Werkgymnasium Heidenheim

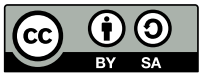
Informatik 7

Scratch-Projekt

Dieses Skript gehört:

Autor:
Fabian Drechsler

4. Dezember 2025



Inhaltsverzeichnis

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 Aufbaukurs Informatik Klasse 7 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 für die Bedienung der Oberfläche an.
Web-App: `https://scratch.mit.edu/projects/editor`
Offline-Editor:`https://scratch.mit.edu/download`

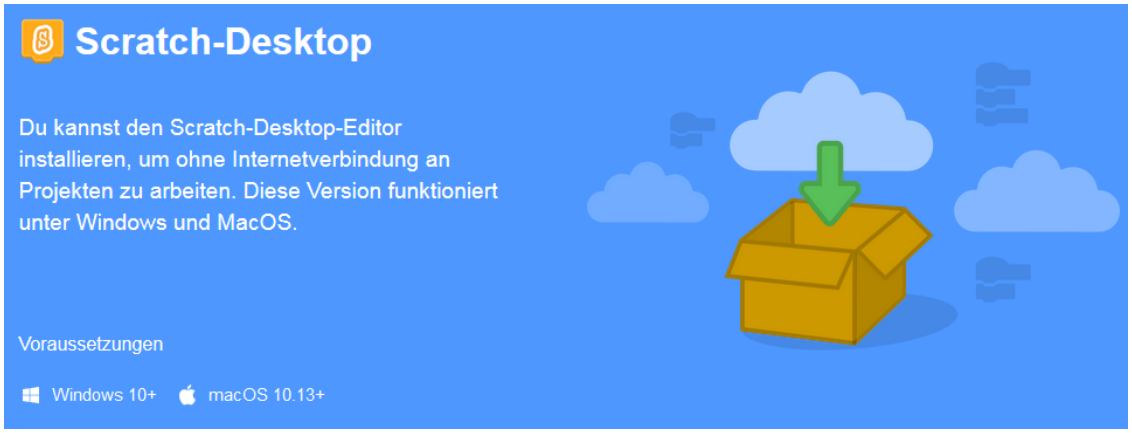


Abbildung 1: Downloadseite des Offline-Editors

Wissen: Programmiersprachen

Programmiersprachen bestehen, genauso wie gesprochene Sprachen, aus Wörtern, die eine bestimmte Bedeutung haben. Man verwendet eine Programmiersprache, um einem Computer Anweisungen/Befehle zu geben, die der Computer dann ausführt.

Wissen: Programme und Algorithmen

Ein Programm besteht aus einer Reihe von Befehlen, die hintereinander ausgeführt werden. Ein Algorithmus ist wie ein Kochrezept für den Computer – eine klare Schritt-für-Schritt-Anleitung, die exakt angibt, was getan werden muss, um ein Problem zu lösen.

Wissen: Programmieren

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. Mann 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 in Scratch kennen.

2 Die ersten Programme

2.1 Scratch-Oberfläche

Die Oberflächen des Online- und Offline-Editors unterscheiden sich nicht, dadurch kann problemlos zwischen beiden gewechselt werden.

