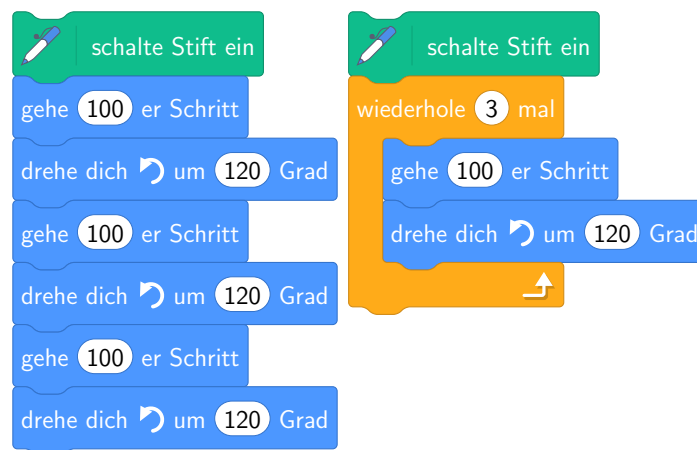


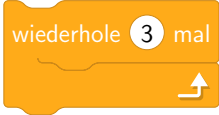
wiederholen. Dies muss einfacher möglich sein!

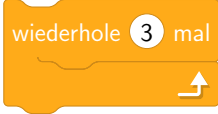
In diesem Kapitel lernst du, wie du dem Computer auf mitteilen kannst eine Folge von Befehlen x-mal auszuführen. Damit ist es dir möglich schon komplexere Automatisierungsaufgaben zu bewältigen, ohne die Übersicht zu verlieren und ohne zu lange Programme schreiben zu müssen.

Beispiel 6:

Mit der Anweisung "wiederhole 3 mal" sagst du dem Computer, dass er eine Folge von Befehlen dreimal wiederholen soll. Vergleiche die beiden folgenden Ausschnitte aus einem Programm. Sie führen exakt die gleiche Tätigkeit aus:



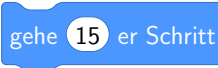
Info: Zu jeder -Anweisung gehören die Anzahl der Wiederholungen und eine

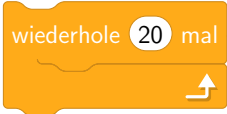
Folge von Befehlen, die wiederholt werden sollen. -Anweisung zusammen mit

den zu wiederholenden Befehlen nennt man eine **Schleife**.

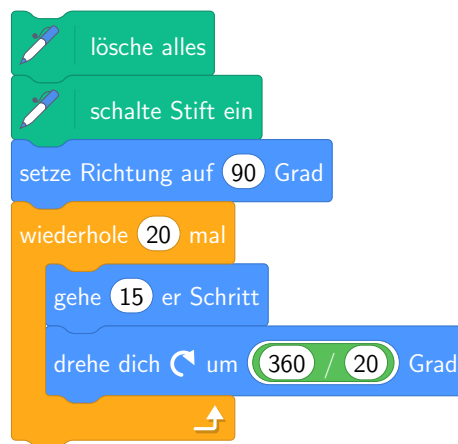
Bei Programmieren beinhaltet eine Schleife immer einen Programmteil, der mehrfach ausgeführt wird. Dieser Programmteil heißt **Körper der Schleife**

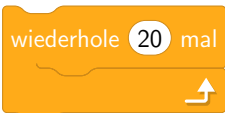
Beispiel 7:

Um ein 20-Eck zu zeichnen, müsstest du die Befehle  und

 20-mal schreiben. Weil wir aber die -Anweisung

verwenden, weiß der Computer, dass er die beiden inneren Befehle 20 mal wiederholen soll.



Die **Schleife** besteht aus der -Anweisung mit der Anzahl der Wiederholun-

gen. Davon eingeschlossen stehen die Befehle, die eine bestimmte Anzahl Male ausgeführt

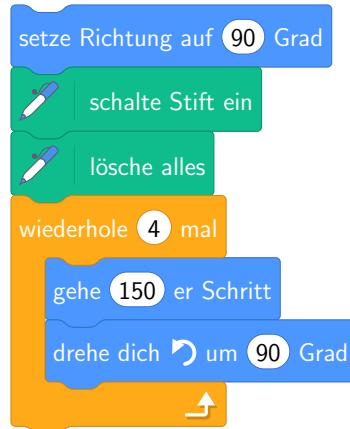
werden sollen. Die Befehle  und  sind der

Körper der Schleife.

Beispiel 8:

Erinnerst du dich noch an die Aufgabe, bei der du ein Quadrat zeichnest? Für das Quadrat wurden viermal die beiden Befehle `gehe 150 er Schritt` und `drehe dich um 90 Grad` verwendet.

Dies lösen wir nun kürzer:



Die Befehle vor der Schleife stellen sicher, dass unsere Figur immer in die gleiche Richtung startet und alte Malspuren gelöscht werden.

Aufgabe 3.1:

Schreibe jeweils ein Programm, um ein regelmäßiges Vieleck mit der geforderten Anzahl der Ecken und der geforderten Seitenlänge zu Zeichnen.

(Tipp: Starte bei den Koordinaten $x : 0$ $y : 120$ und verwende vor der Schleife jeweils den gleichen Block, wie in Beispiel 8. Die verschiedenen Programmblöcke speicherst du bitte alle in der gleichen Datei.)

- A 11 Ecken, Seitenlänge 50
- B 15 Ecken, Seitenlänge 30
- C 7 Ecken, Seitenlänge 100

Beispiel 9:

Der Programmblock in diesem Beispiel zeichnet den Stern wie in Abbildung 8 daneben. Schreibe das Programm ab und führe es aus. (Setze zur besseren Übersicht die Figurgröße auf 20%.)

