

<b>Einleitung</b>	6
<b>A. Die Scratch-Umgebung</b>	6
<b>A.1. Das Scratch Programmfenster</b>	6
<b>A.2. Hauptelemente eines Scratchprojektes</b>	7
A.2.1 Die Bühne	7
A.2.2 Sprites	7
A.2.3 Befehle	7
A.2.4 Programme	8
<b>A.3. Sprites bearbeiten</b>	10
<b>A.4. Sprites haben Kostüme</b>	12
<b>A.5. Die Bühne</b>	14
<b>A.6. Projekt dokumentieren</b>	15
<b>B. Scratch-Projekte mittels Sequenzen entwickeln</b>	16
<b>C. Auf Ereignisse reagieren</b>	18
<b>D. Die Endlosschleife</b>	20
<b>E. Entscheidungen treffen</b>	22
<b>F. Zwischenprojekt</b>	26
<b>G. Bedingte Schleifen</b>	27
<b>H. Nachrichten senden und empfangen</b>	29
<b>I. Steuerung anhand einer Schleife</b>	32
<b>I.1. Benötigte Blockpaletten</b>	33
<b>I.2. Zeichnen eines Quadrats</b>	33
<b>I.3. Zeichnen eines ersten Mandalas</b>	34
<b>I.4. Weiterführende Aufgaben</b>	36
<b>I.5. Einige einfache, zusammengesetzte Mandalas</b>	37
<b>J. Zwischenprojekt</b>	38
<b>K. Variablen und mathematische Ausdrücke</b>	39
<b>K.1. Variablen</b>	39
K.1.1 Variablen erzeugen und anzeigen	39
K.1.2 Mit Variablen arbeiten	40
K.1.3 Vordefinierte Variablen	43
<b>K.2. Ausdrücke</b>	43
K.2.2 Wahr oder falsch	45
K.2.3 Für Fortgeschrittene: Zusammengesetzte Bedingungen	48

## Einleitung

Scratch ist eine neue Programmiersprache, die es leicht macht, die Programmierung eines Computers zu erlernen. Mit ihr kannst du kreative interaktive Geschichten, Spiele und Animationen erstellen.

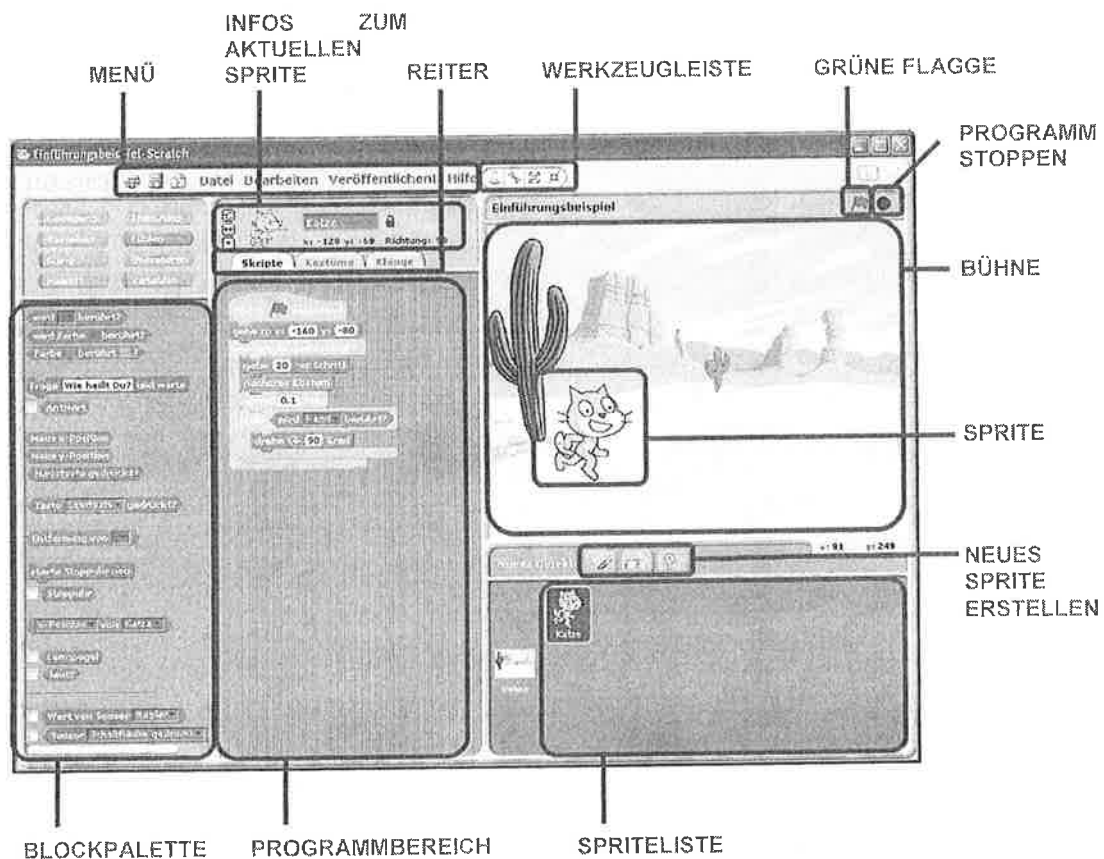
Das Programm Scratch kannst du gratis auf der Webseite [scratch.mit.edu](http://scratch.mit.edu) herunterladen. Auf dieser Webseite findest du auch eine Fülle von Informationen über Scratch (Handbücher, Videoanleitungen, Scratchkarten, Frequently Asked Questions (FAQs) usw). Du kannst dich auch auf der Scratchseite anmelden und dort deine Projekte veröffentlichen.

Dieses Skript führt dich schrittweise in die Programmierung mit Scratch ein.

## A. Die Scratch-Umgebung

In diesem Kapitel lernst du die Umgebung kennen mit der du Scratch-Projekte erstellen kannst.

### A.1. Das Scratch Programmfenster





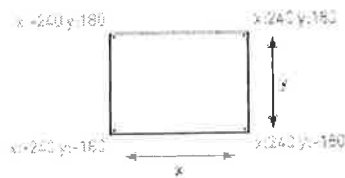
## A.2. Hauptelemente eines Scratchprojektes

### A.2.1 Die Bühne

Die Bühne ist der Platz, wo Geschichten, Spiele und Animationen zum Leben erwachen.

Sprites bewegen sich und interagieren miteinander auf der Bühne.

Die Bühne ist 480 Einheiten breit und 360 Einheiten hoch. Sie ist in einem XY-Koordinatensystem eingeteilt. Der Mittelpunkt der Bühne hat eine X-Koordinate von 0 und eine Y-Koordinate von 0.



### A.2.2 Sprites

Scratch Projekte setzen sich aus Objekten, so genannten Sprites (A.d.Ü.: ein Sprite ist wortwörtlich ein Kobold, bedeutet aber hier einfach eine Computergrafik) zusammen. Sprites werden mitunter auch **Objekte** genannt.

Du kannst das Aussehen eines Sprites verändern und zum Beispiel ein Sprite wie eine Person aussehen lassen, wie einen Zug, einen Schmetterling oder irgendetwas anderes.

Sprites haben einen Namen. Scratch gibt den Sprites automatisch Namen wie Objekt1, Objekt2, usw. Um dich besser in deinem Projekt zurechtzufinden sollst du den Sprites aussagekräftige Namen geben.



### A.2.3 Befehle

Du kannst den Sprite Befehle ausführen lassen, ihm sagen sich zu bewegen, Musik abzuspielen oder ihn mit anderen Sprites interagieren lassen. Um dem Sprite die gewünschten **Befehle** mitzuteilen, musst du graphische **Blöcke** in den Programmbereich ziehen.

Diese Befehlsblöcke befinden sich in den Blockpaletten. Ein Befehlsblock hat, je nach Art des Befehls, eine andere Farbe und befindet sich in einer anderen Blockpalette. So sind beispielsweise alle Befehle die einen Sprite bewegen dunkelblau und befinden sich in der Palette **Bewegung**.

Hier die verschiedenen Blockpaletten :





Außerdem unterscheiden sich die Befehlsblöcke durch ihre Form.

### A.2.4 Programme

Ein Programm ist eine Folge von Befehlen welche nacheinander ausgeführt werden. In Scratch erstellst du ein Programm indem du die benötigten Befehlsblöcke, ähnlich wie bei einem Puzzle, zusammenfügst.

Ein Sprite kann mehrere Programme enthalten, welche auch als Skript(e) bezeichnet werden.

Wenn du ein Programm doppelklickst, führt Scratch die Blöcke von oben bis unten aus.

Um einen Befehlsblock, welchen du nicht mehr benötigst, zu entfernen, ziehe ihn in eine Blockpalette zurück.


#### Aufgabe A-01 Mein erstes Programm

🕒 15 min

##### Ziel:

Erstellen eines Programms in Scratch.

##### Beschreibung

- Starte das Programm Scratch.
- Benenne das Sprite um in **Katze**.
- Ziehe den Block  in den Programmbereich. Doppelklicke auf diesen Block. Was stellst du fest? Beobachte die Position des Sprites auf der Bühne! Erkläre!




---



---



---

- Erstelle folgendes Programm :



- e) Starte das Programm indem du es doppelklickst.  
 f) Stoppe das Programm.  
 g) Wie kannst du das Programm noch starten?

- h) Ändere den Rotationsmodus des Sprites und probiere die verschiedenen Möglichkeiten aus:

Erkläre die verschiedenen Rotationsmodi:



Welcher Rotationsmodus ist für dieses Projekt am besten geeignet?

- i) Speichere das Projekt unter dem Namen **Aufgabe A-01** ab.

# SCRATCH

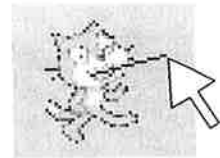
## A.3. Sprites bearbeiten

Um ein Sprite auf eine andere Stelle auf der Bühne zu setzen kannst du es mit gedrückter Maustaste **bewegen**.

Du kannst Sprites **duplizieren**, **löschen**, **vergrößern** und **verkleinern**: in der Werkzeugleiste die gewünschte Option auswählen und auf das Sprite klicken.



Du kannst einen Sprite auch **drehen** indem du mit gedrückter Maustaste die Richtung des Sprites änderst.