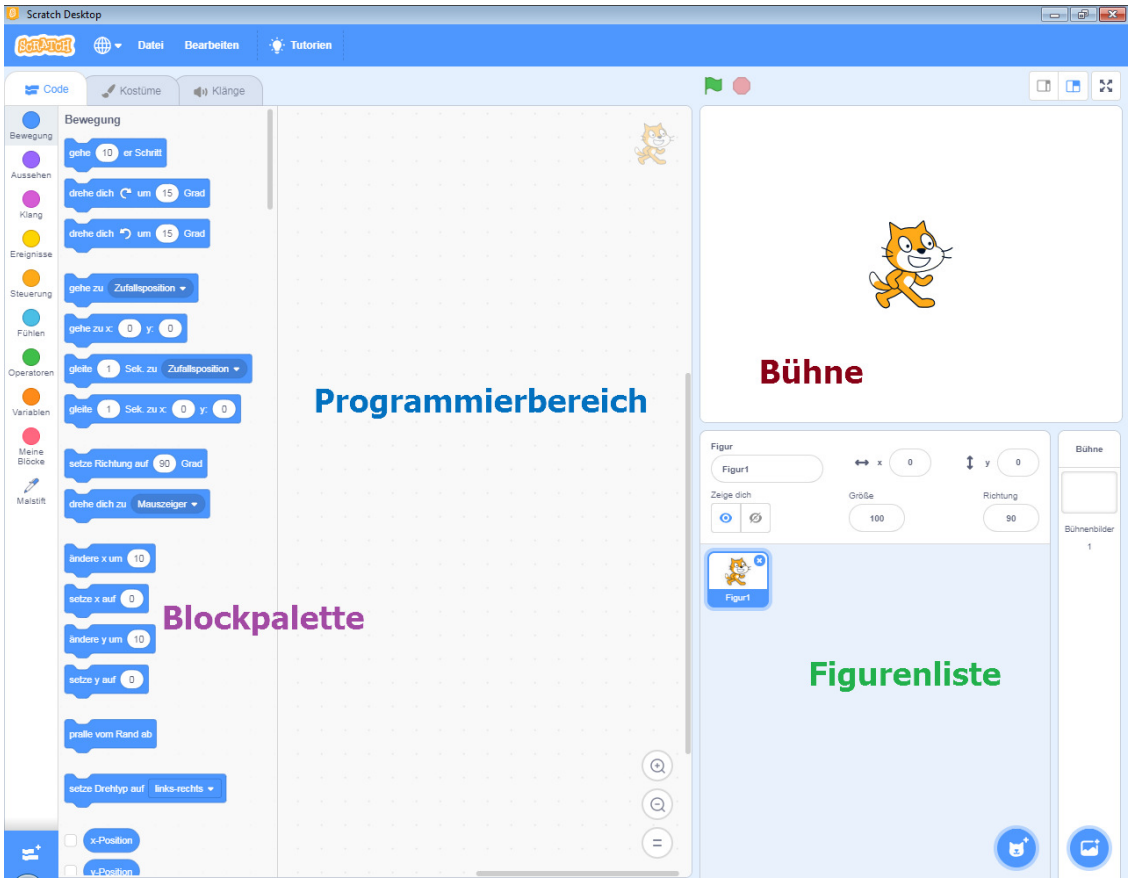


Abbildung 2: Scratch-Oberfläche

Die Scratch-Oberfläche (siehe Abb. 2) teilt sich in vier Bereiche auf.

Bühne:

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

Figurenliste:

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 (z.B. Größe) festlegen. Außerdem findest du dort auf der rechten Seite den Hintergrund

deiner Bühne, die Bühnenbilder. Über den blauen Knopf in der ganz rechten unteren Ecke kannst du neue Bühnebilder hinzufügen.

Blockpalette:

In der **Blockpalette** findest du die Blöcke, die du zum Programmieren brauchst. Du kannst ihre Funktionen dort direkt durch einen Doppelklick testen. Ähnlich wie LEGO-Steine werden die Blöcke im **Programmierbereich** zu einem Programm zusammengefügt.

Programmierbereich:

Ähnlich wie LEGO-Steine werden die Blöcke im **Programmierbereich** zu einem Programm zusammengefügt. Ziehe sie dazu mit der Maus aus der **Blockpalette** in den **Programmierbereich**.
Du kannst ein Programm durch einen Doppelklick auf den Programmblock ausführen oder, falls du den entsprechenden Block an den Anfang des Programms gesetzt hast, mit der grünen Fahne.

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

Aufgabe 2.1
Du lernst nun, wie du eine neue Figur in deiner Programmierumgebung erstellst. Führe dafür folgende Schritte der Reihe nach aus.

1. Klicke auf den „Katzenkopf“ in der unteren rechten Ecke.
2. Suche nach der Figur mit dem Namen „Beetle“.
3. Füge die Figur durch einen Klick auf diese ein.
4. Klicke die Standardfigur mit dem Namen „Figur1“ einmal und lösche sie mit einem Klick auf das Mülleimersymbol.

Verwende für deine Programmieraufgabe in Zukunft die neue Beetle-Figur. Diese hat den Vorteil, dass du die Blickrichtung der Figur leicht erkennen kannst.

2.2 Neue Befehle

Zur Steuerung deiner Figur auf der Bühne gibt einige grundlegende Befehle. Diese findest du in Tabelle 1.
Die meisten Befehlsblöcke sind durch ihre Beschriftung selbsterklärend. Anonsten kannst du die Funktion herausfinden, indem du den Block in den Programmierbereich ziehst, ausführst und beobachtest, was mit deiner Figur auf der Bühne passiert.

Aufgabe 2.2
Finde heraus, was die Blöcke in der Tabelle bewirken. Tipp: Ziehe deine Figur, bevor du einen Block testest, immer auf eine zufällige Position auf der Bühne.

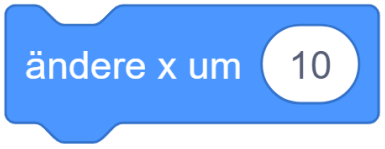
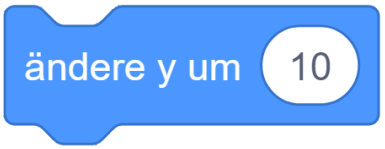
	
	
	
	
	
	
	