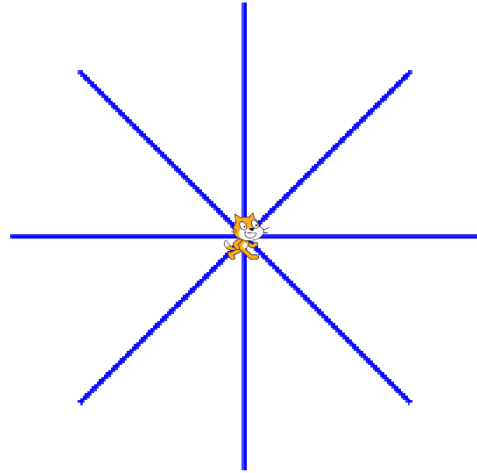
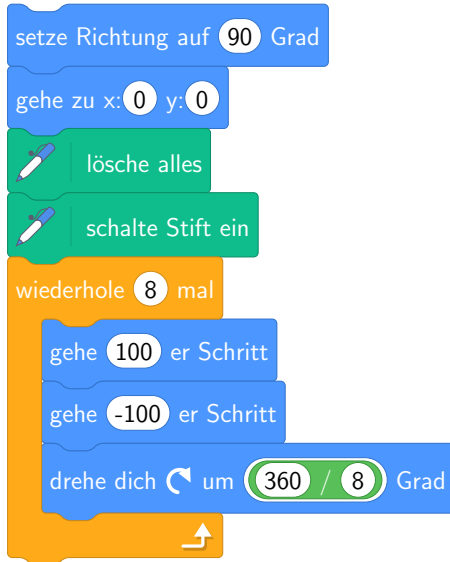


Abbildung 8: Stern mit 8 Spitzen

Aufgabe 3.2:

- A Ändere das Programm aus Beispiel 9 so ab, dass der Stern doppelt so viele Zacken hat (siehe Abbildung 9).
- B Zeichne einen Stern mit 360 Zacken.

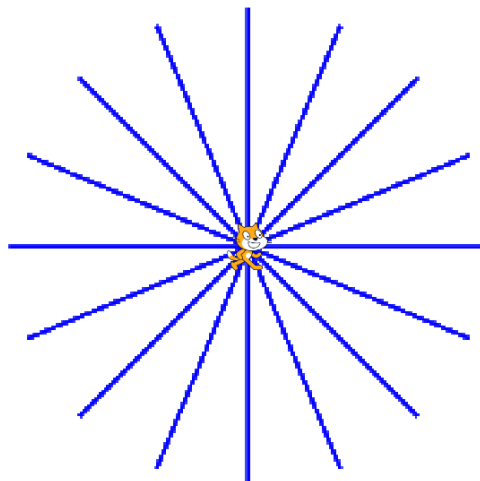


Abbildung 9: Stern mit 16 Zacken.

Aufgabe 3.3:

(Schwer) Schreibe ein Programm, das einen fünfzackigen Stern wie in der Abbildung 3.3 zeichnet.

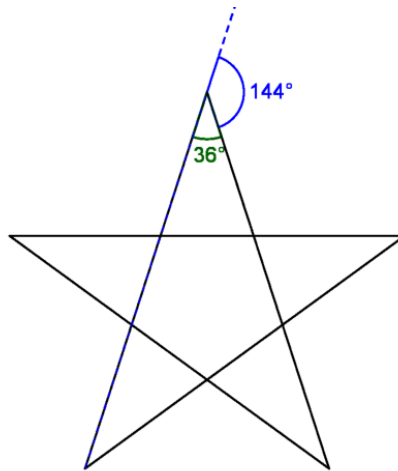


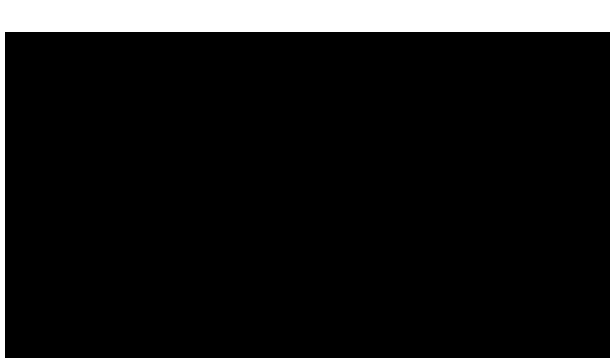
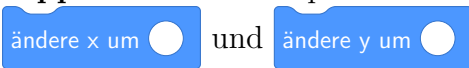
Abbildung 10: Pentagramm

Aufgabe 3.4:

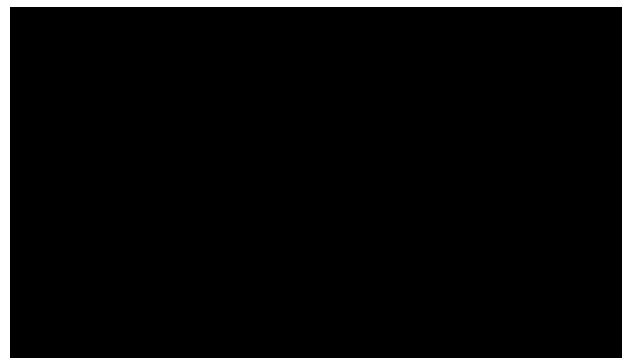
Schreibe drei Programme (bzw. Programmblöcke) und speichere alle in einer Datei, die eine Treppe wie in Abbildung 11(b) zeichnen. Die Höhe und Länge der einzelnen Treppenstufen (siehe Abbildung 11(a)) wählst du wie folgt:

- A Anzahl 10, Höhe 25, Länge 25
- B Anzahl 25, Höhe 5, Länge 10
- C Anzahl 12, Höhe 20, Länge 5

Tipp: Setze den Startpunkt in den linken unteren Bereich der Bühne. Verwende die Blöcke



(a)



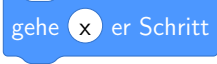
(b)

Abbildung 11: Darstellung der Stufen

4 Parameter und Variablen

In diesem Kapitel lernst du, was ein **Parameter** ist, **Variablen** zu deklarieren und zu nutzen.