Um weitere Sprites in deinem Projekt zu **erstellen** gibt es mehrere Möglichkeiten:

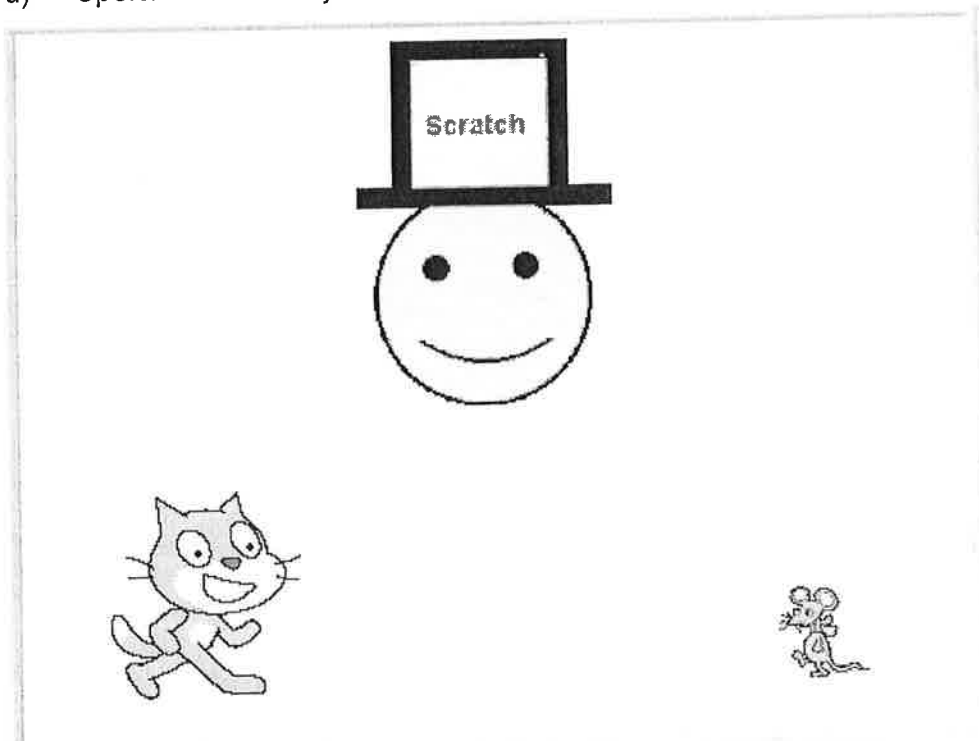


**Aufgabe A-02 Weitere Sprites** 15 min**Ziel:**

Erstellen und bearbeiten neuer Sprites in einem Projekt.

**Beschreibung**

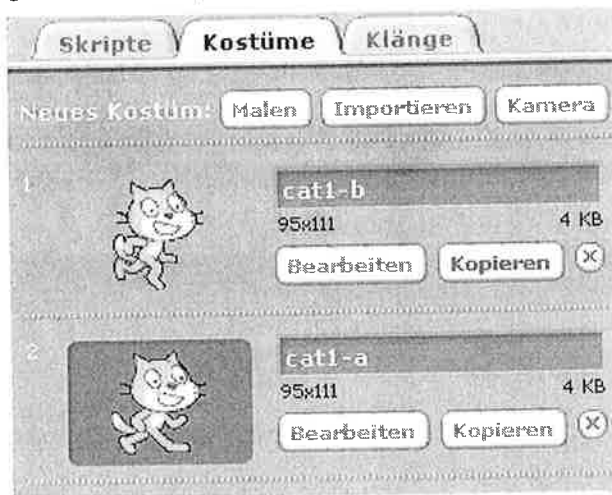
- a) Öffne das Projekt aus der **Aufgabe A-01**.
- b) Erstelle neue Sprites und platziere sie auf der Bühne wie unten abgebildet. Die Maus kannst du aus einer Datei laden, den Smiley sollst du selbst zeichnen.
- c) Gib den neuen Sprites Namen.
- d) Speichere das Projekt unter dem Namen **Aufgabe A-02** ab.





## A.4. Sprites haben Kostüme

Sprites können verschiedene Kostüme haben und deshalb ihr Aussehen verändern. Um dir die Kostüme eines Sprites anzusehen klicke in der Spriteliste auf das gewünschte Sprite und klicke anschließend auf den Reiter **Kostüme**.



Die Katze hat beispielsweise zwei Kostüme. Das aktuelle Spritekleid (**cat1-a**) ist hervorgehoben. Um zu einem anderen Kostüm zu wechseln, klicke einfach auf das Vorschaubild des Kostüms.

Es gibt drei Möglichkeiten um neue Kostüme zu erstellen :

- Klicke **Malen**, um ein neues Kostüm mit dem Paint Editor zu entwerfen.
- Klicke **Importieren**, um eine Bilddatei, die sich auf deiner Festplatte befindet, zu importieren.
- Wenn dein Rechner mit einer Webcam ausgestattet ist, klicke **Kamera** um neue Kostüme zu fotografieren.

Scratch kann viele verschiedene Bildformate verarbeiten: JPG, GIF, BMP und PNG.

Du kannst die Reihenfolge der Kleider ändern, indem du die Vorschaubilder bewegst.

Du kannst auch ein bestehendes Kostüm ändern indem du **Bearbeiten** klickst.



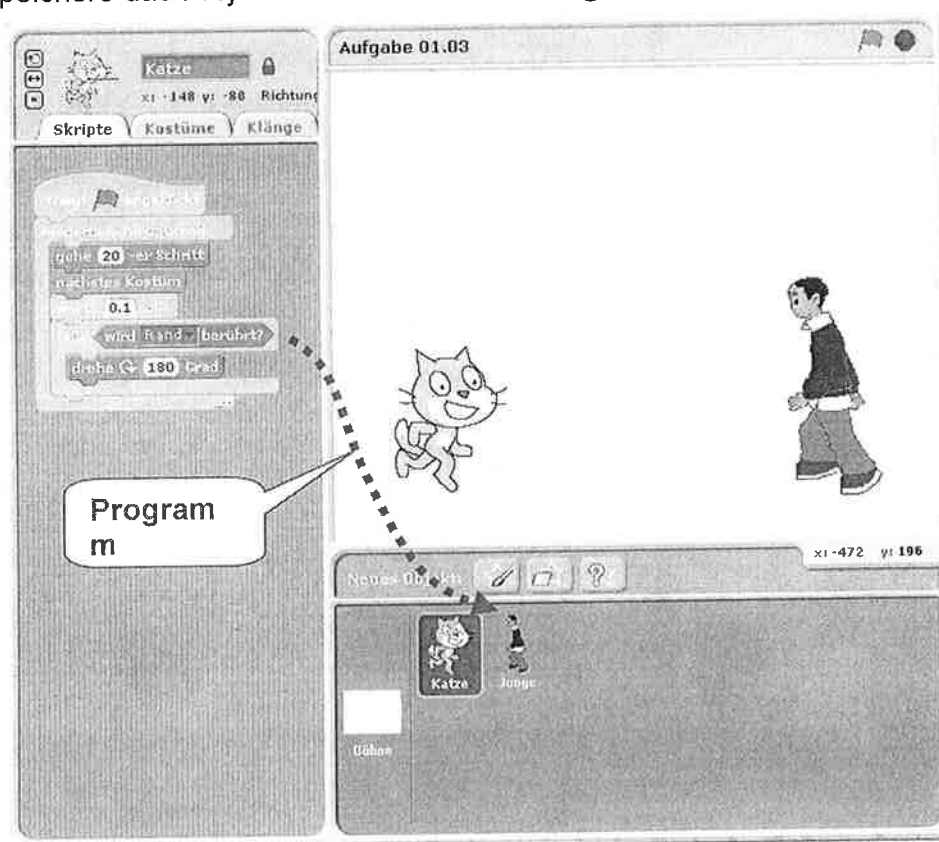
## Aufgabe A-03 Kostüme

🕒 10 min

**Ziel:** Kostüme bearbeiten.

### Beschreibung

- Öffne das Projekt aus der **Aufgabe A-01**.
  - Lade das Sprite **boy4-walking-a** von der Festplatte und platziere es auf der Bühne wie unten abgebildet.
  - Nenne das Sprite **Junge**.
  - Wie viele Kostüme hat der Junge?
- 
- Erstelle weitere Kostüme für den Jungen indem du die Dateien **boy4-walking-b**, **boy4-walking-c**, **boy4-walking-d** und **boy4-walking-e** importierst.
  - Kopiere das Programm des Sprites **Katze** in das Sprite **Junge** (Programm aus dem Programmbereich in das Sprite **Junge** in der Spriteliste ziehen (siehe Abbildung)).
  - Starte nun beide Programme (Klick auf die grüne Flagge).
  - Speichere das Projekt unter dem Namen **Aufgabe A-03** ab.



## A.5. Die Bühne

Genau wie die Sprites ihr Aussehen verändern können mit einem Kostümwechsel, kann die Bühne ihr Aussehen verändern, indem man die **Hintergründe** wechselt.

Außerdem kannst du die Bühne auch programmieren.

Um Hintergründe und Programm der Bühne zu verändern, klicke auf das Bühnensymbol, das sich links von der Spriteliste befindet.

### Aufgabe A-04 Bühne – Hintergründe



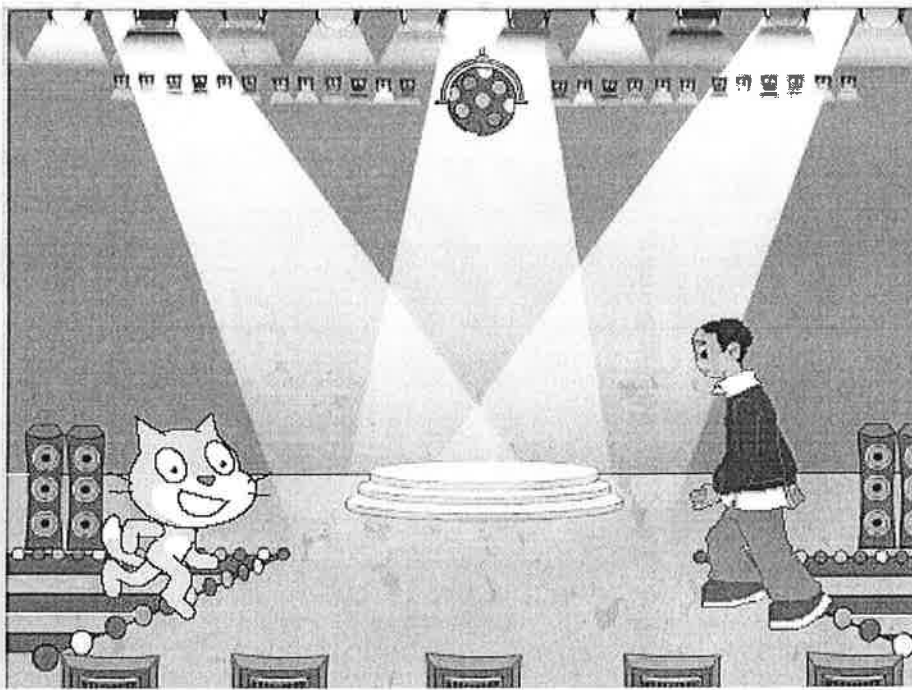
5 min

#### Ziel:

Hintergründe der Bühne bearbeiten.

#### Beschreibung

- Öffne das Projekt aus der **Aufgabe A-03**.
- Ändere den Hintergrund der Bühne wie unten abgebildet.
- Speichere das Projekt unter dem Namen **Aufgabe A-04** ab.