Tabelle 1: Bewegung Grundbefehle

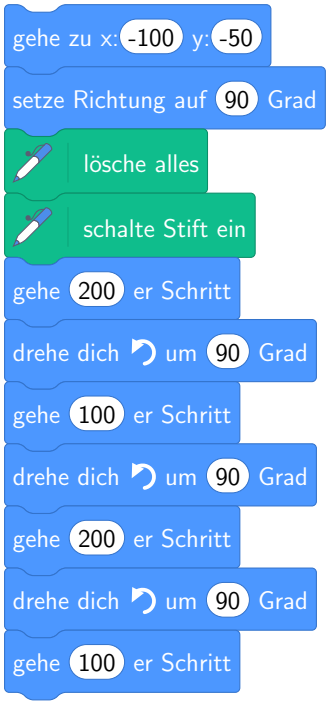
2.3 Erweiterungen

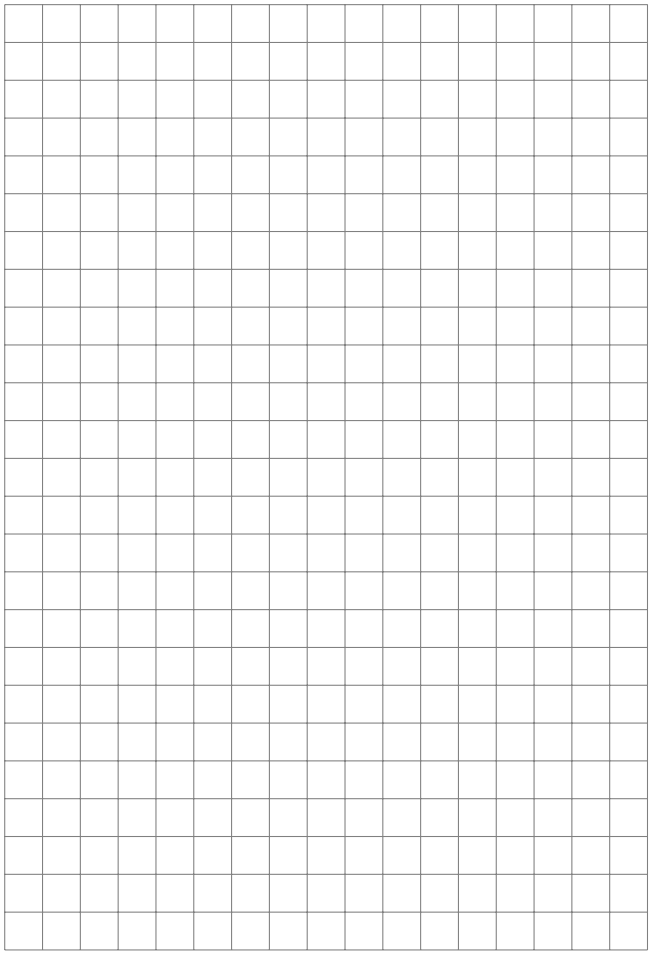
Scratch bietet eine Vielzahl an Erweiterungsmöglichkeiten für die verschiedensten Anwendungsfälle.
Die Übersicht der verfügbaren Erweiterungen kann man über einen Klick auf das blaue Symbol in der linken unteren Ecke (siehe Abb. 2) öffnen.
Für uns ist die Erweiterung *Malstift* unverzichtbar. Damit kannst du die Auführung von Befehlen (und eventuelle Fehler im Programm) leichter nachvollziehen, weil die Figur ihre Bewegungen auf die Bühne zeichnet.

Aufgabe 2.3
Füge deiner Blockpalette die Erweiterung ***Malstift*** hinzu.

Beispiel 1

Überlege dir, welche Funktion die einzelnen Codeblöcke haben und notiere sie daneben. Skizziere neben dem Codeblock was die Figur bei der Ausführung des Programms auf die Bühne zeichnet.





Überprüfe deine Hypothese, indem du den Codeblock programmierst und anschließend ausführst (Doppelklick auf den Codeblock).

Speichere jede der folgenden Aufgaben in einer eigenen Datei ab. Der Dateiname soll die Nummer der Aufgabe sein (Bsp: "Aufgabe 2.4").

Aufgabe 2.4

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

Aufgabe 2.5

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

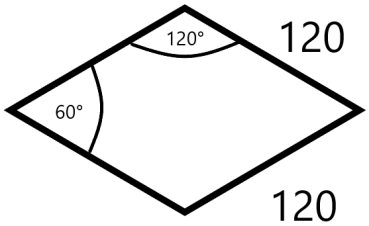
1. Skizziere hier den Weg, den deine Figur laufen soll und an welchen Stellen sie sich drehen muss.

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.)

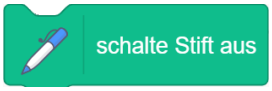
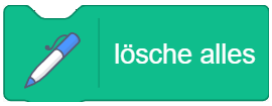
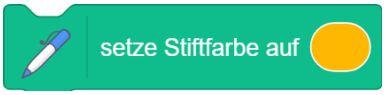
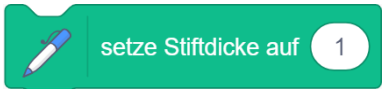
Aufgabe 2.6

Schreibe ein Programm, dass die abgebildete Skizze zeichnet.
Lege als erstes einen geeigneten Startpunkt fest und die Blickrichtung fest.
(Tipp: Du kannst in die Skizze unten mit Bleistift Eintragungen vornehmen.)

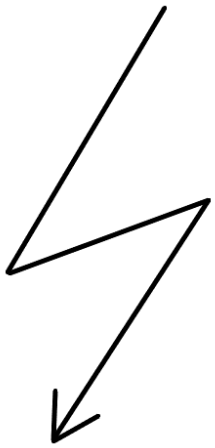


Farben

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

	Schaltet den Malstift ein.
	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.
	Löscht alle bisherigen Malspuren, z.B. nachdem die Figur an ihrer Startposition angekommen ist.
	Legt die Farbe des Stifts fest. Ein Klick in das Farbfeld öffnet den Farbeditor.
	Setzt die Strichstärke auf die gewünschte Größe.

Aufgabe 2.7
Schreibe ein Programm, das einen Blitz (siehe Skizze) zeichnet. Wähle selbst die Länge der Linien und der Winkel.
Bonus: Ändere die Farbe und Dicke der Linie so, dass es tatsächlich wie ein Blitz aussieht.



Beispiel 2
Im nächsten Schritt soll die Figur ein regelmäßiges Sechseck wie in Abbildung 3 zeichnen.