4.1 Was ist ein Parameter?

Beim Befehl  darf für **x** eine Zahl eingesetzt werden. Diese Zahl, die man für **x** einsetzt, legt fest, wie viele Schritte die Figur gehen soll. Bei , wählt man für **winkel** eine Zahl und gibt an, um wie viel Grad sich die Figur nach links drehen soll. Den Namen **x** in  oder **winkel** in  nennt man einen **Parameter**. Der Zahlenwert, den man dabei einsetzt, heißt **Wert des Parameters**. Die Werte eines Parameters dürfen auch andere Objekte als Zahlen sein: Ein Parameter kann für eine Farbe stehen, dann ist der Wert des Parameter ein Farbname.

4.2 Variablen

Mit Parametern kann man mit einem einzigen Befehl, je nach Parameterwert, unterschiedliche Tätigkeiten ausführen lassen. Zum Beispiel genügt ein Befehl, um eine Anzahl an Schritten zu gehen.

Aus Computersicht ist ein Parameter ein Speicherplatz mit einem bestimmten Namen. Alle Parameter eines Befehls bilden eine Tabelle und immer wenn der Computer den Wert eines Parameters benötigt, liest er den Wert aus dieser Tabelle aus. Er arbeitet anschließend mit dem Wert, der im Speicherplatz **x** (oder **winkel**) abgelegt ist. Dieses Konzept von Parametern verallgemeinern wir zu **Variablen**.

Parameter nutzen wir um Werte unter einem Namen zu speichern und später wieder zu verwenden. Bei **Variablen** kommt hinzu, dass die Werte während der Ausführung des Programms geändert werden können und die geänderten Werte beim Zeichnen, Rechnen, ... zu nutzen.

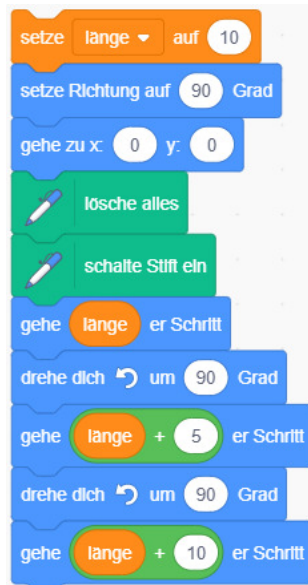
Variablen werden in Scratch über den Button „neue Variable“ angelegt.

Beispiel 10:

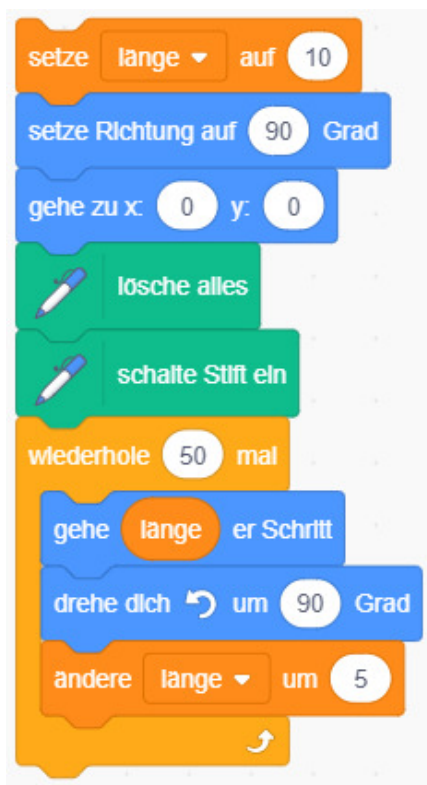
Wir möchten eine Spirale zeichnen. Die Figur soll jede Linie ein wenig länger als die vorherige zeichnen. Dies ist eine einfache Übung, wird aber sehr umfangreich, wenn die Spirale aus 50 oder mehr Seiten bestehen soll. Von den vielen notwendigen Blöcken ist nur ein kleiner Teil abgebildet.

Das Programm wiederholt immer wieder die gleichen Befehle:  und

. Nur die Schrittweite wird geändert.



Dieses Programm soll nun mit einer Schleife programmiert werden und der Computer soll so viel Arbeit übernehmen wie es geht. Das sieht dann so aus:



Zu Beginn wird der Startwert der Variable „länge“ auf 10 festgelegt und die Figur auf eine Startposition bewegt.

Anschließend wird die Schleife 50-mal durchlaufen.

Der Befehl am Ende der Schleife erhöht bei jeder Wiederholung den Wert der Variable „länge“ um 5.

Lege die Variable „länge“ an und zeichne die Spirale.

Aufgabe 4.1:

Ändere das Programm aus Beispiel 10 so ab, dass es eine Spirale mit folgenden Eigenschaften zeichnet:

- Die erste Seite hat die Länge 6.
- Jede weitere Seite ist um 2 länger.
- Die Spirale hat 36 Seiten.

Aufgabe 4.2:

Entwickle ein Programm, das eine Spirale von außen nach innen zeichnet. Die Spirale hat folgende Eigenschaften:

- Lege eine geeignete Startposition fest.
- Die erste Seite hat die Länge 200.
- Jede weitere Seite ist um 3 kürzer.
- Die Spirale hat insgesamt 50 Seiten.

Aufgabe 4.3:

Ändere das Programm in Beispiel 10 so, dass eine sechseckige Spirale entsteht. Die Länge der ersten Seite ist 10. Die Seite wird immer um das 0,1-fache der aktuellen Länge geändert. Die Schleife wird 18 Mal wiederholt.

Lösung:

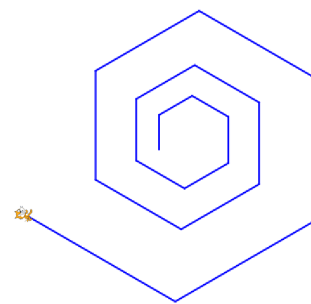


Abbildung 12: Lösung Aufgabe sechseckige Spirale

Aufgabe 4.4:

Schreibe ein Programm, das eine sechseckige Spirale zeichnet (siehe Abb. 13). Das Programm soll die Variablen „Anzahl“ und „Länge“ verwenden. Der Parameter Anzahl gibt die Anzahl der gezeichneten Seiten an und Länge beschreibt die Weite der ersten Seite. Während des Zeichenvorgangs soll die Länge der Seite immer um 3 erhöht.