## A.6. Projekt dokumentieren

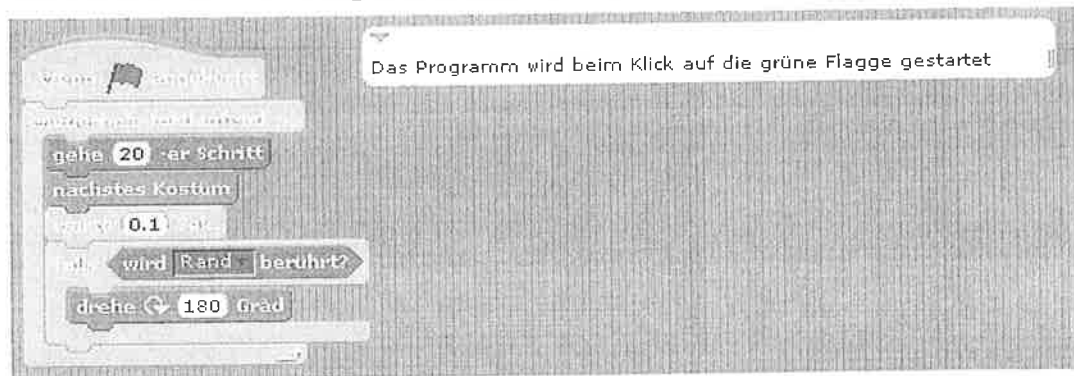
Es ist immer eine gute Idee seine Projekte ausreichend zu dokumentieren. Wenn du im Nachhinein ein Projekt ändern oder erweitern möchtest hilft dir eine gute Dokumentation dich besser zurecht zu finden.

Es gibt zwei Arten von Dokumentation.

- 1) Mit **Projektanmerkungen** kannst du deinem Projekt einen kleinen Text hinzufügen. Wähle dafür den Menüpunkt **Datei** → **Projektanmerkungen...** . Hier kannst du eine globale Beschreibung deines Projektes eingeben: beispielsweise was dein Projekt macht und wie man es benutzt.  
Bemerkung: Die Projektanmerkungen werden auch auf der Scratch Homepage angezeigt, falls du es später dort veröffentlichen willst.

- 2) Mit den **Kommentaren** kannst du einzelne Programmteile dokumentieren. Dies ist besonders bei komplexeren Programmen interessant um nicht die Übersicht zu verlieren.

Klicke mit der rechten Maustaste in den Programmbereich, wähle im Kontextmenü **Anmerkung hinzufügen** und gib einen Kommentar ein.



### Aufgabe A-05 Projektdokumentation

🕒 10 min

**Ziel:**

Projekt dokumentieren

**Beschreibung**

- a) Öffne das Projekt aus der **Aufgabe A-04**.
- b) Füge deinem Projekt eine Projektanmerkung bei.
- c) Versuche herauszufinden wie das Programm des Sprites **Katze** funktioniert und kommentiere jeden Befehlsblock.
- d) Speichere das Projekt unter dem Namen **Aufgabe A-05** ab.



## B. Scratch-Projekte mittels Sequenzen entwickeln

Die einfachste Form der Programmstruktur ist die Sequenz. Mehrere Anweisungen werden nacheinander ausgeführt. In vielen Programmiersprachen werden sie jeweils durch ein Semikolon (Strichpunkt ;) voneinander getrennt. Durch Klammerung mit entsprechenden Schlüsselwörtern oder Zeichen (z. B. *begin* und *end* bei Pascal) werden die Anweisungen zu Blöcken zusammengefasst.

Beispiel einer Sequenz in Scratch:



### Aufgabe B-01

**Ziel:**

Projekte mittels Sequenzen erstellen.

**Beschreibung**

Die Katze soll:

- sich zur Position  $x=0$ ,  $y=-100$  begeben
- nach links schauen
- während 2 Sekunden "Hallo" sagen.
- Speichere das Projekt unter dem Namen **Aufgabe B-01** ab.

## Aufgabe B-02 "Le corbeau et le renard!"

**Ziel:** Projekte mittels Sequenzen erstellen.

### Beschreibung

a) Erstelle ein Projekt mit folgenden Sprites (siehe Bild).  
Den Käse kannst du nach deinem Geschmack selbst zeichnen.

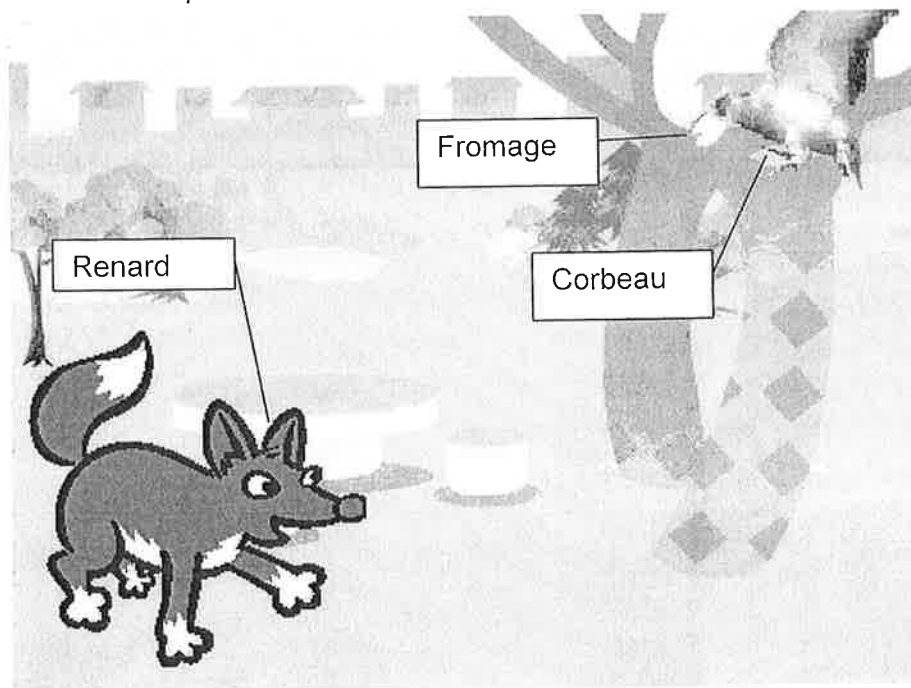
b) Renard soll nacheinander folgenden Text aufsagen:

*Et Bonjour Monsieur du Corbeau,*

*Que vous êtes joli! Que vous me semblez beau!*

*Sans mentir, si votre ramage se rapporte à votre plumage*

*Vous êtes le phénix des hôtes de ces bois*



c) Anschließend soll Corbeau den Text "Merci" sagen, wonach das Käsestück auf den Boden fällt.

### Bemerkungen:

- Überlege dir vorher, wie du die zeitliche Reihenfolge der Aktionen der 3 Sprites aufeinander abstimmen kannst.
  - Der Käse fällt am schönsten mit einer ‚gleite zu ...‘ Bewegung.
  - Achte darauf, dass sich der Käse beim nächsten Start des Programms wieder im Schnabel von ‚Corbeau‘ befindet.
- d) Speichere das Projekt unter dem Namen **Aufgabe B-02** ab.

# SCRATCH

## C. Auf Ereignisse reagieren

Ereignisse werden mit Blöcken symbolisiert, die man als ‚Hüte‘ bezeichnet. Diese Blöcke haben abgerundete Oberkanten, wie zum Beispiel:



Das angedockte Programm wird ausgeführt, wenn die grüne Flagge angeklickt wird.



Das angedockte Programm wird ausgeführt, wenn das angegebene Sprite angeklickt wird.



Das angedockte Programm wird ausgeführt, wenn die angegebene Taste gedrückt wird.

Diese Blöcke werden als oberste Blöcke in Stapeln platziert. Sie warten darauf, dass ein Ereignis geschieht (Z.B. wenn eine bestimmte Taste gedrückt wird). Daraufhin werden dann die Befehlsblöcke unterhalb vom Hutblock ausgeführt.

### Aufgabe C-01

**Ziel:**

Auf Mausclick reagieren

**Beschreibung**

- Erstelle ein Projekt mit einem Sprite, das eine volle Flasche und ein leeres Glas darstellt. Da dieses Sprite nicht als Datei existiert, musst du es entweder selbst zeichnen oder aus dem Internet herunterladen.
- Wenn man auf das Sprite klickt, wird die Flasche nach und nach leer und das Glas nach und nach voll. Wenn die Flasche leer ist, fängt das Ganze von vorne an.



- Speichere das Projekt unter dem Namen **Aufgabe C-01** ab.

## Aufgabe C-02

### Ziel:

Auf Tastaturereignisse reagieren

### Beschreibung

- a) Erstelle ein Projekt, in dem sich die Katze beim Drücken der vier Pfeiltasten in die jeweilige Richtung bewegt.
- b) Speichere das Projekt unter dem Namen **Aufgabe C-02** ab.

# SCRATCH

## D. Die Endlosschleife

In dieser Einheit werden wir uns mit dem Steuerungsblock „wiederhole fortlaufend“ beschäftigen, welcher es uns erlaubt Aktionen endlos zu wiederholen. In der Programmierung nennen wir dies eine ‚Endlosschleife‘.



Wenn wir die Programmblöcke aus früheren Aufgaben in diese Endlosschleife einfügen, stellen wir fest, dass diese Blöcke ausgeführt werden bis wir das rote Stop-Icon drücken.

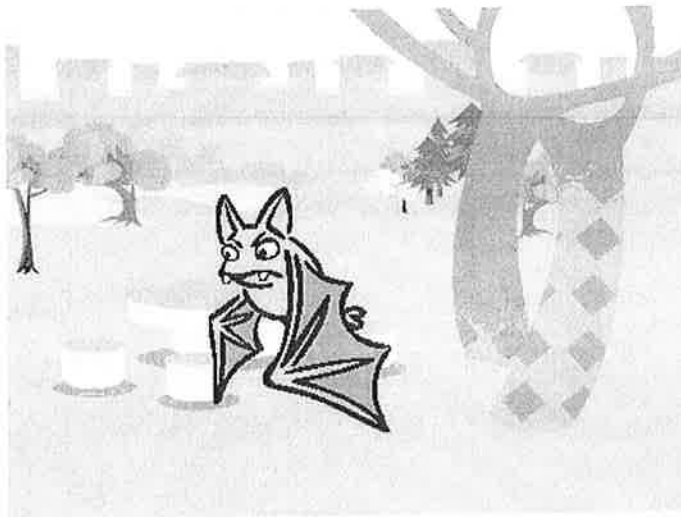
### Aufgabe D-01

#### Ziel:

Benutzen der Endlosschleife.

#### Beschreibung

- a) Erstelle ein Projekt mit folgendem Sprite.



- b) Die Fledermaus hat zwei Kostüme damit man sie fliegen sieht. Wenn die Startflagge angeklickt wird, soll die Fledermaus auf der Bühne hin- und her fliegen.

Speichere das Projekt unter dem Namen **Aufgabe D-01** ab.