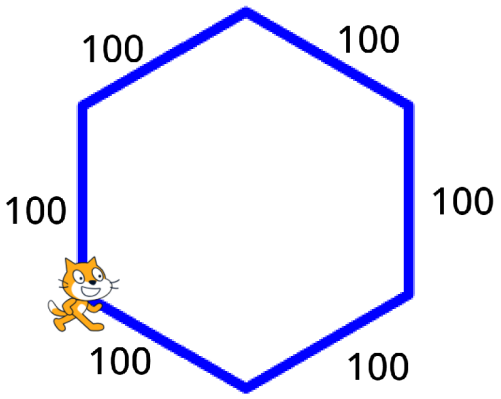


Abbildung 3: Sechseck

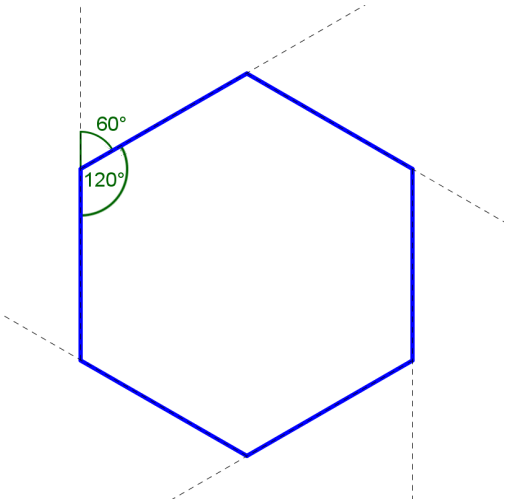


Abbildung 4: Sechseck mit Winkeln

Vor der Programmierung des Skripts müssen wir festlegen, an welcher Stelle unsere Figur mit dem Zeichnen beginnt. In Abb.3 sitzt die Figur bereits an entsprechender Stelle.
Als nächstes überlegen wir, um wie viel Grad sich die Figur 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 Figur 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 $\frac{360°}{6} = 60°$. 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 2.8
Schreibe ein Programm, das ein regelmäßiges Fünfeck mit Seitenlänge 110 zeichnet. Gehe wie in Beispiel 2 vor (Skizze auf Papier, Winkel einzeichnen ...).

2.4 Programme mit Rechnungen

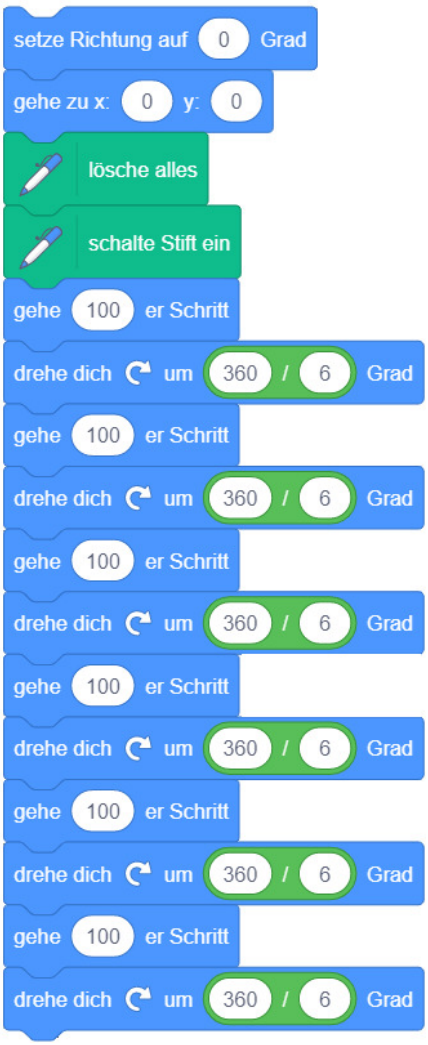
Computer können mathematische Berechnungen sehr schnell durchführen, weshalb sie auch "Rechner"genannt werden. Rechnungen sind in Scratch im Block *Operatoren* untergebracht.

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 $\frac{360°}{6}$. Wir müssen dies nicht selbst berechnen sondern können es das Programm überlassen. Unser Beispiel sieht jetzt aus wie nebenan.



Aufgabe 2.9
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 Hypothenuse y (lange Seite) lässt sich mit dem Satz des Pythagoras ("Die Summe der Kathetenquadrate ist gleich dem Quadrat der Hypothenuse") berechnen. Du lernst ihn in Klasse 9 im Fach Mathematik kennen. Für jetzt reicht das Ergebnis: $y \approx x \cdot 1,414214$.

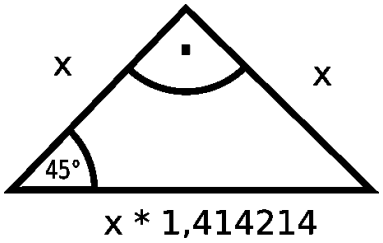
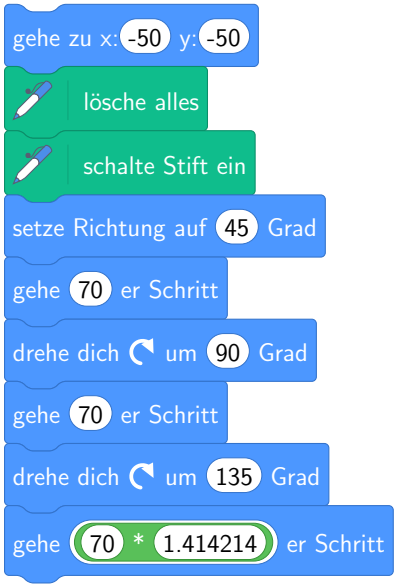


Abbildung 6: Rechtwinkliges Dreieck

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



Aufgabe 2.10
Schreibe ein Programm, welches mit Scratch das Haus des Nikolaus (sie 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.

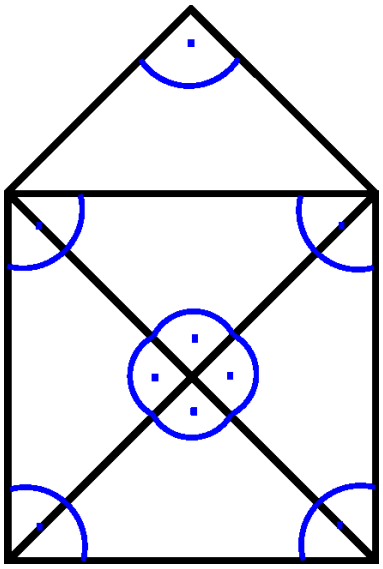


Abbildung 7: Schema Haus des Nikolaus

Zeichne die in das Feld einmal das Haus des Nikolaus, so wie du es zeichnen würdest. Achte darauf, dass keine Seite doppelt gezeichnet werden darf! Zeichne nun die acht Zeichenschritte als einzelne Bilder nebeneinander, damit hast du einen leichteren Überblick über die einzelnen Programmierschritte.
(Tipp: Füge nach jeder gezeichneten Seite einen Warteblock mit 1 Sekunde ein)