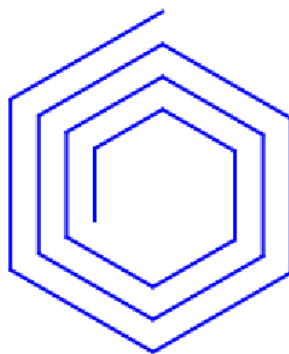
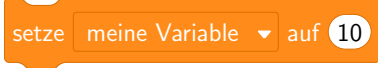


Abbildung 13: Schema Spirale

4.3 Neue Befehle

	Legt eine neue Variable an. Eine Variable ist ein Speicherplatz mit einem Namen. In einer Variablen kann immer genau ein Wert gespeichert werden.
	Legt den Wert einer Variablen fest.
	Erhöht den Wert der Variablen um den angegebenen Wert. Um den Wert der Variablen zu verringern können negative Werte eingetragen werden.

4.4 Zusammenfassung

Variablen heißen so, weil sich ihr Wert während der Ausführung verändern (variieren) darf. Variablen, deren Wert zu Beginn eines Programms festgelegt werden und danach nicht mehr verändern, heißen **Parameter**. Mit den Blöcken  und

 kann man den Wert einer Variablen festlegen bzw. ändern.

Bei jeder Programmausführung legt der Computer eine Tabelle an, in der zu jeder Variablen ihr Wert steht. Ist die Variable in der Tabelle noch nicht vorhanden, so wird sie der Tabelle mit ihrem Wert hinzugefügt. Existiert die Variable bereits, wird der Wert der Variablen geändert.

Computer speichern immer nur den aktuellen Wert der Variablen, alte Werte werden nicht gespeichert.

4.5 Teste dich selbst

Wissensfragen

1. Was ermöglichen Variablen, was Parameter alleine nicht können?

2. Warum nennt man eine Variable „Variable“?

3. Was ist der Unterschied zwischen einer Variable und einem Parameter?

4. Woher weiß der Computer, welchen Wert eine Variable hat? Wie kann sich der Computer die Werte aller Variablen merken?

5. Mit welchem Befehl kannst du den Wert einer Variablen um eine feste Zahl erhöhen?

6. Mit welcher Kombination aus Befehlen kannst du den Wert einer Variablen verdoppeln?

5 Verzweigungen und bedingte Schleifen

5.1 if-Verzweigung (Befehl ausführen, wenn ...)

Beim Programmieren kommt es häufig vor, dass das Programm auf verschiedene Situationen unterschiedlich reagieren soll. Man denke an einfach die Steuerung einer Figur, deren Richtung von unterschiedlichen Tasten abhängt.

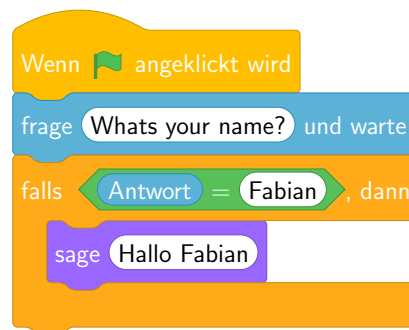
Dafür benötigt man die *if*-Anweisung. Diese prüft, ob eine Bedingung erfüllt ist („if“ bedeutet im Englischen „wenn“ oder „falls“). In Scratch steht hier der Block



zur Verfügung.

Beispiel 11:

Herr Drechsler hat den Computer so programmiert, dass er ihn erkennt und begrüßt, wenn er sich mit seinem Vornamen vorstellt. Jeder andere wird nicht begrüßt. Bei einer *if*-Anweisung prüft der Computer die Bedingung, ob der Wert der Variable **Antwort** gleich „Fabian“ ist. Nur wenn dies der Fall ist, führt er den Code innerhalb der Anweisung aus und sagt die Begrüßung.



Aufgabe 5.1:

Schreibe einen Begrüßungsprogramm für dich selbst (Es darf gerne komplexer sein.)

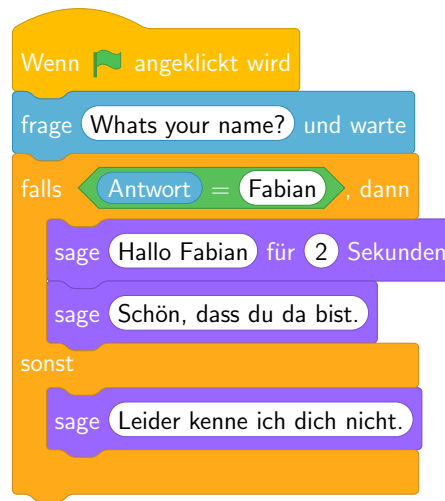
Mit einer *if*-Anweisung sagt man dem Computer, dass er den Block nur ausführen soll, wenn die Bedingung erfüllt ist. Wenn diese nicht erfüllt ist, dann wird der Block komplett übersprungen.

Man kann in einer *if*-Anweisung auch direkt angeben, was der Computer machen soll, wenn die Bedingung nicht erfüllt ist. Das ist eine *if-else*-Anweisung („else“ bedeutet im Englischen „andernfalls“).

Bei einer *if-else*-Anweisung wählt der Computer eine von zwei Alternativen aus. Je nachdem, ob die Bedingung erfüllt ist führt er den ersten Codeblock aus, andernfalls den zweiten.

Beispiel 12:

Das folgende Programm ähnelt dem im vorherigen Beispiel. Hier wird direkt dem Computer mitgeteilt, was er tun soll, wenn die Bedingung `Antwort = Fabian` nicht erfüllt ist.



5.2 EXKURS: Logische Operatoren

Mit der if-Anweisung (`falls,  dann`) prüfst du, ob eine Bedingung erfüllt ist. Wenn sie

erfüllt ist, führt der Computer die Anweisungen im Körper der if-Anweisung aus. In manchen Fällen, soll etwas ausgeführt werden, wenn die Bedingung nicht erfüllt ist. Oder der Computer soll die Anweisung nur ausführen, wenn zwei Bedingungen gleichzeitig erfüllt sind. Dies geschieht in einer Programmiersprache mit logischen Operatoren, die du in diesem Kapitel kennen lernst.