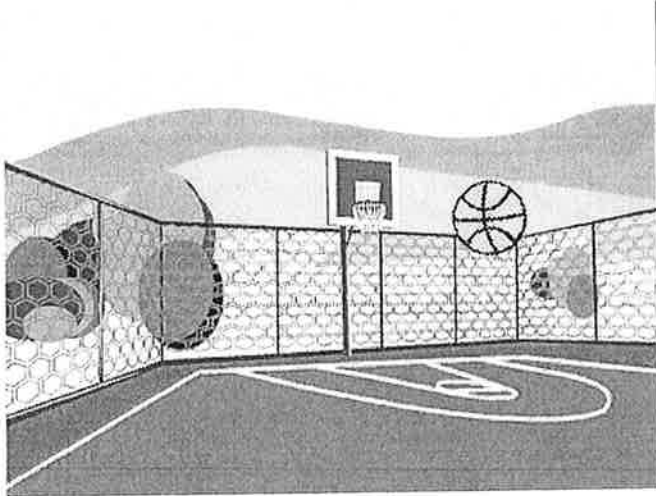
SCRATCH

**Aufgabe D-02      Der verrückte Ball****Ziel:**

Benutzen der Endlosschleife.

**Beschreibung**

- a) Erstelle ein Projekt mit einem Sprite, das einen Ball darstellt.



- b) Der Ball soll sich andauernd zufällig in irgendeine Richtung drehen, sich um 20 Schritte bewegen, vom Rand abprallen und eine 10tel Sekunde warten. Wenn der Ball angeklickt wird, soll er während 1 Sekunde "OK" sagen.
- c) Speichere das Projekt unter dem Namen **Aufgabe D-02** ab.

# SCRATCH

## E. Entscheidungen treffen

Du hast sicher schon bemerkt, dass es in der Programmierung öfters vorkommt, dass man Entscheidungen treffen muss (dies ist im "richtigen Leben" auch nicht anders). So kannst du dein Sprite veranlassen, dies oder jenes zu tun, je nachdem welche Situation sich stellt.

Die Blöcke welche hierzu eingesetzt werden können, sind folgende:



falls eine Bedingung erfüllt ist, führe die Befehle aus



falls eine Bedingung erfüllt ist, führe die Befehle aus, sonst führe andere Befehle aus

Die Bedingung muss jeweils "richtig" oder "wahr" sein um die Befehle ausführen zu können, die unter "falls" stehen.

Es gibt mehrere Arten von Bedingungen, hier sind zwei davon:



so kannst du testen, ob z.B. der Rand oder ein Sprite berührt wurde



so kannst du testen, ob eine Farbe berührt wurde.

Diese Blöcke findest du in der Blockpalette „Fühlen“.

Das Unterkapitel K.2.1 zeigt dir wie du Bedingungen miteinander verknüpfen kannst.

S. 43

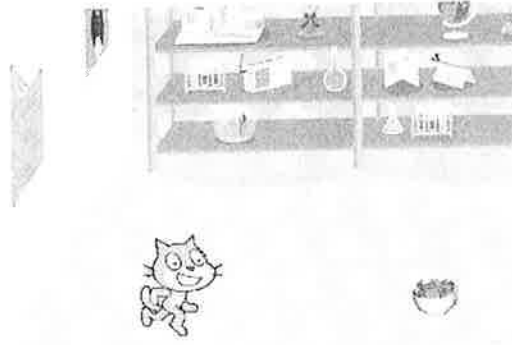
# SCRATCH

## Aufgabe E-01 Such das Futter!

 10 min

### Ziel:

Die Katze sucht das Futter. Hier ein Beispiel:



### Beschreibung:

- Die Katze ist sehr hungrig (und verfressen). Während sie im Zimmer hin- und herläuft, denkt sie "Wo ist das Futter bloß?"
- Findet sie das Futter, sagt die Katze "Hab ich dich!" und das Programm endet.
- Speichere die Lösung als **Aufgabe E-01** ab.

### Erweiterung:

Sobald du das Versenden von Nachrichten mit Scratch beherrschst, kannst du die Katze das Futter fressen lassen.


**Aufgabe E-02 Free the crab!**

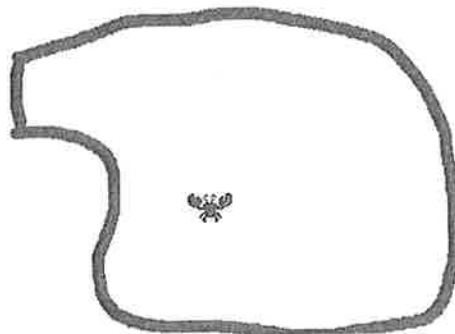
🕒 20 min

**Ziel:**

Die Krabbe soll sich aus dem Käfig befreien.

**Beschreibung:**

- a) Benutze das Sprite **Krabbe**  aus den Vorlagen
- b) Erstelle einen Bühnenhintergrund ("Käfig") welcher aus einer Farbe besteht. Der Käfig hat einen Ausgang, welcher durch eine andere Farbe dargestellt wird, wie zum Beispiel:



- c) Die Krabbe versucht nun, aus dem Käfig auszubrechen. Trifft sie auf die Randfarbe (blau), so denkt sie "Hmmm..", dreht sich etwas und versucht es auf ein Neues. Trifft sie auf die Öffnung (rot), so sagt sie "Endlich frei" und das Programm stoppt.
- d) Speichere dein Programm als **Aufgabe E-02** ab.

**Erweiterung für Fortgeschrittene:**

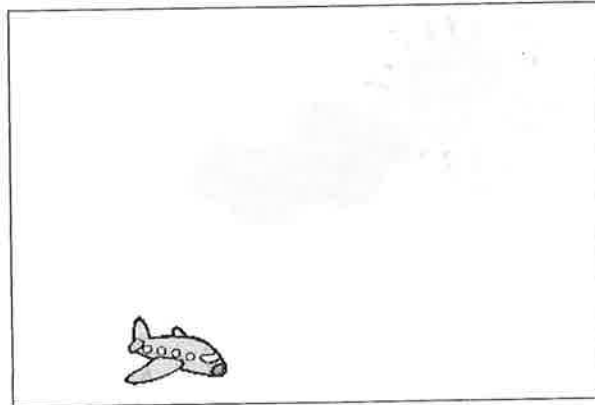
Beim Start des Programms soll die Krabbe in der Mitte des Käfigs erscheinen und in eine zufällig gewählte Richtung schauen.

## Aufgabe E-03 Warm, wärmer, am wärmsten ...

 20 min

### Ziel:

Ein Flugzeug steigt in die Höhe und kann auf eine Wolke oder auf die Sonne treffen und ... schmelzen. Hier ein Beispiel:



### Beschreibung:.

- Erstelle drei Sprites, ein (kleines) **Flugzeug**, eine (graue) **Regenwolke** sowie eine (gelbe) **Sonne**.
- Das **Flugzeug** zeigt in eine beliebige Richtung und steigt auf. Es prallt vom Rand ab, ohne sich auf den Kopf zu drehen.
- Falls es auf die Wolke trifft, landet es, ändert seine Flugrichtung und startet erneut (dabei kann es auch seine Farbe wechseln).
- Falls es auf die Sonne trifft, schmilzt es und verpufft<sup>1</sup>. Das Programm sollte dann stoppen.
- Speichere dein Programm als **Aufgabe E-03** ab

### Erweiterung:

Da du ja während des Programms die Objekte bewegen kannst, könntest du die Wolke vor die Sonne schieben. Programmiere die Wolke so, dass sie sich langsam auf die Sonne zu bewegt.

### Nur für Asse:

Bewege die Sonne auch halbkreisförmig über den Hintergrund.

<sup>1</sup> Diesen Effekt kannst du erreichen, indem du in 10er Schritten die Größe des Flugzeugs änderst und zum Schluss "Pufffff!" sagst, danach das Flugzeug verschwinden lassen.



## **F. Zwischenprojekt**

---

## G. Bedingte Schleifen

Bis heute hast du schon vieles in Scratch gelernt und geübt. Du kannst Befehlsfolgen programmieren, Endlosschleifen und Bedingungen einsetzen.

Beides, Endlosschleifen und Bedingungen kann man auch manchmal kombinieren – du erhältst so eine bedingte Schleife.



Die innere Befehlsfolge wird solange abgearbeitet wie eine Bedingung zutrifft. Trifft die Bedingung nicht zu, wird kein Befehl ausgeführt. Sobald die Bedingung wieder zutrifft, werden die Befehle wieder ausgeführt. Es handelt sich um eine Endlosschleife, d.h. dieser Block wird nicht mehr verlassen.



In diesem Fall wird die Befehlsfolge erst einmal ausgeführt, dann wird geprüft, ob die Bedingung zutrifft. Falls ja, wird die Schleife gestoppt und es geht mit den Befehlen hinter der Schleife weiter. Falls nicht, werden die Befehle aufs Neue ausgeführt.

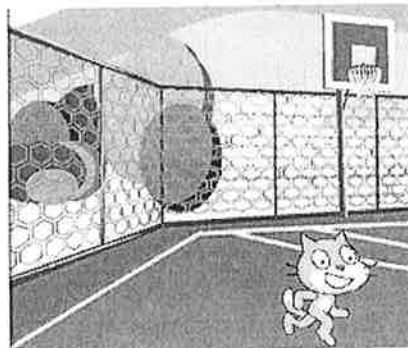
### Aufgabe G-01

### Leerlauf

🕒 5-10 min

Ziel:

Die Katze soll laufen, während die Leertaste gedrückt ist. Hier ein Beispiel:



**Beschreibung:**

- Solange die Leertaste gedrückt ist, läuft die Katze hin und her.
- Speichere als **Aufgabe G-01** ab.

Was passiert wenn du die Taste los lässt?

Was passiert wenn du die Taste erneut drückst?

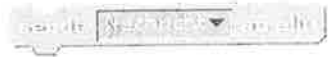


## H. Nachrichten senden und empfangen

Es gibt Situationen in denen ein Sprite ein anderes Sprite (oder die Bühne) beauftragen möchte eine Aktion (Programm) auszuführen.

Scratch ermöglicht dies durch Senden und Empfangen von Nachrichten.

Es gibt zwei Befehle um eine Nachricht zu senden:

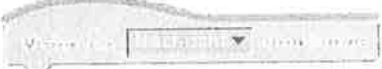


Sendet eine Nachricht an alle Sprites und die Bühne, was diese dann veranlasst Aktionen auszuführen und führt den nächsten Block aus, ohne auf die Fertigstellung der Aktionen zu warten.