2.5 Teste dich selbst

Wissensfragen

1. Was haben Computer mit Automatisierung zu tun?

2. Ist eine Programmiersprache eine Sprache?

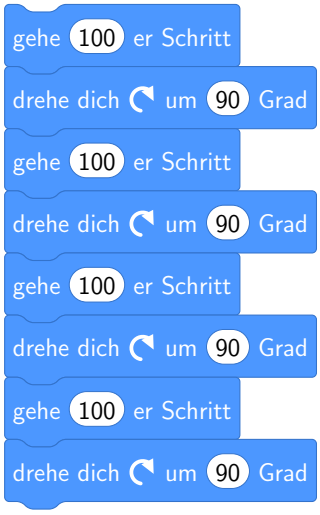
3. Wozu dienen Programmiersprachen?

4. Was ist ein Programm?

5. Was haben Programme mit einer Programmiersprache zu tun?

6. Welche Befehle hast du in Scratch kennen gelernt? Nenne 5.

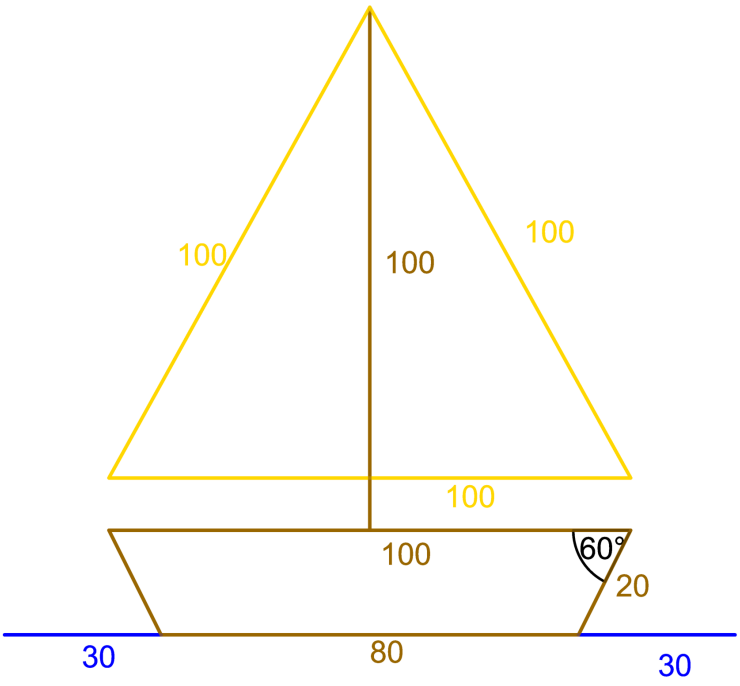
7. Beschreibe kurz den Ablauf des Codeausschnitts und zeichne das Bild dazu.



Programmieraufgaben

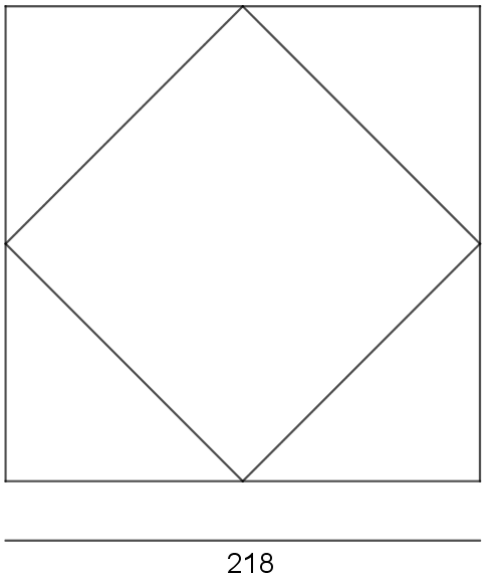
Aufgabe 2.11

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



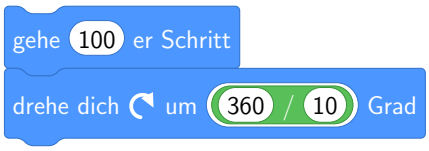
Aufgabe 2.12

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 dies Seiten des kleinen Quadrats. Schreibe ein Programm, dass diese Figur mir den zwei Quadraten zeichnet. Das große Quadrat soll eine Seitenlänge von 218 haben und die Figur soll sich am Ende verstecken.



3 Vorgänge wiederholen - Einfache Schleifen

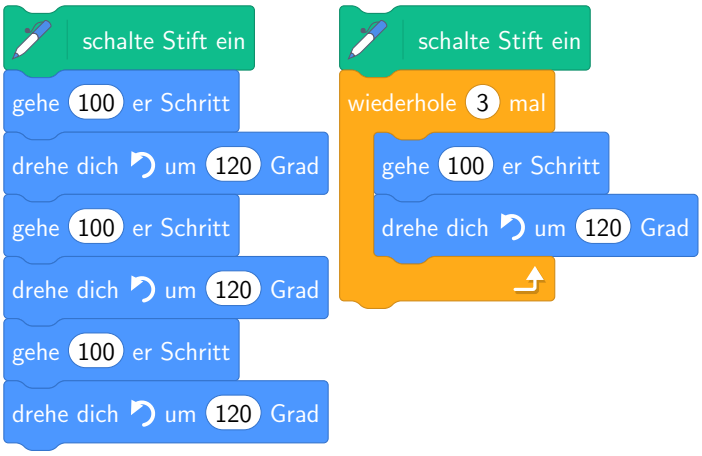
In unserem Alltag kommen immer wieder Routinetätigkeiten vor die einfach und langweilig sind. Mit Hilfe der Informatik kann man solche langweiligen Tätigkeiten automatisieren. Computer können immer mehr und komplexere Aufgaben übernehmen und führen zu einer Entlastung der Menschen und bieten die Möglichkeit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

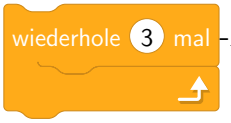


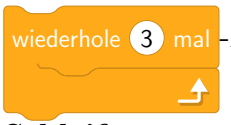
wiederholen. Dies muss einfacher möglich sein!