Negation

Das Gegenteil der Behauptung „die Wand hat die Farbe blau“ lautet „die hat nicht die Farbe blau“.

Das Gegenteil der Behauptung „x ist unterschiedlich von 5“ lautet „x ist gleich 5“.

Das Gegenteil von „ $x < 5$ “ lautet „ $x \geq 5$ “. Statt dem Begriff „Gegenteil“ verwendet man das Wort „Negation“. Wir sagen also „ $x \geq 5$ “ ist die Negation von „ $x < 5$ “.

Wahrheitstabellen

Mit „und“ und „oder“ kann man zwei Behauptungen A und B verknüpfen und so komplexe Aussagen treffen. Um alle möglichen Varianten der Verknüpfungen übersichtlich darzustellen, verwendet man sogenannte Wahrheitstabellen. Für zwei Behauptungen A und B gibt es in einer Wahrheitstabelle genau 4 Möglichkeiten:

1. Beide Behauptungen A und B gelten.
2. A gilt und B nicht.
3. A gilt und B nicht.

4. Beide Behauptungen A und B gelten nicht.

Jede dieser vier Möglichkeiten entspricht einer Zeile in der Wahrheitstabelle:

Behauptung A	Behauptung B	Verknüpfung →	A oder B	A und B
gilt	gilt		gilt	gilt
gilt	gilt nicht		gilt	gilt nicht
gilt nicht	gilt		gilt	gilt nicht
gilt nicht	gilt nicht		gilt nicht	gilt nicht

In der den ersten beiden Spalten sind alle möglichen Kombination von A und B aufgelistet. Die dritte Spalte gibt die Verknüpfung der ersten beiden durch den Operator „A oder B“ wieder. Diese gilt immer dann, wenn mindestens eine der beiden Behauptungen A oder B gilt. Nur im letzten Fall, wenn beide Behauptungen nicht gelten gilt „A oder B“ ebenfalls nicht.

Die letzte Spalte zeigt die Verknüpfung „A und B“, Diese Verknüpfung gilt nur, wenn sowohl A als auch B gilt, d.h. beide Bedingungen müssen erfüllt sein. In allen anderen Fällen gilt „A und B“ nicht.

Bedingungen verknüpfen in Scratch

Bislang wurde in einer if-Anweisung genau eine Bedingung geprüft. In manchen Fällen ist es aber notwendig, dass mehrere Bedingungen mit einander verknüpft werden. Beispielsweise in einem Quiz mit einer Multiple-Choice-Frage die zwei richtige Antworten besitzt. Dort sollte die Antwort nur als richtig gewertet werden, wenn beide Antworten gegeben wurde. Dies lässt sich in Scratch über die Operatoren  und  realisieren.

5.3 while-Schleifen

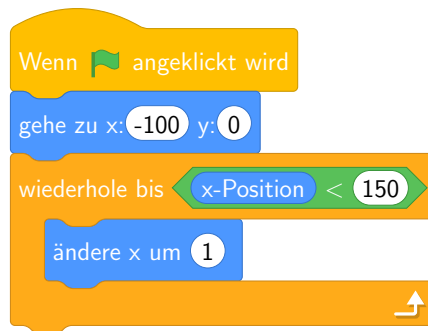
Mit if-Schleifen lässt man den Computer nur Tätigkeiten ausführen, wenn eine definierte Bedingung erfüllt ist. Mit Wiederholungsschleifen kannst du den Computer einen Befehl eine bestimmte Anzahl oft wiederholen lassen. Manchmal ist es wünschenswert, dass der Computer einen Befehl ausführt bist eine gewisse Bedingung erfüllt ist. Dafür gibt es die sogenannten *while*-Schleifen.

Die *while*-Schleife wiederholt eine Tätigkeit, solange die Bedingung erfüllt ist. **Vorsicht:** Wenn die Bedingung immer gilt, läuft die Schleife einfach ewig weiter.

While-Schleifen eignen sich besonders dann, wenn man vorher nicht weis, wie oft eine Schleife durchlaufen werden soll.

Beispiel 13:

Mit dieser *while*-Schleife lassen wir die Figur solange nach rechts laufen, bis die Grenze für die X-Koordinate(Das Ende der Bühne) erreicht ist .



5.4 Neue Befehle

Beim Programmieren braucht man die Möglichkeit, je nach Situation verschiedene Tätigkeiten auszuführen. Die Situation wird durch die Werte einer Variablen beschrieben. Der Computer muss in Abhängigkeit des Wertes einer Variablen eine Entscheidung treffen. Das ermöglichen die Anweisungen *if*, *else*, *while*.

In diesem Kapitel hast du folgende Befehle kennengelernt:

	Die Anweisungen im Körper zu <i>if</i> werden nur dann ausgeführt, wenn die Bedingung erfüllt ist. Zwei Bedingungen können mit den „und“ oder „oder“ Operatoren verknüpft werden.
	Es wird entweder der Codeblock 1 oder Codeblock 2 ausgeführt. Wenn die Bedingung erfüllt ist Codeblock 1, ansonsten Codeblock 2.
	Solange die Bedingung erfüllt ist, wird der Körper in der <i>while</i>-Schleife wiederholt. Nach jedem Durchlauf der Schleife prüft der Computer, ob die Bedingung erfüllt ist. Falls ja, wird die Schleife nochmals wiederholt.
	Diese <i>while</i>-Schleife läuft endlos. Bis das Programm über das Stop-Symbol beendet wird. Sehr wichtig bei der Steuerung von Figuren oder wenn Prozesse immer laufen sollen.