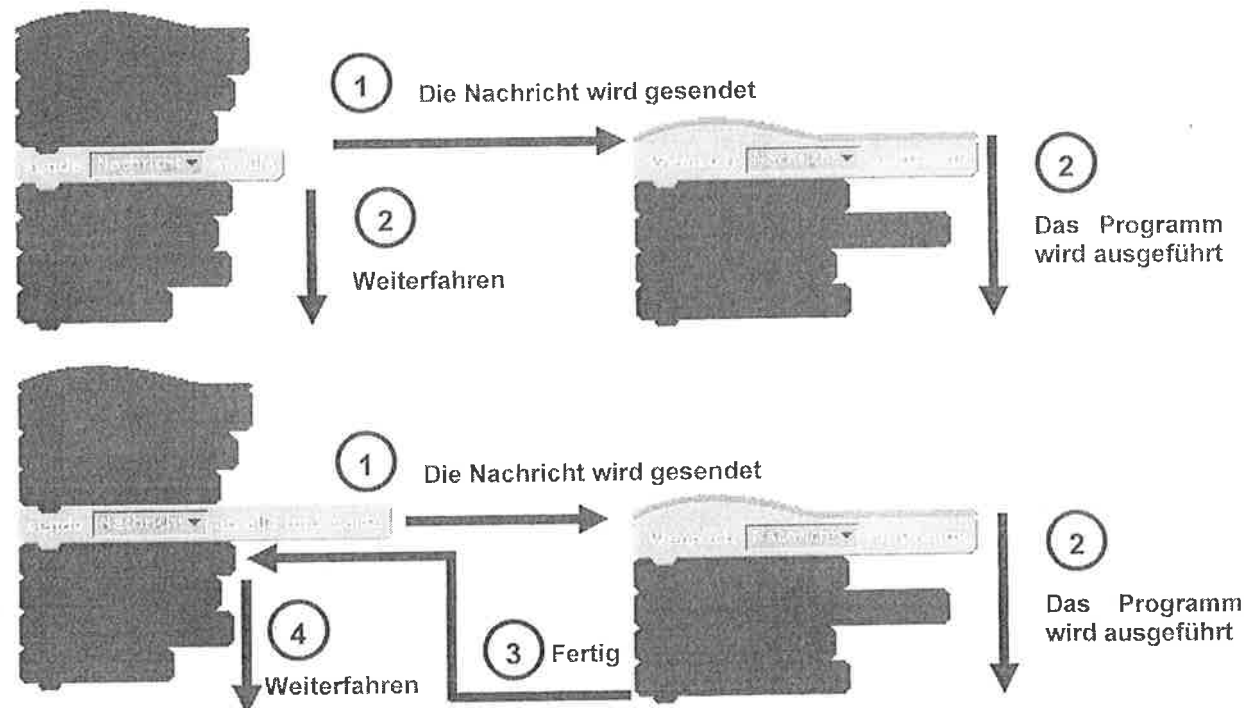


Sendet eine Nachricht an alle Sprites und die Bühne, was diese dann veranlasst Aktionen auszuführen, und wartet bis diese die Aktionen ausgeführt haben, ehe der nächste Block ausgeführt wird.

Wenn ein Sprite oder die Bühne die Nachricht erhält wird das Programm unter dem

Block  ausgeführt.

Die folgenden Schemas sollen den Unterschied zwischen den zwei Sende-Befehlen verdeutlichen:





# SCRATCH

## Aufgabe G-02

### Leerlauf 2

🕒 10 min

#### Ziel:

Die Katze soll laufen, bis die Leertaste gedrückt ist.

#### Beschreibung:

- Die Katze läuft hin und her, bis die Leertaste gedrückt ist.
- Speichere als **Aufgabe G-02** ab.

Was passiert wenn du die Taste drückst?

---

Was passiert wenn du die Taste erneut drückst?

---

## Aufgabe G-03

### Freiheit für die Katze

🕒 20 min

**Ziel:** Die Katze will aus dem Käfig befreit werden.



#### Beschreibung:

- Die Katze läuft im Käfig hin und her (d.h. sie läuft solange sie den Käfig berührt). Am grauen Rand des Käfigs wechselt sie die Richtung. Das Sprite für den Käfig musst du selbst zeichnen.
- Nur für Fortgeschrittene: Alle 10 Sekunden denkt sie "Wie komm ich bloß hier raus?" (hierzu werden die mathematischen Ausdrücke benutzt, siehe Kapitel K).
- Falls die Leertaste gedrückt wird, öffnet sich der Käfig und die Katze dankt für ihre Freiheit.
- Speichere als **Aufgabe G-03** ab.

#### Erweiterung:

Stelle sicher, dass der Käfig sich beim Start des Programms im Vordergrund befindet.

Stelle sicher, dass die Katze sich beim Start des Programms im Käfig befindet.

**Aufgabe H-01 "Husch ins Körbchen !"**

🕒 15 min

**Ziel:**

Senden und Empfangen von Nachrichten.

**Beschreibung**

- a) Erstelle ein Projekt mit folgenden Sprites.



- b) Struppi soll wenigstens zwei Kostüme haben damit man ihn so richtig watscheln sieht.  
 c) Wenn die Oma angeklickt wird, sagt sie "Husch ins Körbchen!" und Struppi soll sich dann sofort in sein Körbchen begeben.

Programmiere die Oma und Struppi indem du folgende Befehle benutzt.



- d) Speichere das Projekt unter dem Namen Aufgabe H-01 ab.  
 e) Erweitere das Projekt folgendermaßen: Wenn Struppi im Körbchen angelangt ist sagt Oma "Braver Struppi !"

# SCRATCH

## Aufgabe H-02 Die Katze rennt durchs Haus

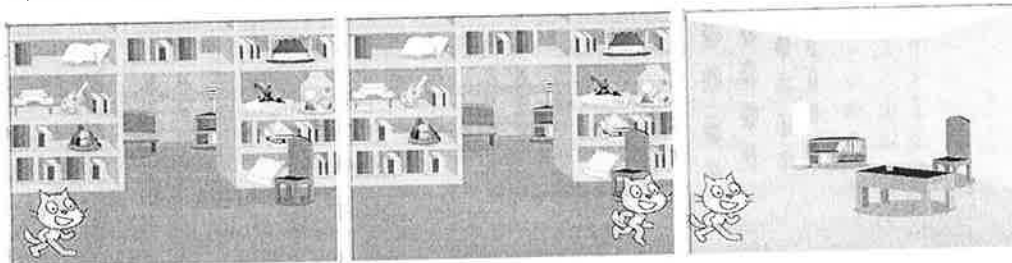
🕒 30 min

### Ziel:

Senden und Empfangen von Nachrichten.

### Beschreibung

- Erstelle ein Projekt mit der Katze und importiere verschiedene Hintergründe für die Bühne, welche verschiedene Räume eines Hauses darstellen.
- Die Katze soll von einem Raum in den anderen rennen. Sie durchquert jeden Raum von links nach rechts. Ist sie in einem Raum am rechten Rand angelangt erscheint sie anschließend im nächsten Raum am linken Rand.



- Speichere das Projekt unter dem Namen **Aufgabe H-02** ab.

## I. Steuerung anhand einer Schleife

In diesem Kapitel lernst du, wie Anweisungen eines Programms kontrolliert mehrmals ausgeführt werden können.

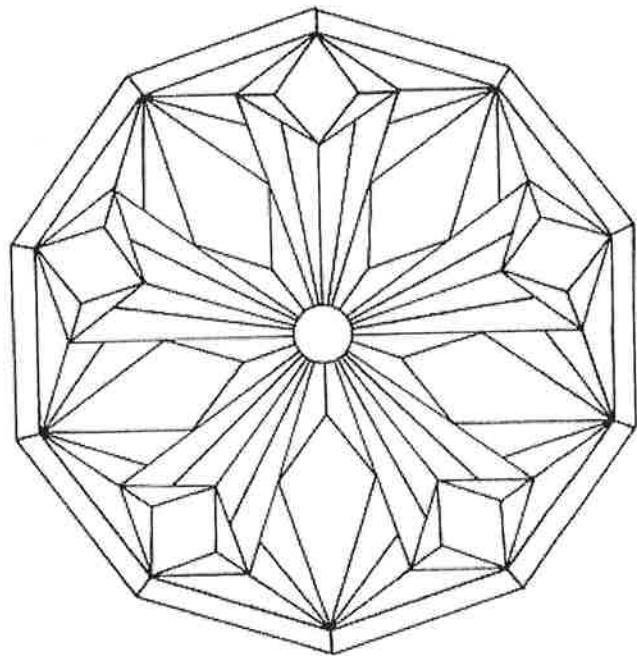
Um diese Einheit zu bewältigen, werden deine Kenntnisse in der Geometrie aufgefrischt (Winkelberechnung: *calcul d'angles* ; Punktsymmetrie / Zentralsymmetrie: *symétrie centrale* ; Drehsymmetrie: *rotation* ; Koordinatensystem: *système de coordonnées, quatre quadrants*).

### Mandala

Das Wort **Mandala** (aus dem Sanskrit) bedeutet so viel wie Kreis und bezeichnet ein kreisförmiges oder quadratisches symbolisches Gebilde mit einem Zentrum, das ursprünglich im religiösen Kontext verwendet wurde (Wikipedia).

Heute denkt man an Ausmalbilder für Kinder und Jugendliche, wenn man von Mandalas redet.

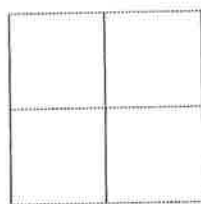
Du wirst in dieser Einheit ein weitaus einfacheres Mandala programmieren.



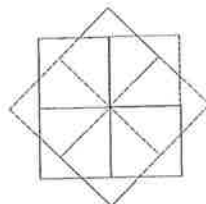
Das Mandala ist um das Zentrum (zentraler Punkt) aus ähnlichen Teilstücken (hier 5, oder genauer  $2 \times 5$ ) zusammengesetzt.

Das Mandala, welches jetzt realisiert wird, besteht zunächst aus einer Vielzahl von Quadraten, welche dann um einen zu berechnenden Winkel um das Zentrum rotierend (drehend) gezeichnet werden.

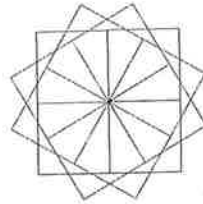
Beispiel:



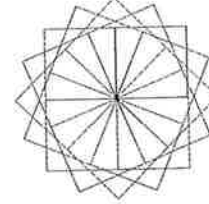
4 Quadrate



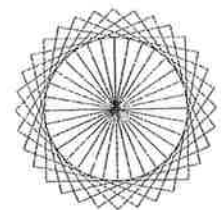
8 Quadrate



12 Quadrate



16 Quadrate



32 Quadrate



## I.1. Benötigte Blockpaletten



## I.2. Zeichnen eines Quadrats

Das Quadrat wird ausgehend vom Zentrum der Bühne (Punkt 0 ; 0) gezeichnet.

Ein Quadrat besteht aus ..... Seiten und ..... Winkel. Die Summe der Winkel im Quadrat beträgt ..... Ein Winkel misst genau ....., es handelt sich um einen ..... Winkel.

Bevor du das Quadrat zeichnest, ist es äußerst wichtig, die Winkel zu identifizieren, welche für das Zeichnen dieses Quadrates relevant sind. Bearbeite zu diesem Zweck zuerst die **Aufgabe I-3**, Blatt „Caractéristiques des polygones réguliers“.

<b>Aufgabe I-01</b>	<b>Ein Quadrat</b>	🕒 20 min
---------------------	--------------------	----------

### - Benötigte Programmblöcke :



Die Argumente der Programmblöcke müssen den Bedürfnissen angepasst werden!  
Jeder Block kann eventuell mehrmals verwendet werden.