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:



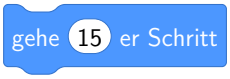
Info: Zu jeder 

-Anweisung gehören die Anzahl der Wiederholungen und eine Folge von Befehlen, die wiederholt werden sollen. 

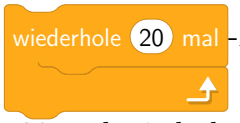
-Anweisung zusammen mit den zu wiederholenden Befehlen nennt man eine **Schleife**.
Bei Programmieren beinhaltet eine Schleife immer einen Programmteil, der mehrfach ausgeführt wird. Dieser Programmteil heißt **Körper der Schleife**

.

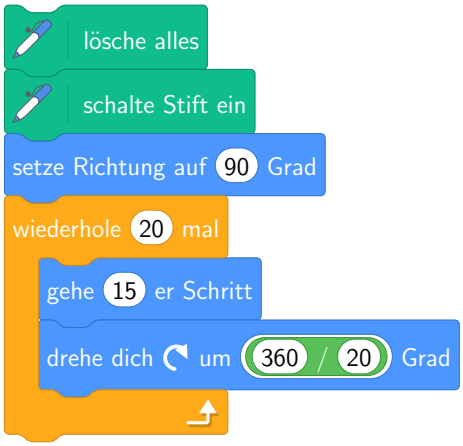
Beispiel 7

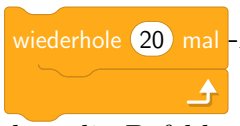
Um ein 20-Eck zu zeichnen, müsstest du die Befehle 

und 

20-mal schreiben. Weil wir aber die 

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

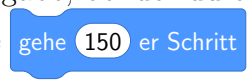


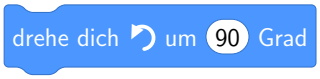
Die **Schleife** besteht aus der 

-Anweisung mit der Anzahl der Wiederholungen. Davon eingeschlossen stehen die Befehle, die eine bestimmte Anzahl Male ausgeführt werden sollen. Die Befehle 

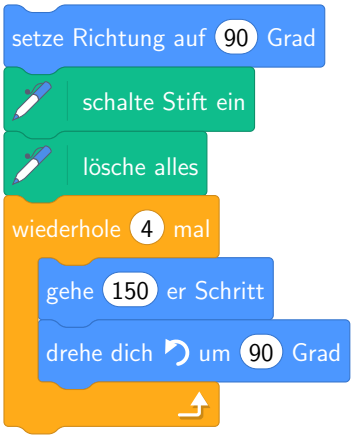
und 

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 

und 

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.

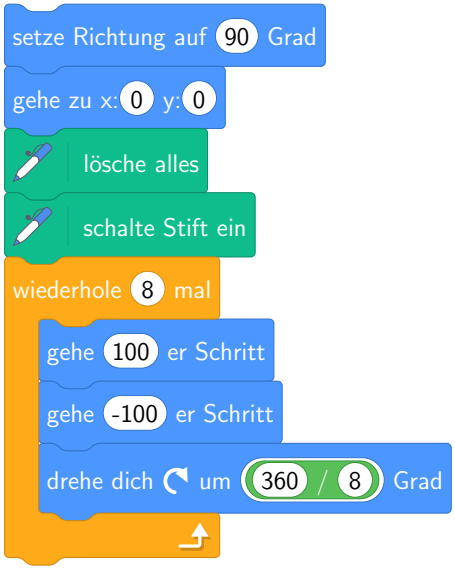
Aufgabe 3.1
Schreibe jeweils ein Programm, um ein regelmäßiges Vieleck mit der geforderten Anzahl der Ecken und der geforderten Seitenlänge zu Zeichnen.
(Tipp: Starte bei den Koordinaten $x : 0$ $y : 120$ und verwende vor der Schleife jeweils den gleichen Block, wie in Beispiel 8. Die verschiedenen Programmblöcke 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%.)



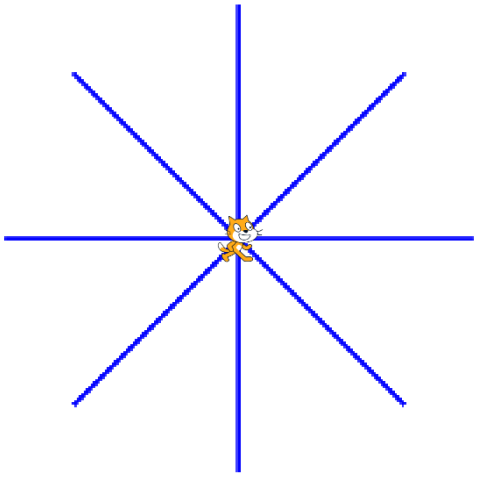


Abbildung 8: Stern mit 8 Spitzen

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.

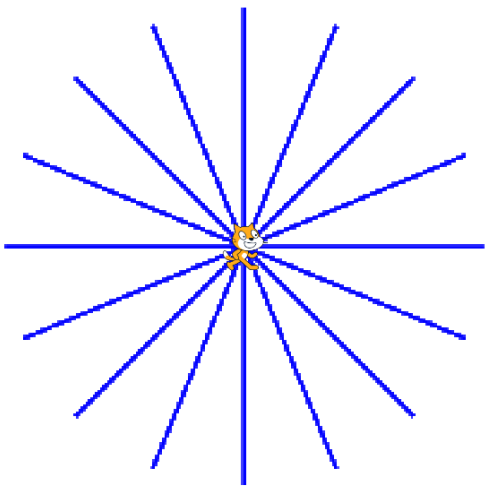


Abbildung 9: Stern mit 16 Zacken.

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

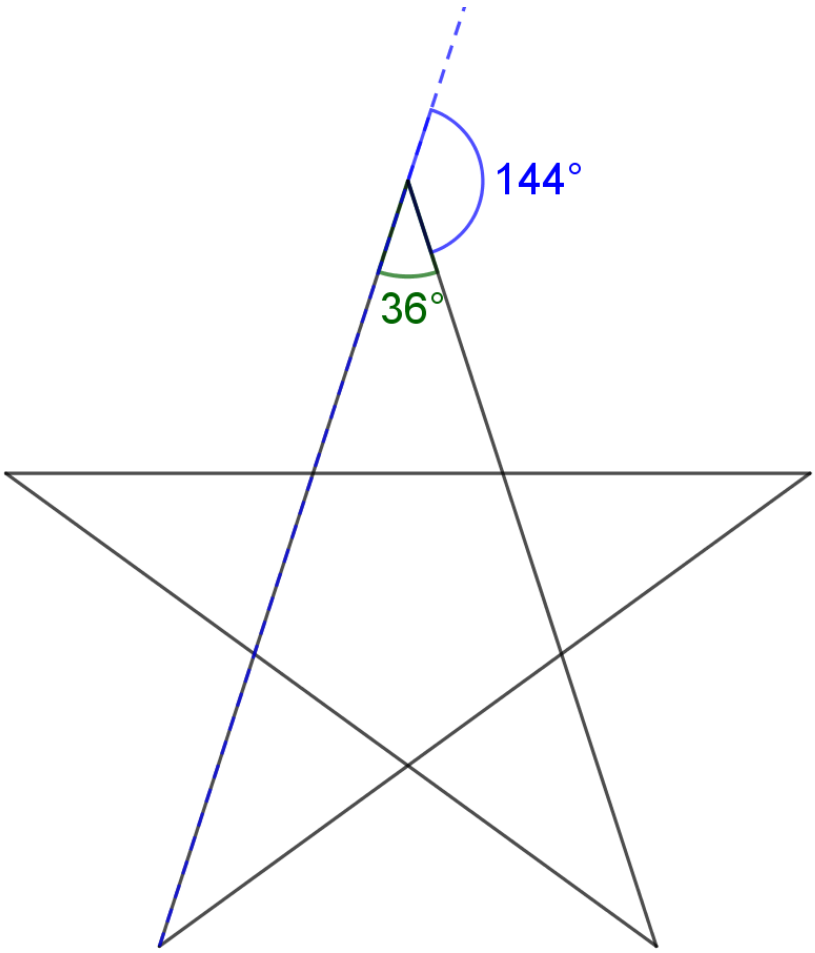


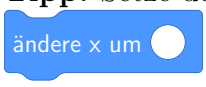
Abbildung 10: Pentagramm

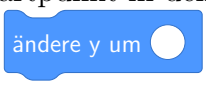
Aufgabe 3.4
Schreibe drei Programme (bzw. Programmblöcke) und speichere alle in einer Datei, die eine Treppe wie in Abbildung 11(b) zeichnen. Die Höhe und Länge der einzelnen Treppenstufen (siehe Abbildung 11(a)) wählst du wie folgt:

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 

und 

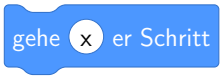
(a) (b)

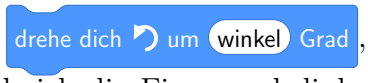
Abbildung 11: Darstellung der Stufen

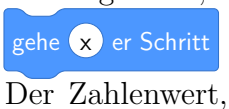
4 Parameter und Variablen

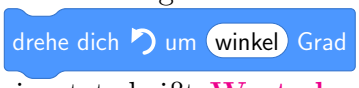
In diesem Kapitel lernst du, was ein **Parameter** ist, **Variablen** zu deklarieren und zu nutzen.

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 

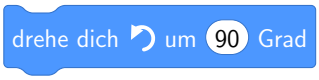
nennt man einen **Parameter**. Der Zahlenwert, den man dabei einsetzt, heißt **Wert des Parameters**. Die Werte eines Parameters dürfen auch andere Objekte als Zahlen sein: Ein Parameter kann für eine Farbe stehen, dann ist der Wert des Parameter ein Farbname.

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.

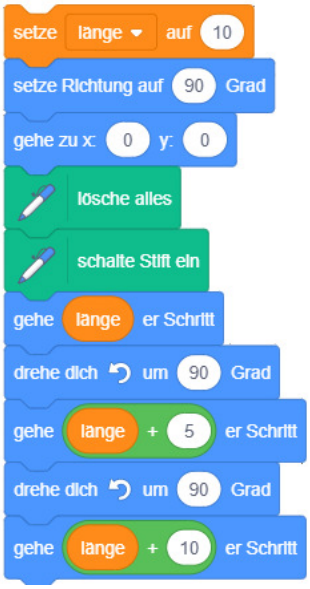
Beispiel 10
Wir möchten eine Spirale zeichnen. Die Figur soll jede Linie ein wenig länger als die vorherige

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, dass 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:

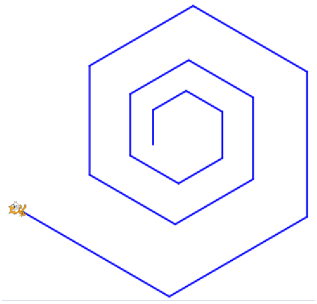


Abbildung 12: Lösung Aufgabe sechseckige Spirale

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.

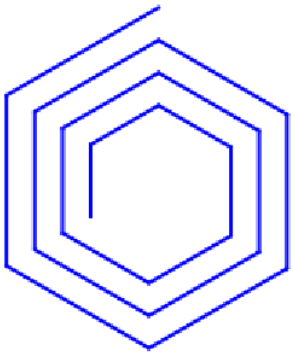
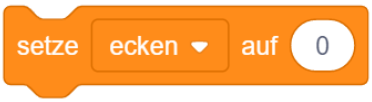


Abbildung 13: Schema Spirale

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?

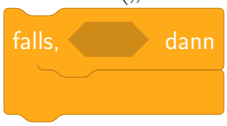
4. Woher weiß der Computer, welchen Wert eine Variable hat? Wie kann sich der Computer die Werte aller Variablen merken?

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?

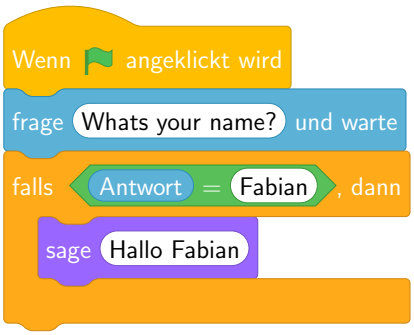
5 Verzweigungen und bedingte Schleifen

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 

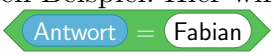
zur Verfügung.

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.

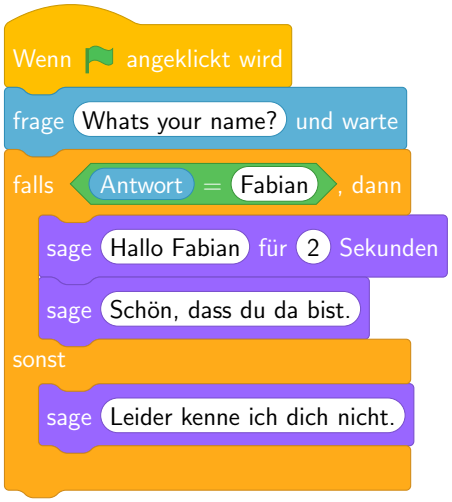


Aufgabe 5.1
Schreibe einen Begrüßungsprogramm für dich selbst (Es darf gerne komplexer sein.)

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 

nicht erfüllt ist.



5.2 EXKURS: Logische Operatoren

Mir der if-Anweisung (

) 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:

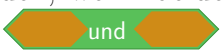
Behauptung A	**Behauptung B**
gilt	gilt
gilt	gilt nicht
gilt nicht	gilt
gilt nicht	gilt nicht

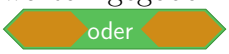
Verknüpfung
$\longrightarrow$

A oder B	**A und B**
gilt	gilt
gilt	gilt nicht
gilt	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.

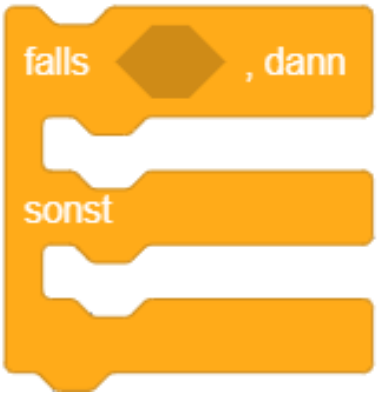
5.3 while-Schleifen

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 *if* 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 *while*-Schleife wiederholt. Nach jedem Durchlauf der Schleife prüft der Computer, ob die Bedingung erfüllt ist. Falls ja, wird die Schleife nochmals wiederholt.
	Diese *while*-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.

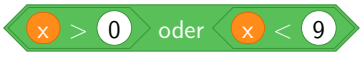
5.5 Teste dich selbst

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?

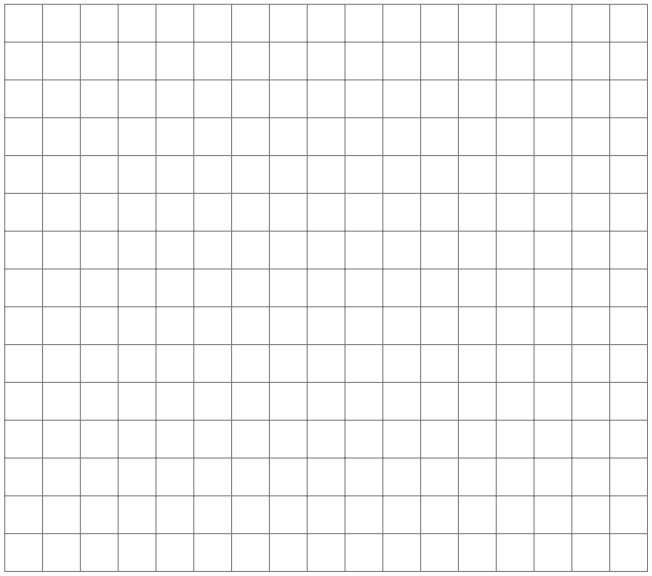
3. Was passiert, wenn die Bedingung einer *while*-Schleife erfüllt ist? Was passiert wenn sie nicht erfüllt ist? Was hat das mit der Bedeutung des englischen Wortes „while“ zu tun?

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.