5.5 Teste dich selbst

Wissensfragen

1. Wie kann man erreichen, dass ein Programm einen Befehl nur unter gewissen Bedingungen ausführt?

2. Wie kann man einem Programm die Möglichkeit geben, sich für eine von zwei möglichen Alternativen zu entscheiden?

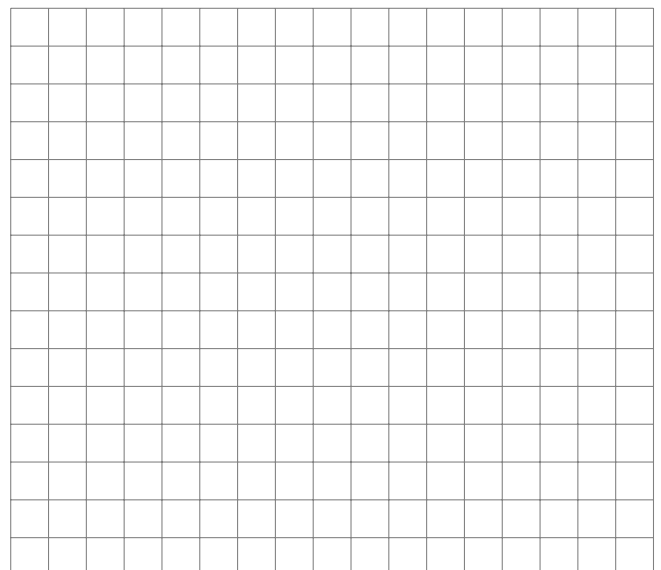
3. Was passiert, wenn die Bedingung einer *while*-Schleife erfüllt ist? Was passiert wenn sie nicht erfüllt ist? Was hat das mit der Bedeutung des englischen Wortes „while“ zu tun?

4. Ist die zusammengesetzte Bedingung  auch erfüllt, wenn der Wert von x gleich 4 ist.

Programmieraufgaben

Aufgabe 5.2:

Das folgende Programm fordert auf eine Zahl einzugeben. Was macht das Programm mit dieser Zahl? Was gibt das Programm aus, wenn man 10 eingibt. Stelle vor dem Programmieren zuerst eine Vermutung auf.



Aufgabe 5.3:

Entwickle ein kleines Quiz-Programm. Das Programm stellt zuerst eine Frage. Dann prüft es, ob die Antwort richtig oder falsch ist. Wenn die Antwort richtig ist, gibt das Programm „richtig“ aus, ansonsten „falsch“.

Schreibe das Programm so, dass es nacheinander mehrere verschiedene Fragen stellt (mindestens drei). Nach jeder Frage prüft, das Programm mit einer *if*-Anweisung, ob die Eingabe richtig ist. Erst danach kommt die nächste Frage.

Das Programm soll in einer Variablen mitzählen, wie viele richtige Antworten bisher gegeben wurden.

Optional: Gestalte drei Bühnenbilder für das Quiz, eines für die Fragen, eines für eine richtige und eines für eine falsche Antwort. Die Bühnenbilder können dann jeweils geändert werden.

6 Projekt

In diesem Kapitel sollen die Grundlagen für dein eigenes Scratch-Spiel erklärt werden und Inspirationen für eigene Ideen gegeben werden. Begonnen wird mit der grundlegenden Steuerung einer Figur mittels den Pfeiltasten. Im Anschluss wird die Arbeit mit Kostümen und Bühnenbildern erläutert.

6.1 Steuerung der Spielfigur

Unser Ziel ist es jetzt, die Katze mit den Pfeiltasten der Tastatur steuern zu können. Dazu benötigt man die if-Anweisungen aus dem vorherigen Kapitel.

Wiederholung: Mit der if-Anweisung () wird überprüft, ob eine Bedingung

erfüllt ist und falls ja werden die Befehle innerhalb der Anweisung ausgeführt.

Wir beginnen mit der Bewegung nach rechts. Wenn die „rechte Pfeiltaste“ gedrückt wird, soll sich die Katze nach ein kleines Stück nach rechts bewegen.

Um zu überprüfen, ob die „rechte Pfeiltaste“ gedrückt wurde, benötigen wir einen entsprechenden Sensorblock. diesen finden wir bei den Befehlen in der Kategorie „Fühlen“ (hellblau).

Wir wählen dort  aus. Die Tasten lassen sich im Drop-Down-Menü beliebig auswählen.

Ist die Taste gedrückt, soll sich die Katze nach rechts bewegen. Die entsprechende if-Anweisung sieht so aus:

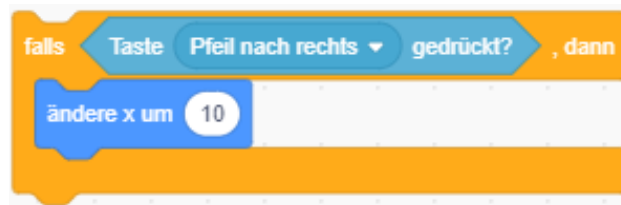


Abbildung 14: Einfache if-Schleife, ob die Pfeiltaste nach rechts gedrückt ist.

Führst du diese Anweisung aus, stellst du fest, dass die Figur sich nicht bewegt bzw. nur ein kleines Stück und du für jede Bewegung auf die Anweisung Doppelklicken musst.

Das liegt daran, dass Scratch nur ein einziges Mal überprüft, ob die Pfeiltaste gedrückt ist oder nicht und beendet danach das Programm. Damit Scratch ab dem Programmstart dauerhaft prüft, ob die Pfeiltaste gedrückt ist, muss die if-Anweisung von einer Schleife („wiederhole fortlaufend“) umgeben werden und am Anfang des Blocks ein  stehen. Unser fertiges Programm sieht dann folgendermaßen aus:

Stehen. Unser fertiges Programm sieht dann folgendermaßen aus:



Abbildung 15: Bewegungen der Katze nach rechts mit der Pfeiltaste.

Der Block  stellt sicher, dass die Figur nach dem Start auch sichtbar ist.

Aufgabe 6.1:

Ergänze das Programm aus Abbildung 15 so, dass die Katze sich mit den Pfeiltasten über den Bildschirm in jede Richtung steuern lässt.