Startbedingung: Das Quadrat wird gezeichnet, wenn die Taste „1“ gedrückt wird.  
Die Seitenlänge beträgt 100.

Der Malstift muss auf Position (0 ; 0) gebracht werden.

Alle Malspuren werden vorher entfernt.

## I.3. Zeichnen eines ersten Mandalas

Das Mandala wird aus 8 Quadraten zusammengesetzt (Siehe Beispiel am Kapitelanfang)

Zunächst muss der Winkel zwischen 2 benachbarten Quadraten berechnet werden, damit das Mandala symmetrisch aussieht.

- Miss dazu den Winkel am obigen Mandala und überprüfe folgende Aussage:

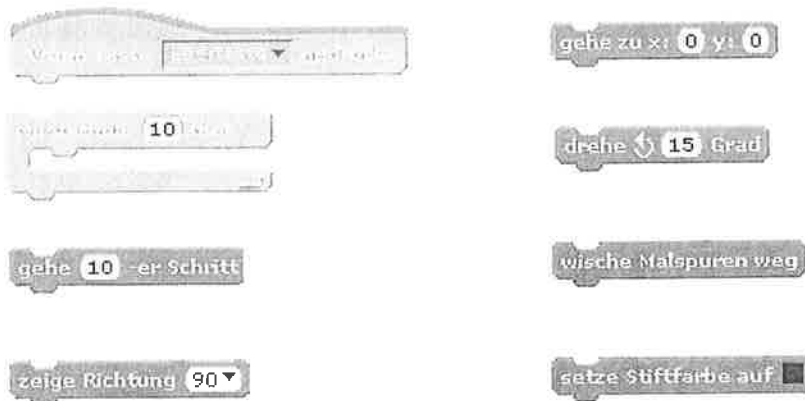
$$\text{Winkel} = 360^\circ / \text{Anzahl der ähnlichen Teilstücke}$$

Berechne jetzt den benötigten Winkel für dein Mandala.

$$\text{Winkel} = \dots\dots^\circ / \dots\dots = \dots\dots^\circ$$

<b>Aufgabe I-02</b>	<b>Mein erstes Mandala</b>	<b>25 min</b>
---------------------	----------------------------	---------------

- **Benötigte Programmblöcke :**



Die Argumente der Programmblöcke müssen den Bedürfnissen angepasst werden!

Startbedingung: Das Mandala wird gezeichnet, wenn die Taste „8“ gedrückt wird.

Der Malstift muss auf Position (0 ; 0) gebracht werden.

Alle Malspuren werden entfernt.

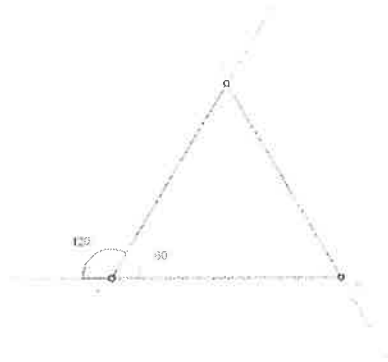
## Aufgabe I-03

## Regelmäßige Polygone

🕒 20 min

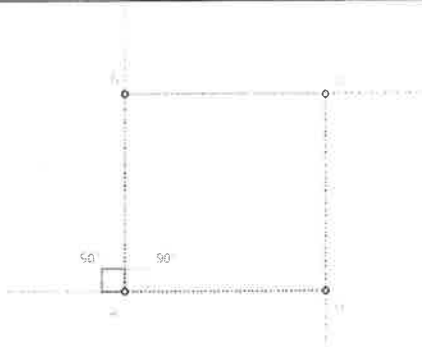
"Vielecke können gleichseitig oder gleichwinklig sein. Hat ein Vieleck gleiche Seiten und gleiche Innenwinkel, dann wird es als reguläres oder regelmäßiges Vieleck bezeichnet."<sup>2</sup>

Zeichne regelmäßige Polygone, nachdem du ihre Eigenschaften analysiert und verstanden hast. Hierzu helfen die hier aufgeführten Texte:



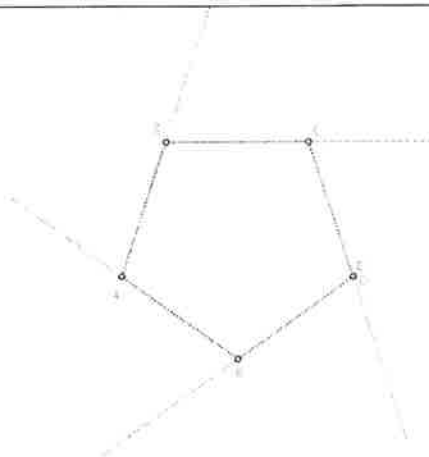
## Triangle équilatéral

1. Identifie les angles qui sont nécessaires pour tracer la figure. Marque-les en rouge !
2. Combien de fois le curseur doit-il changer de direction pour tracer la figure (nombre d'angles)?
3. Que peut-on dire de la somme des angles ?
4. Comment peut-on calculer la valeur d'un des angles ?



## Carré

1. Identifie les angles qui sont nécessaires pour tracer la figure. Marque-les en rouge !
2. Combien de fois le curseur doit-il changer de direction pour tracer la figure (nombre d'angles)?
3. Que peut-on dire de la somme des angles ?
4. Comment peut-on calculer la valeur d'un des angles ?



## Pentagone

Utilise les informations que tu as identifiées pour tracer les deux figures précédentes.

1. Identifie les angles qui sont nécessaires pour tracer la figure. Marque-les en rouge !
2. Combien de fois le curseur doit-il changer de direction pour tracer la figure (nombre d'angles)?
3. Que peut-on dire de la somme des angles ?
4. Comment peut-on calculer la valeur d'un des angles ?

<sup>2</sup> <http://de.wikipedia.org/wiki/Polygon>

## I.4. Weiterführende Aufgaben

### Aufgabe I-04 Zeichnen eines Rahmens

 10 min

Das Mandala wird hervorgehoben, wenn du es mit einem doppelten Rahmen versiehst.

- Oberer rechter Punkt des inneren Rahmens : 150 ; 150
- Stiftdicke 1
- Oberer rechter Punkt des äußeren Rahmens : 160 ; 160
- Stiftdicke 2

Startbedingung: der Rahmen wird gezeichnet, wenn die Taste „r“ gedrückt wird.


### Aufgabe I-05 Mein zweites Mandala

 25 min

Das Mandala besteht nun aus 16 Quadraten. Die Berechnung des Winkels erfolgt mit den vorhandenen Operatoren.

Startbedingung: dieses Mandala wird gezeichnet, wenn die Taste „m“ gedrückt wird.

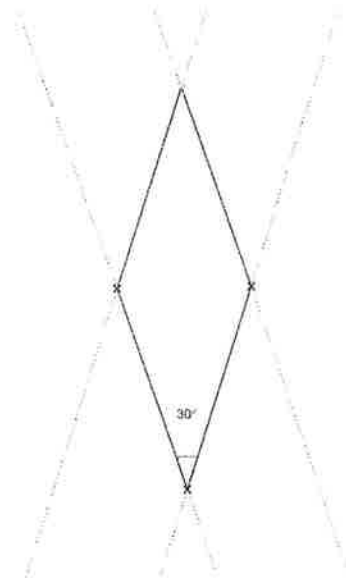
### Aufgabe I-06 Ein schöneres Mandala

 30 min

Das Mandala wird zusehends schöner, wenn es mit Rauten gezeichnet wird.

Startbedingung: dieses Mandala wird gezeichnet, wenn die Taste „t“ gedrückt wird.

Die Raute: Bestimmung der Winkel durch Berechnung und Eintrag auf der beiliegenden Zeichnung.



### Aufgabe I-07 Zusammengesetzte Mandalas

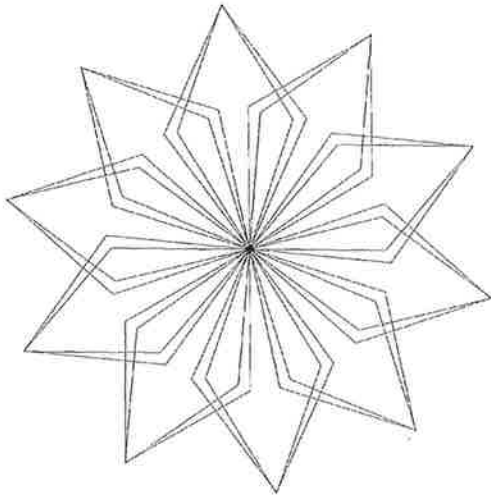
 30 min

Ein Mandala wird aus 2 oder mehreren Mustern zusammengesetzt (gezeichnet).

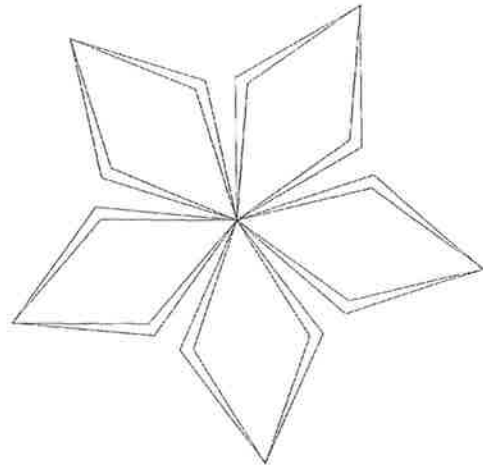


## I.5. Einige einfache, zusammengesetzte Mandalas

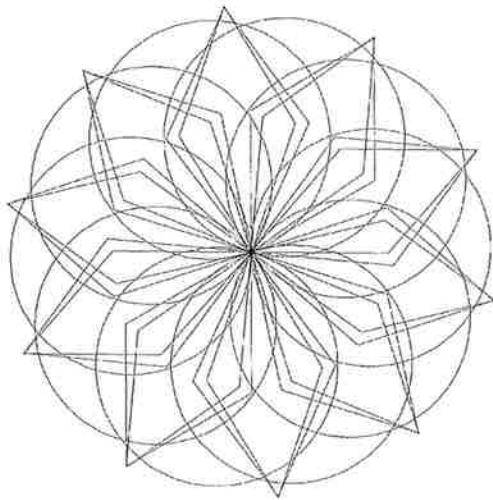
---



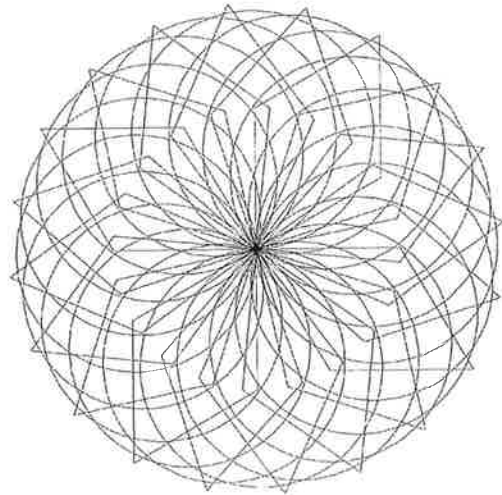
Grosse Raute :  $s = 82$  , Winkel :  $130 / 50$   
 Kleine Raute :  $s = 80$  , Winkel :  $140 / 40$   
 10 Drehungen



Grosse Raute :  $s = 82$  , Winkel :  $130 / 50$   
 Kleine Raute :  $s = 80$  , Winkel :  $140 / 40$   
 5 Drehungen



Kreis :  $s = 8$  ; Winkel =  $6$   
 10 Drehungen



Kreis :  $s = 8$  ; Winkel =  $6$   
 20 Drehungen



## J. Zwischenprojekt

---

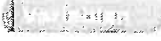
## K. Variablen und mathematische Ausdrücke

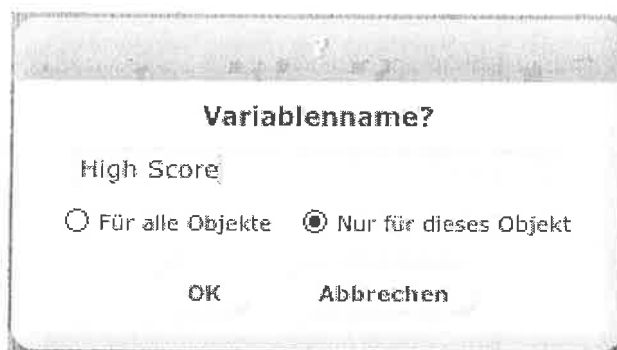
### K.1. Variablen

Sehr oft muss man sich irgendwelche Dinge merken: Eine Hausnummer, den Pin-Code eines Telefons, ein Passwort, den Namen eines Telefonpartners, wie viel Geld man noch in der Tasche hat, was man am Nachmittag noch alles erledigen muss, usw. Anstelle von Notizzettelchen oder Knoten im Taschentuch benutzt man in der Programmierung zu diesem Zweck so genannte ‚Variablen‘.

Variablen ermöglichen es uns Werte zu speichern, zu lesen und zu verändern. In Scratch können Variablen verschiedener Art benutzt werden: ganze Zahlen, Dezimalzahlen oder Texte.

#### K.1.1 Variablen erzeugen und anzeigen

In der Blockpalette  können neue Variablen erzeugt, verändert oder gelöscht werden. Beim Erzeugen einer neuen Variablen müssen wir der Variablen zuerst einen guten und sinnvollen Namen geben:




Wir entscheiden hier auch, welche Objekte in unserem Programm die Variable benutzen und verändern dürfen.

#### Prinzip:

Im Allgemeinen soll eine Variable immer so privat wie möglich bleiben ( → ‚nur für dieses Objekt‘). Kein anderes Objekt soll die Möglichkeit haben unsere Variable absichtlich oder unbeabsichtigt zu verändern. (Wir lassen unsere Adress- und Telefonlisten auch nicht offen in der Mitte des Klassensaales liegen.)

Nur in Ausnahmefällen müssen wir Daten ‚für alle Objekte‘ zur Verfügung stellen.

Sobald eine Variable erstellt wurde, sehen wir die Variable selbst als Block:

 (das Häkchen zeigt an, dass die Variable auf der Bühne sichtbar ist)

Auf der Bühne erscheinen die Variablen als



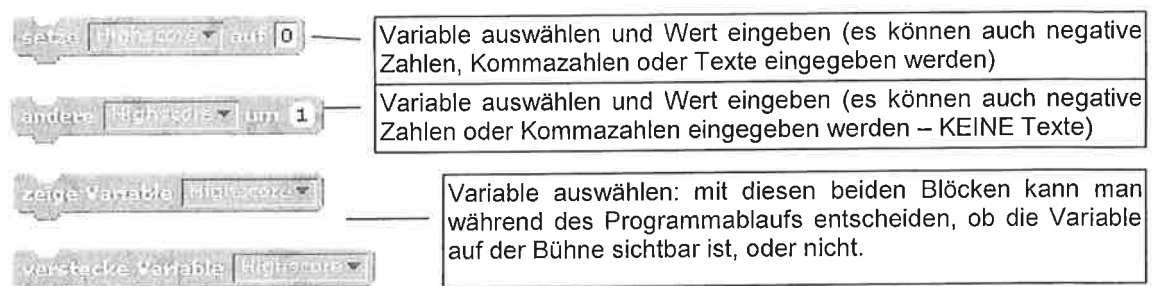
Das Aussehen der Variablen kann auf der Bühne durch einen Doppelklick oder einen Klick mit der rechten Maustaste auf die Variable verändert werden.

### Spezialfall Regler:

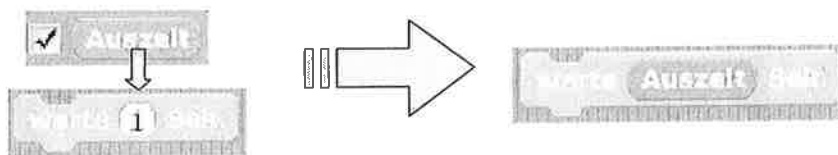
- Regler funktionieren natürlich nur für Zahlen.
- Bei einem Regler kann man mit einem Rechtsklick auch den ‚Reglerbereich festlegen‘, d.h. man kann das Maximum und das Minimum für den Regler einstellen.

### K.1.2 Mit Variablen arbeiten

Nachdem die erste Variable erzeugt wurde erscheinen einige Blöcke, die es uns erlauben die Variablen mit unseren Skripten zu ändern:



In Programmblöcken können Variablen überall eingesetzt werden, wo man bisher Zahlen oder Texte direkt eingetippt hat. Man muss den Variablenblock nur mit der Maus an die richtige Stelle in einen Programmblock hineinziehen.



**Achtung: Texte und Zahlen können nicht beliebig vermischt werden!**

- Zahlenvariablen können überall verwendet werden.
- Textvariablen sollte man in den dafür vorgesehenen Stellen einsetzen (sage [ ]... , denke [ ] ... , frage [ ] ...)

**Aufgabe K-01      Free the crab - reloaded!**  **5 min**

**Ziel:**

Die Krabbe soll sich aus dem Käfig befreien und mitzählen, wie oft sie an die Wände stößt.

**Beschreibung:**

- a) Öffne deine **Aufgabe E-02** („Free the crab“).
- b) Füge eine Variable ‚Zusammenstösse‘ hinzu, die jedes Mal erhöht wird, wenn die Krabbe an eine blaue Wand stößt.
- c) Starte das Programm erneut, nachdem die Krabbe den Ausgang gefunden hat und neu positioniert wurde. Was stellst du fest? Behebe dieses Problem!
- d) Speichere dein Programm als **Aufgabe K-01** ab.

**Aufgabe K-02      Free the crab - revisited!**  **5 min**

**Ziel:**

Die Krabbe soll sich aus dem Käfig befreien und mitzählen, wie oft sie an die Wände stößt. Die Geschwindigkeit der Krabbe ist einstellbar.

**Beschreibung:**

- a) Öffne deine **Aufgabe K-01** („Free the crab - reloaded!“).
- b) Füge eine Variable ‚Auszeit‘ hinzu, und stelle sie als Regler dar. Die Werte des Reglers können zwischen 0 und 0.5 verändert werden. Baue die Variable *Auszeit* so in dein Programm ein, dass sie die Bewegungsgeschwindigkeit der Krabbe bestimmt.
- c) Speichere dein Programm als **Aufgabe K-02** ab.

**Aufgabe K-03**

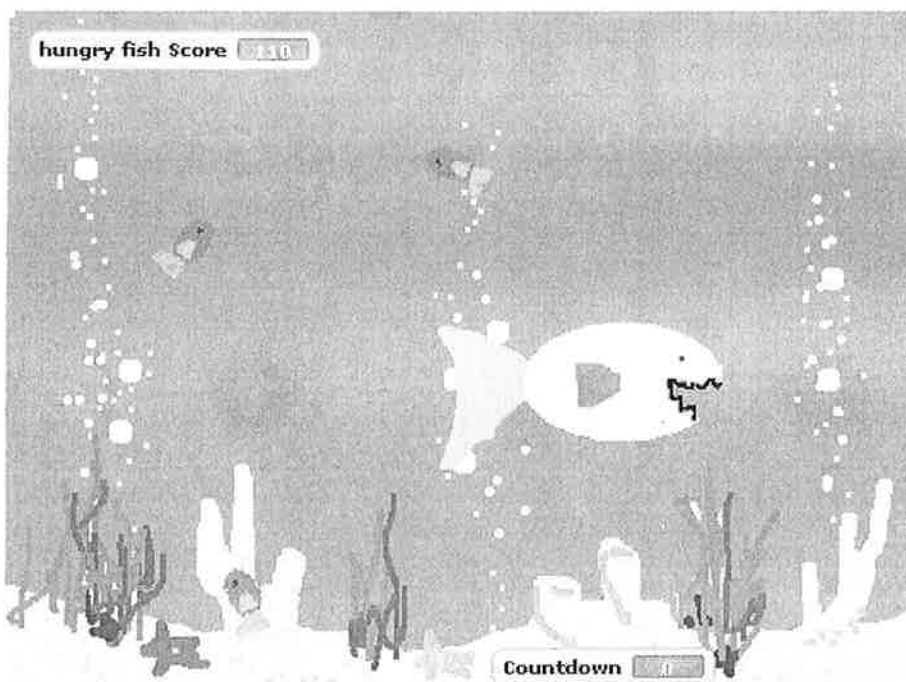
**Fishchomp - II**

 **20 min**

**Ziel:** Ein existierendes Spiel mit Punktestand und Zeitlimit ausstatten.

**Beschreibung:**

- Öffne das im Scratch Paket enthaltene Spiel **FishChomp** (Beispiele → Games → Fishchomp). Teste das Spiel kurz (!)  
Prinzip: der große Fisch folgt der aktuellen Mausposition um die roten Fische zu fressen.  
 Siehe dir die Skripte der einzelnen Objekte an und versuche die Funktionsweise des Programms nachzuvollziehen.
- Erstelle eine neue Variable ‚Score‘, die bei jedem Verschlucken eines roten Fisches um 10 Punkte erhöht wird. Bei Programmstart muss Score natürlich auf 0 zurückgesetzt werden.
- Erstelle eine Variable ‚Countdown‘, die du beim Programmstart auf 30 setzt. Benutze eine Schleife, um den Countdown jede Sekunde um 1 zu erniedrigen. Sind 30 Sekunden abgelaufen, wird das Spiel angehalten.
- Zeige die Spielanleitung nur während den ersten 5 Sekunden des Spielablaufs.
- Speichere dein Programm als **Aufgabe K-03** ab.