Tipp: Dupliziere die *if*-Anweisung innerhalb der Dauerschleife und passe die Tasten und Richtungen an. Teste, ob deine Katze in die jeweils richtige Richtung sich bewegt und was passiert, wenn zwei Pfeiltasten gleichzeitig gedrückt werden (Dann sollte sie schräg laufen).

6.2 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 Bild/Kostüm, auf dem sie nach links blickt.

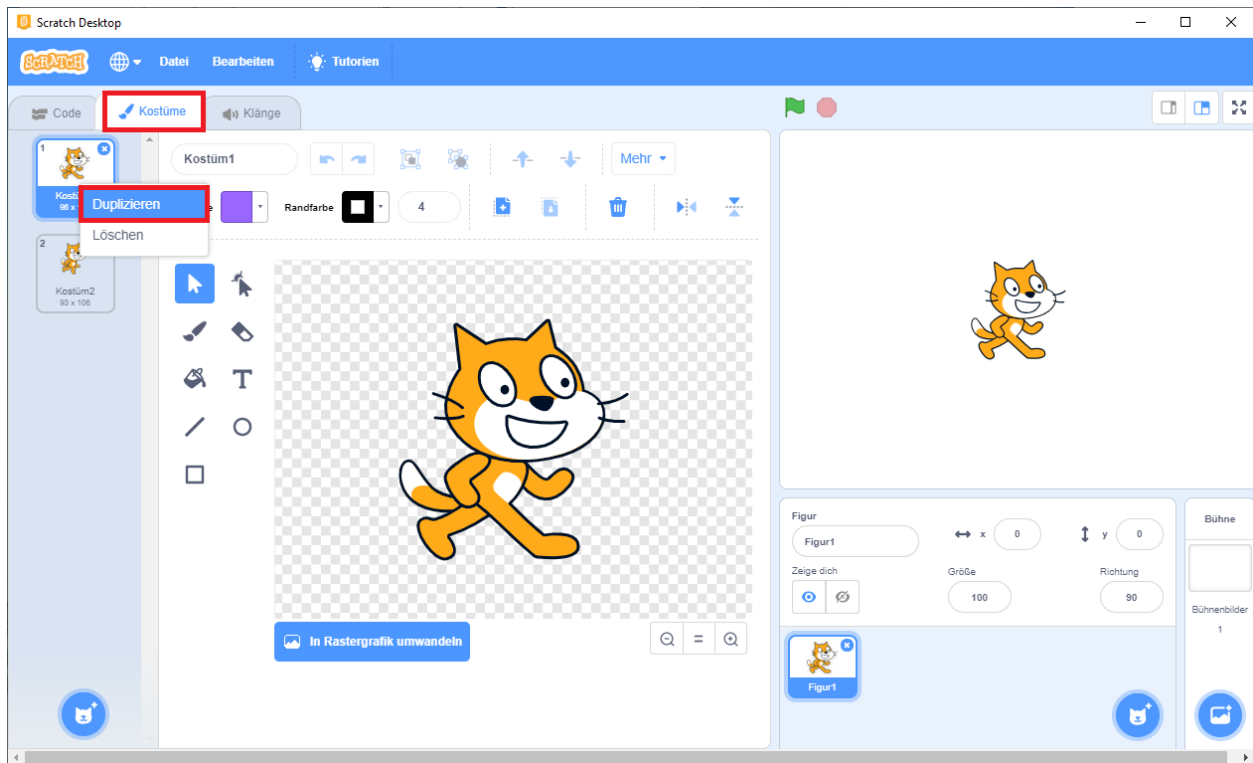


Abbildung 16: Kostümeditoransicht.

Im Reiter “Kostüme” (siehe Abb. 16) siehst du alle Kostüme einer Figur und kannst ihr neue anlegen. Die Katze besitzt bereits zwei Kostüme. Das aktuell zum Bearbeiten ausgewählte Kostüm ist immer blau umrandet. Das zweite Kostüm benötigen wir nicht, daher wählen wir es aus und löschen es mit einem Klick auf dem Mülleimer.

Um ein weiteres Kostüm zu erstellen, gibt es mehrere Möglichkeiten. Man kann über das Menü unten links Kostüme hochladen, selber malen oder aus bestehenden eines Auswählen. Zu Anfang nutzen wir das bereits bestehende Kostüm indem wir es duplizieren und dann verändern. Klicke mit der rechten Maustaste auf “Kostüm 1” und wähle “Duplizieren” aus. Klicke auf den Icon für „Links und Rechts vertauschen“ und gib dann dem Kostüm einen sinnvollen Namen.

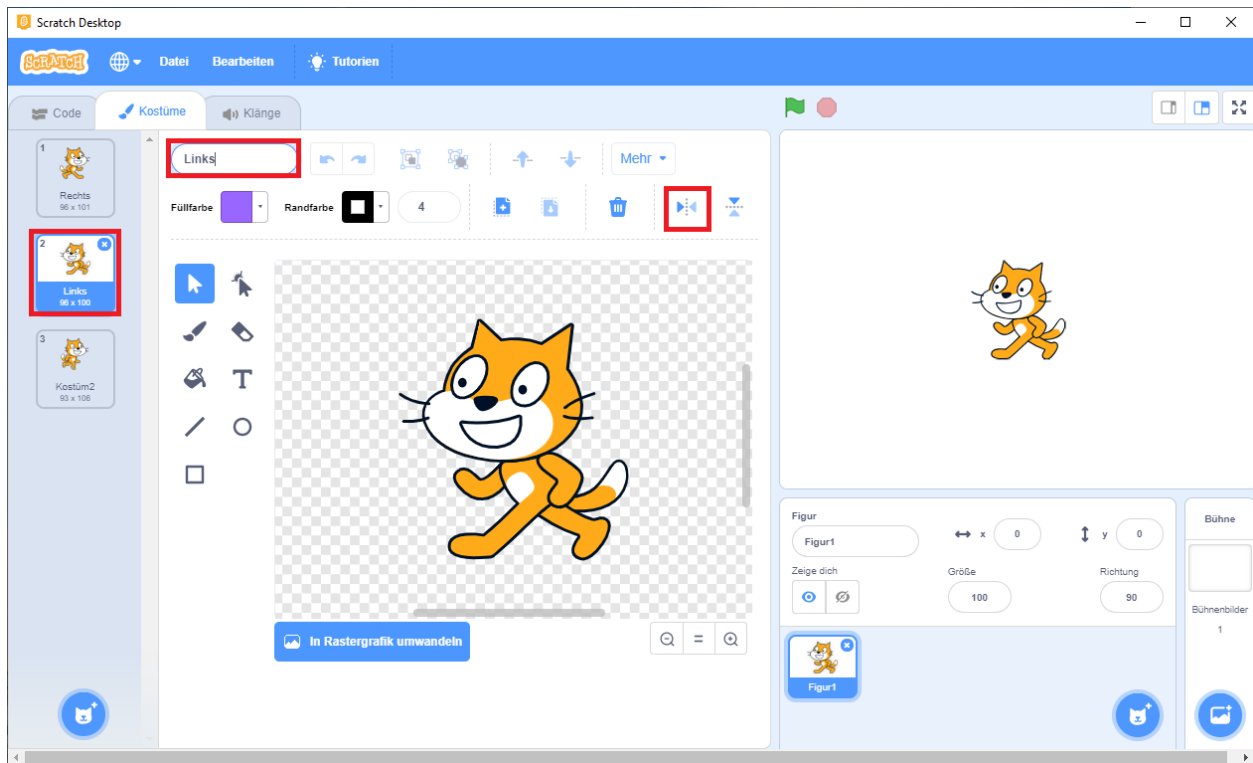


Abbildung 17: Kostüm bearbeiten b.

Verändere auch den Namen für das normale Kostüm in Katze rechts.

Wechsle wieder in den Programmierbereich Code.

Die Blöcke für den Kostümwechsel findest du in der Blockpalette unter „Aussehen“

Aufgabe 6.2:

- 1 Verändere die Tastaturereignisse so, dass der Katze jeweils vor dem Loslaufen das richtige Kostüm angezogen wird.
- 2 Teste dein Programm und lasse die Katze in alle Richtungen über die Bühne laufen.
- 3 Speichere alles ab.

6.3 Die Katze läuft über die Straße

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 6.3:

1. Klicke zunächst auf die Schaltfläche „In Rastergrafik umwandeln“.
2. Erkunde zunächst den Zeicheneditor.
3. Zeichne eine Straße auf das Bühnenbild und färbe die Umgebung grün.
4. Gib dem Bühnenbild einen neuen Namen: Level1
5. Ändere anschließend die Größe der Katze auf 50

Dein Bühnenbild könnte etwa so aussehen:

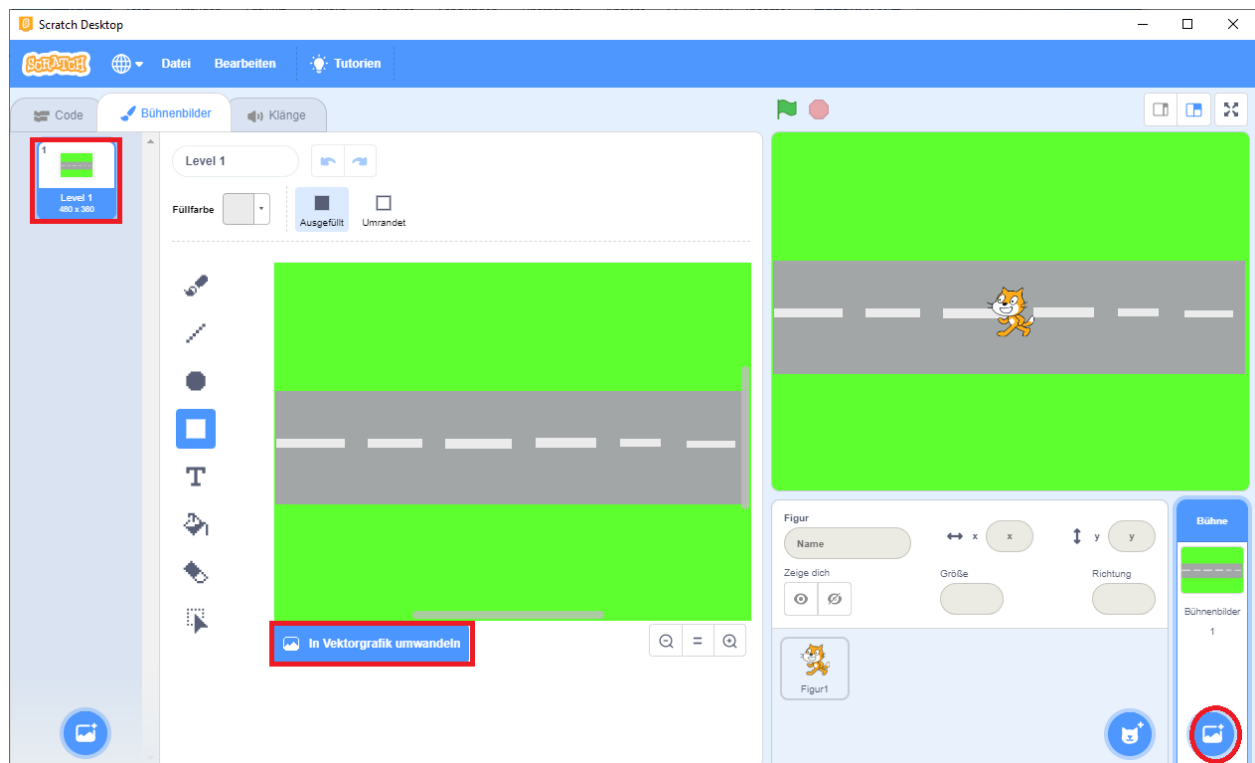


Abbildung 18: Kostüm Zeicheneditor

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? Du kannst dir selbst ein Auto zeichnen
oder ein Bild aus einer Bildbibliothek wählen.