### K.1.3 Vordefinierte Variablen






Für jedes Objekt gibt es eine Reihe von vordefinierten Variablen, die man genau so verwenden kann, wie die selbst definierten Variablen:

- Blockpalette **Bewegung** : x,y-Position, Richtung (Winkel)
- Blockpalette **Aussehen** : Kostüm Nr. , Größe
- Blockpalette **Klang** : Lautstärke, Tempo
- Blockpalette **Fühlen** : Antwort, Stoppuhr, Lärmpegel

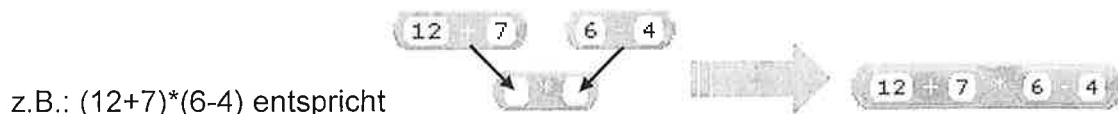
### K.2. Ausdrücke

Wie in jeder Programmiersprache kann man in Scratch auch Werte (Zahlen und Texte) vergleichen und mit ihnen rechnen. Man benötigt nicht viele verschiedene Operationsblöcke um eine sehr mächtige Programmiersprache zu erhalten.

Die meisten dieser Blöcke befinden sich in der Blockpalette „Operatoren“ und behandeln Zahlen:

	Addition, Subtraktion Multiplikation, Division	für ganze Zahlen und Dezimalzahlen
	Zufallszahl liefern	nur für ganze Zahlen
	Rest einer Division	nur für ganze Zahlen
	auf/abrunden einer Zahl	für Dezimalzahlen Resultat: ganze Zahl
	verschiedene mathematische Funktionen	Wichtigste Funktionen: Wurzel : (fr. <i>racine carrée</i> ) Betrag : (fr. <i>valeur absolue</i> ) Weitere: sin, cos, tan, asin, acos, atan, ln, log, e <sup>^</sup> , 10 <sup>^</sup>


Man kann Rechenblöcke ineinander einsetzen um komplexere Berechnungen zu erhalten. Dies ist vergleichbar mit den Klammern in der Mathematik:



Dabei muss man jedoch darauf achten, dass die Blöcke in der richtigen Reihenfolge eingesetzt werden.

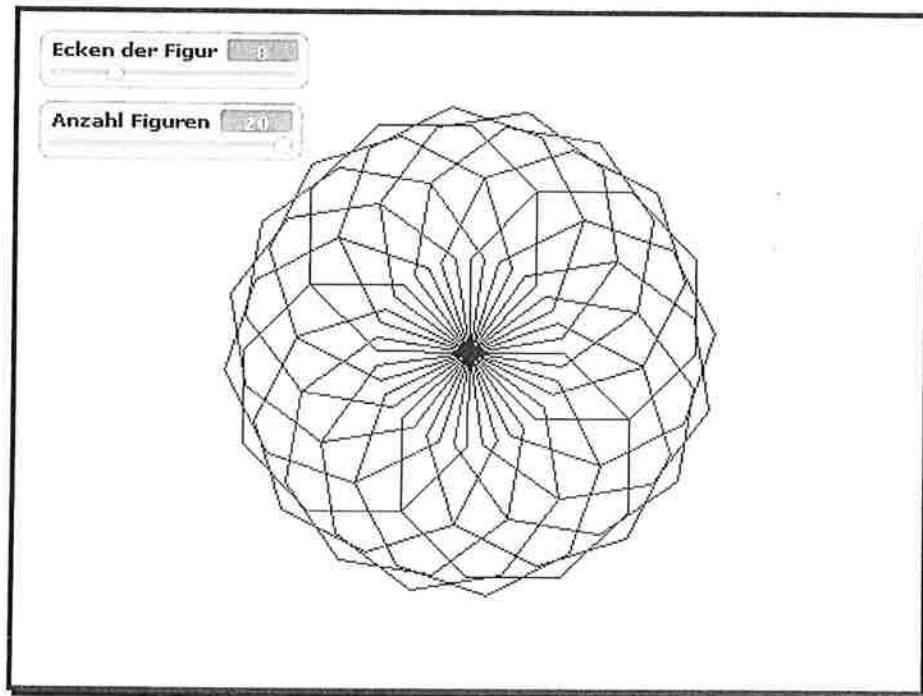
**Prinzip:** Berechnung von oben nach unten: d.h. der oberste Block wird zuerst berechnet, der unterste zuletzt.

**Aufgabe K-04 Mandala Generator**

 30 min

Ziel:

Mandalas automatisch berechnen lassen



**Beschreibung:**

- Definiere 2 Variablen ‚Ecken der Figur‘ und ‚Anzahl Figuren‘, welche du als Regler darstellst. Der Wertebereich für beide Regler ist [3...20].
- Wenn die ‚Leertaste‘ (*Space*) gedrückt wird, soll ein Mandala gezeichnet werden, welches aus so vielen Figuren besteht, wie in der ‚Anzahl Figuren‘ eingestellt wurde. Jede Figur hat so viele Ecken, wie in ‚Ecken der Figur‘ eingestellt wurde.  
Lasse auch die Seitenlänge der Figuren automatisch so berechnen, dass die Figuren noch auf den Bildschirm passen.
- Verfeinere die Darstellung der Mandalas nach deinem Geschmack (Farbe, Stiftdicke, Farbstärke, übermalen, ...)
- Speichere dein Programm als **Aufgabe K-04** ab.





### K.2.1 Wahr oder falsch

Wie ihr im Kapitel E (Entscheidungen treffen) schon gesehen habt, dienen Blöcke mit spitzen Enden dazu eine **Bedingung** zu formulieren.

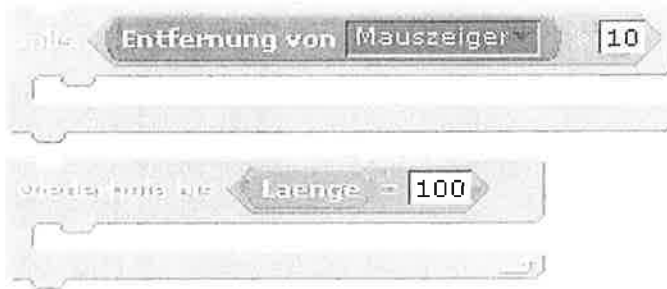
Um genauer zu sein: Ein Block mit spitzen Enden liefert entweder den Wert „**wahr**“ (Bedingung erfüllt) oder „**falsch**“ (Bedingung nicht erfüllt). Solche Blöcke kann man dann z.B. in „falls <...> ...sonst...“ ; warte bis <...> ; wiederhole bis <...> - Blöcken benutzen.

Die Gruppe „**Operatoren**“ ermöglicht es uns nun, Werte (Zahlen oder Texte) miteinander zu vergleichen um diese zur Steuerung unserer Skripte einsetzen zu können.

Hierzu gehören die Vergleiche:



#### Beispiele:



### Aufgabe K-05

### Fishchomp - III



10 min

**Ziel:** Höchste Punktzahl (Highscore) merken und anzeigen

#### Beschreibung:

- Öffne dein Spiel *FishChomp* – II (Aufgabe K.3).
- Ändere die Berechnung des Countdowns so, dass du jetzt eine ‚wiederhole bis‘ Schleife benutzen kannst (Dies ergibt eine flexiblere Lösung, da du nun die Spieldauer des Spieles ändern kannst, indem du nur an einer einzigen Stelle den Anfangswert für ‚Countdown‘ änderst).
- Erstelle dann eine Variable „*Highscore*“, die bei Spielende überprüft, ob *Score* den aktuellen *Highscore* übertrifft. Ist das der Fall, wird der aktuelle *Score* als *Highscore* übernommen und ein Klang (deiner Wahl) wird abgespielt.
- Speichere dein Programm als **Aufgabe K-05** ab.

**Aufgabe K-06**

**Ratespiel**

🕒 >60 min

**Ziel:**

Eine vom Computer ausgedachte geheime Zahl erraten

**Vorgeschichte:**

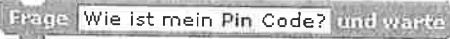

Du musst unbedingt telefonieren, aber der Akku deines Telefons ist leer. Cassy hat ein Telefon dabei, aber sie möchte eigentlich nicht, dass du mit ihrem Telefon anrufst, deshalb stellt sie dir folgende Aufgabe: *„Wenn du meinen vierstelligen geheimen Pin Code in weniger als 15 Versuchen errätst, kannst du telefonieren, solange du willst. Ich sage dir bei jedem Versuch nur, ob mein Pin Code größer oder kleiner ist, als die Zahl, die du rätst.“*



...

## Beschreibung:

Erstelle ein Scratch Programm das Cassy und ihr Ratespiel simuliert:

- Beim Start deines Programms wird in einer für den Benutzer unsichtbaren Variable namens ‚**Pin**‘ eine Zufallszahl zwischen 1000 und 9999 gespeichert. Diese Zahl muss der Spieler dann erraten.
- Die Vorgeschichte kannst du kurz darstellen. (Z.B. Cassy telefoniert anfangs, und sagt nach dem Telefongespräch: „Aha, du willst also mit meinem Telefon anrufen“, „Dazu musst du zuerst meinen Pin Code in weniger als 15 Versuchen erraten!“)
- Benutze den Block  um eine neue Zahl einzugeben. Diesen Block findest du in der Blockpalette ‚**Fühlen**‘. Der Wert, der als Antwort vom Spieler eingetippt wurde, steht dann automatisch in der vordefinierten Variablen .
- Eine weitere (sichtbare) Variable namens ‚**Versuche**‘ zeigt an, wie viele Rateversuche du schon unternommen hast.
- Falls du die Pin-Zahl in weniger als 15 Versuchen geraten hast, ärgert sie sich und gibt dir das Telefon ☹
- Falls du die Zahl nach 14 Versuchen immer noch nicht erraten hast, freut sie sich und tanzt einige Sekunden lang...
- Bilder für Cassy findest du in der Bilderbibliothek von Scratch (→ Kostüme → Importieren → People). Für die Tanzeinlage kannst du Klänge aus der Bibliothek importieren oder eigene Klänge benutzen. Beim Tanzen kann es auch ganz lustig sein einige Effekte aus der Blockpalette ‚**Aussehen**‘ einzusetzen :



Viel Spaß beim Programmieren und beim Raten!

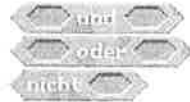


## K.2.2 Für Fortgeschrittene: Zusammengesetzte Bedingungen

Oft benötigt man mehrere Bedingungen, um etwas zu überprüfen. Diese Bedingungen können dann zusammengesetzt (miteinander verknüpft) werden

In der Mathematik z.B. muss man manchmal überprüfen, ob ein Wert in einem bestimmten Bereich liegt:  $0 < X < 1000$

Eine solche Bedingung kann man in Scratch mit folgenden Blöcken erstellen:



<und> bedeutet, dass beide Bedingungen gleichzeitig erfüllt sein müssen

<oder> bedeutet, dass mindestens eine der beiden Bedingungen erfüllt sein muss

<nicht> bedeutet, dass die Bedingung nicht erfüllt sein darf

Um zu überprüfen, ob  $0 < X < 1000$  muss man in Scratch schreiben:



Um zu überprüfen, ob  $X \leq 0$  oder  $X \geq 1000$  ist, kann man in Scratch schreiben:



Dasselbe kann man aber auch so schreiben:

