

Computational Thinking

Jedes Kind verdient es, gut auf die Zukunft vorbereitet zu werden!
Jede Schule und jede Lehrkraft sollte dabei unterstützt werden.

Harald Zigala
h.zigala@tsn.at

Was ist Computational Thinking?

- Seymour Papert: Children, computers and powerful ideas (1980)
- Jeanette Wing: Artikel Computational Thinking (2006)



Was ist Computational Thinking?



☰ Rechnerisches Denken

🌐 20 Sprachen ▾

Artikel [Sprechen](#)

[Lesen](#) [Bearbeiten](#) [Siehe Verlauf](#)

Aus Wikipedia, der freien Enzyklopädie

Computational Thinking (CT) ist die mentale Fähigkeit, Konzepte, Methoden, Problemlösungstechniken und logisches Denken anzuwenden, die aus Informatik und Informatik stammen, um Probleme in allen Bereichen, einschließlich unseres täglichen Lebens, zu lösen. ^[1] In der Bildung ist CT eine Reihe von [Problemlösungsmethoden](#), bei denen Probleme und ihre Lösungen auf eine Weise ausgedrückt werden, die auch ein Computer ausführen könnte. ^[2] Es beinhaltet die Automatisierung von Prozessen, aber auch die Verwendung von Computern, um Prozesse (natürliche und künstliche) zu erforschen, zu analysieren und zu verstehen. ^[3] ^[4] ^[5]

Was ist Computational Thinking?

<https://chat.openai.com/chat>



liefere 10 mögliche übersetzungen des begriffs computational thinking



1. Rechnergestütztes Denken
2. Berechnendes Denken
3. Algorithmisches Denken
4. Datenorientiertes Denken
5. Problemlösungsorientiertes Denken
6. Systematisches Denken
7. Strukturiertes Denken
8. Logisches Denken
9. Abstraktes Denken
10. Analytisches Denken

Was ist Computational Thinking?

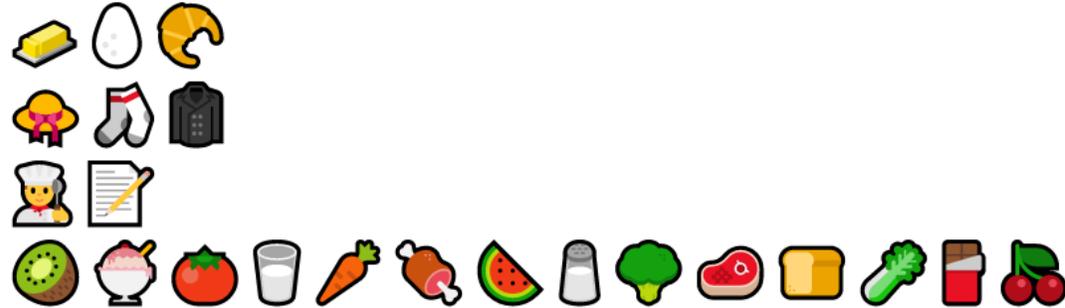
Swisscom.ch

Mit CT werden Probleme und deren Lösungen so formuliert, dass sie sowohl Menschen als auch Maschinen verstehen und ausführen können. Das Problem wird methodisch so zerlegt, dass Muster erkannt und Regeln formuliert werden können, die schliesslich zur Problemlösung führen. So vereinfacht, kann das Problem auch von einem Computer gelöst werden.

CT:= kreative Problemlösung, sodass sie von Menschen oder Computern gemeistert werden kann

Meist wiederkehrende Aufgaben wie ...

- Aufstehen/Frühstück
- Anziehen
- Kochen/Rezept
- Einkauf



... aber auch wie ...

- Hausübungen
- Wanderung
- Programmierungen
- ...

bzw. computational thinking ist die große Schwester des Hausverstands



Robot Karol

- Mini-Language
- 2001 (D); aktuelle Version 3.0.4 (2019)
- kostenlos; keine Hardware
- erste Versionen nur Windows
- Java-basierter Karol für alle Plattformen (Win, iOS, Linux)

- 2D & 3D-Ansicht
- Verschiedene Abmessungen der Welten

- Sprache Deutsch
- Code direkt neben Ausführungsfenster
- z.T. Parameterübergabe
- Syntaxprüfung
- Einzelschritt & Schnelldurchlauf & Stopp

- Automatische Formatierung
- Struktogramm möglich
- Hilfe vorhanden

The screenshot displays the Robot Karol software interface. At the top, the title bar reads "Robot Karol" with standard window controls. Below the title bar is a menu bar with options: "Datei", "Bearbeiten", "Suchen", "Struktogramm", "Welt", "Ablauf", "Einstellungen", and "Hilfe".

The main interface is divided into several sections:

- Code Editor (Top Left):** Contains a list of 12 lines of code:

```
1 {Hallo, ich bin Karol!!}
2 wiederhole 20 mal
3   Hinlegen(rot)
4   Schritt
5   Hinlegen(gelb)
6   Schritt
7   Hinlegen(grün)
8   Schritt
9   Hinlegen(blau)
10  Schritt
11  LinksDrehen
12 *wiederhole
```
- 3D World View (Top Right):** Shows a 3D perspective of a grid world. A small robot figure (Karol) is positioned on the grid. A dashed blue line indicates a path or boundary. A north arrow labeled "N" is visible in the upper left of the grid.
- Command List (Bottom Left):** A tree view of predefined instructions and conditions:
 - Kontrollstrukturen
 - Vordefinierte Anweisungen
 - Schritt
 - Schritt(ANZAHL)
 - LinksDrehen
 - RechtsDrehen
 - Hinlegen
 - Hinlegen(ANZAHL)
 - Hinlegen(FARBE)
 - Aufheben
 - Aufheben(ANZAHL)
 - MarkeSetzen
 - MarkeSetzen(FARBE)
 - MarkeLöschen
 - Warten
 - Warten(mSEK)
 - Ton
 - Langsam
 - Schnell
 - Beenden
 - FARBE=rot,gelb,blau,grün
 - Vordefinierte Bedingungen

- Program Information (Bottom Right):** Displays the current program file path: "Programm : C:\schuljahr22-23\computational thinking\Hallo, ich bin Karol-001.kdp" and the current world name: "Welt :".
- Status Bar (Bottom):** Includes a "Übersicht" button, a "PositionX: 1" field, a "PositionY: 1" field, a "Blickrichtung: S" field, and an "Information" button.

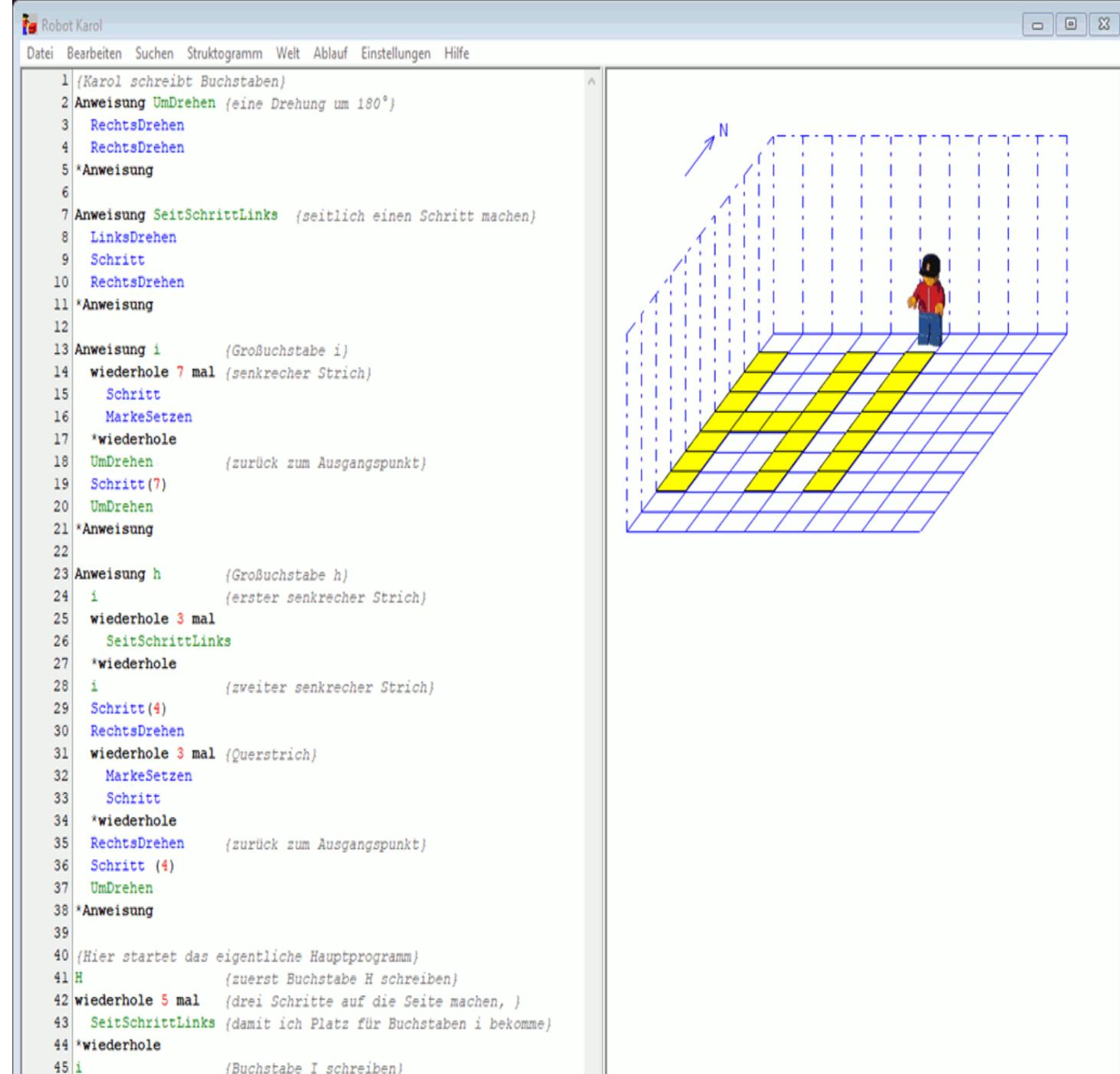
Computational Thinking

1. Dekomposition

- Ein komplexes Problem wird in verschiedene verstehbare Teilprobleme unterteilt.

2. Mustererkennung

- Innerhalb der Teilprobleme werden Muster gesucht. Oft sind die erkannten Abläufe bereits ganz oder in Teilen aus anderen Problemlösungsprozessen bekannt.



The screenshot displays the 'Robot Karol' software interface. On the left is a code editor with the following text:

```
1 {Karol schreibt Buchstaben}
2 Anweisung UmDrehen {eine Drehung um 180°}
3 RechtsDrehen
4 RechtsDrehen
5 *Anweisung
6
7 Anweisung SeitSchrittLinks {seitlich einen Schritt machen}
8 LinksDrehen
9 Schritt
10 RechtsDrehen
11 *Anweisung
12
13 Anweisung i {Großbuchstabe i}
14 wiederhole 7 mal {senkrecher Strich}
15 Schritt
16 MarkeSetzen
17 *wiederhole
18 UmDrehen {zurück zum Ausgangspunkt}
19 Schritt(7)
20 UmDrehen
21 *Anweisung
22
23 Anweisung h {Großbuchstabe h}
24 i {erster senkrecher Strich}
25 wiederhole 3 mal
26 SeitSchrittLinks
27 *wiederhole
28 i {zweiter senkrecher Strich}
29 Schritt(4)
30 RechtsDrehen
31 wiederhole 3 mal {Querstrich}
32 MarkeSetzen
33 Schritt
34 *wiederhole
35 RechtsDrehen {zurück zum Ausgangspunkt}
36 Schritt (4)
37 UmDrehen
38 *Anweisung
39
40 {Hier startet das eigentliche Hauptprogramm}
41 H {zuerst Buchstabe H schreiben}
42 wiederhole 5 mal {drei Schritte auf die Seite machen, }
43 SeitSchrittLinks {damit ich Platz für Buchstaben i bekomme}
44 *wiederhole
45 i {Buchstabe I schreiben}
```

On the right is a 3D perspective view of a grid world. A small robot figure stands on a grid. A yellow path is drawn on the grid, forming the shape of the letter 'H'. A blue arrow labeled 'N' points upwards, indicating North. The grid is enclosed in a dashed blue wireframe box.

Computational Thinking

3. Abstraktion

- Unwichtige Aspekte des Problems werden ausgeblendet. Der Fokus liegt auf den relevanten Details. Das Ausgangsproblem wird stark vereinfacht.

4. Algorithmen

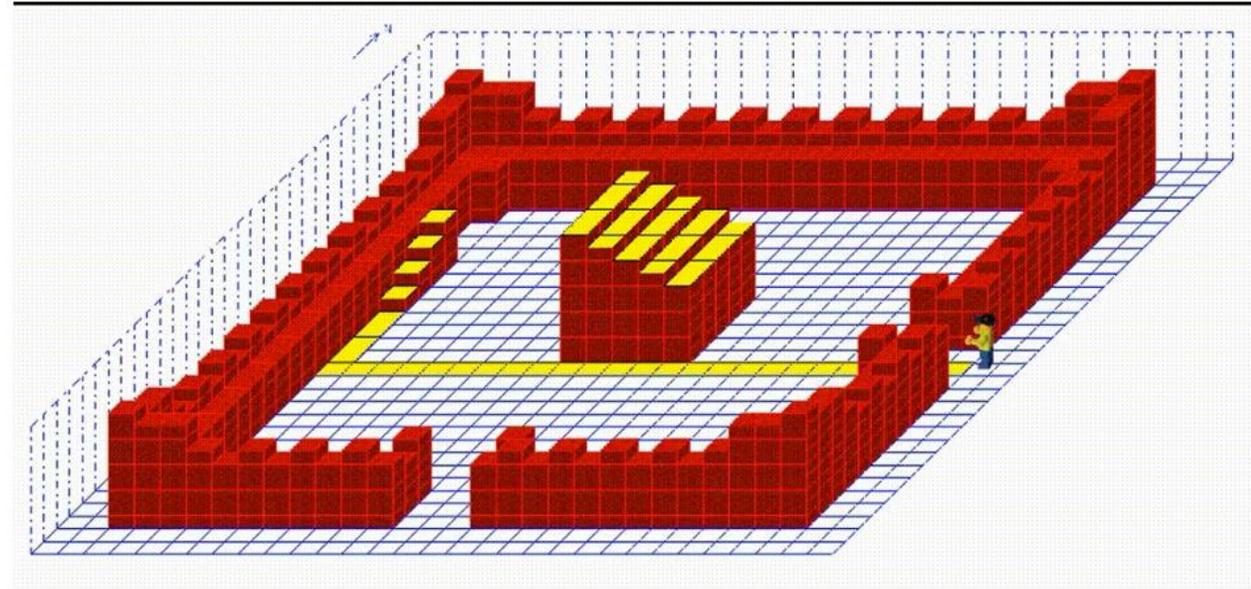
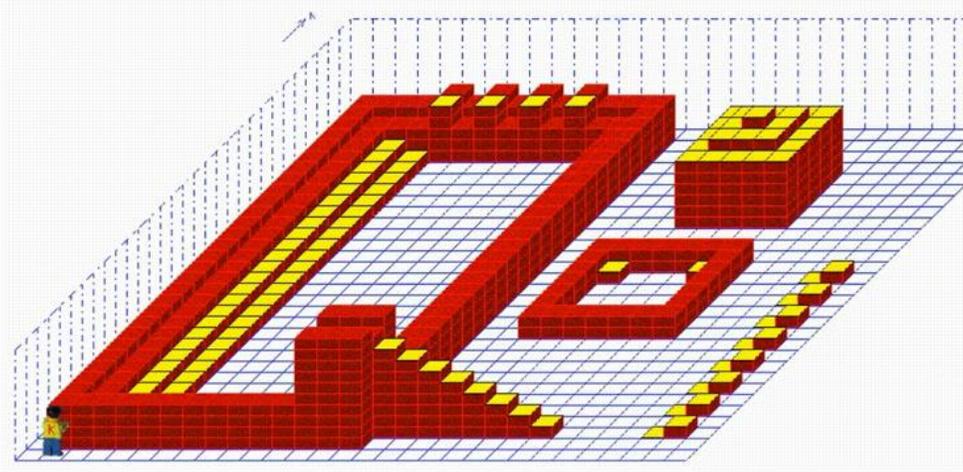
- Durch die vorangehenden Denkschritte wissen wir, nach welchen Mustern das Problem funktioniert und wie wir diese beeinflussen können. Nun können Regeln respektive Algorithmen definiert werden, die zur Problemlösung führen.

```
1 {Unterprogramm für das Ablegen}
2 Anweisung Ablegen
3 wiederhole solange NichtIstZiegel
4 wenn IstWand dann LinksDrehen
5 sonst
6 wiederhole solange NichtIstWand
7 wenn IstWand dann LinksDrehen
8 sonst Hinlegen(grün)
9 Schritt
10 *wenn
11 wenn IstWand dann LinksDrehen
12 Schritt
13 sonst Schritt
14 *wenn
15 *wiederhole
16 *wiederhole
17 *Anweisung
18
19 {Unterprogramm für das Aufnehmen}
20 Anweisung Aufnehmen
21 wiederhole solange IstZiegel
22 wenn IstWand dann LinksDrehen
23 sonst
24 wiederhole solange NichtIstWand
25 wenn IstWand dann LinksDrehen
26 sonst Aufheben
27 Schritt
28 *wenn
29 wenn IstWand dann LinksDrehen
30 Schritt
31 sonst Schritt
32 *wenn
33 *wiederhole
34 *wenn
35 LinksDrehen
36 *wiederhole
37 *Anweisung
38
39 {Hauptprogramm}
40 Ablegen
41 Aufnehmen
```

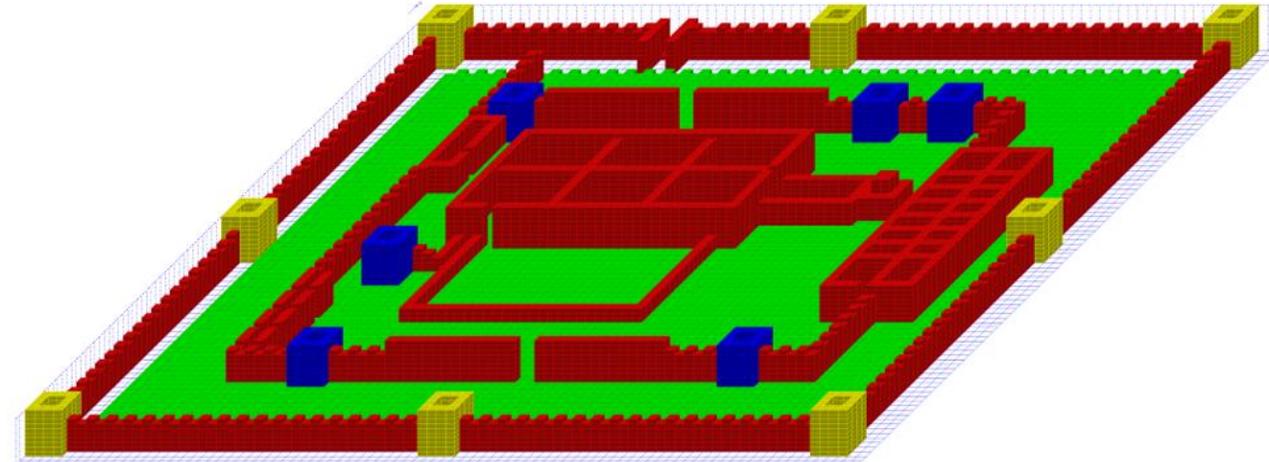
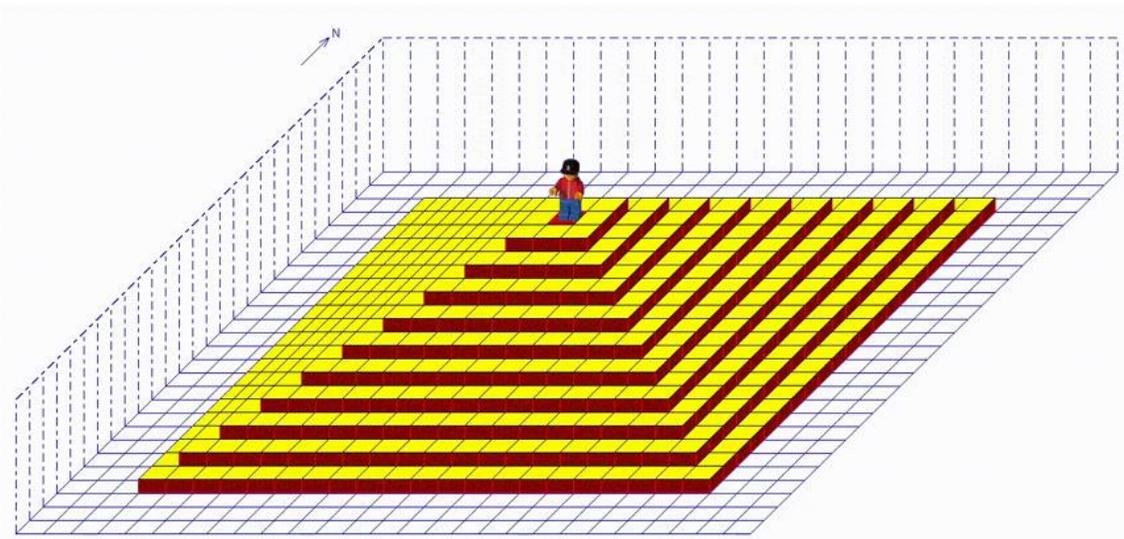
```
1 {Unterprogramm für das Ablegen}
2 Anweisung Ablegen
3 wenn NichtIstWand
4 dann Hinlegen(blau)
5 Schritt
6 wenn NichtIstWand
7 dann Schritt
8 sonst LinksDrehen
9 Schritt
10 *wenn
11 sonst LinksDrehen
12 Schritt
13 *wenn
14 *Anweisung
15
16 {Unterprogramm für das Aufnehmen}
17 Anweisung Aufnehmen
18 wenn NichtIstWand
19 dann Aufheben
20 Schritt
21 wenn NichtIstWand
22 dann Schritt
23 sonst LinksDrehen
24 Schritt
25 *wenn
26 sonst LinksDrehen
27 Schritt
28 *wenn
29 *Anweisung
30
31 {Hauptprogramm}
32 wiederhole 13 mal
33 Ablegen
34 *wiederhole
35 Schritt
36 LinksDrehen
37 wiederhole 13 mal
38 Aufnehmen
39 *wiederhole
40 Schritt
41 LinksDrehen
```

```
1 {Unterprogramm für das Ablegen}
2 Anweisung Ablegen
3 wiederhole solange NichtIstZiegel
4 wenn IstWand dann LinksDrehen
5 sonst
6 wiederhole solange NichtIstWand
7 wenn IstWand dann LinksDrehen
8 sonst Hinlegen(grün)
9 Schritt
10 *wenn
11 wenn IstWand dann LinksDrehen
12 Schritt
13 sonst Schritt
14 *wenn
15 *wiederhole
16 *wiederhole
17 *Anweisung
18
19 {Unterprogramm für das Aufnehmen}
20 Anweisung Aufnehmen
21 wiederhole solange IstZiegel
22 wenn IstWand dann LinksDrehen
23 sonst
24 wiederhole solange NichtIstWand
25 wenn IstWand dann LinksDrehen
26 sonst Aufheben
27 Schritt
28 *wenn
29 wenn IstWand dann LinksDrehen
30 Schritt
31 sonst Schritt
32 *wenn
33 *wiederhole
34 *wenn
35 LinksDrehen
36 *wiederhole
37 *Anweisung
38
39 {Hauptprogramm}
40 Ablegen
41 Aufnehmen
```

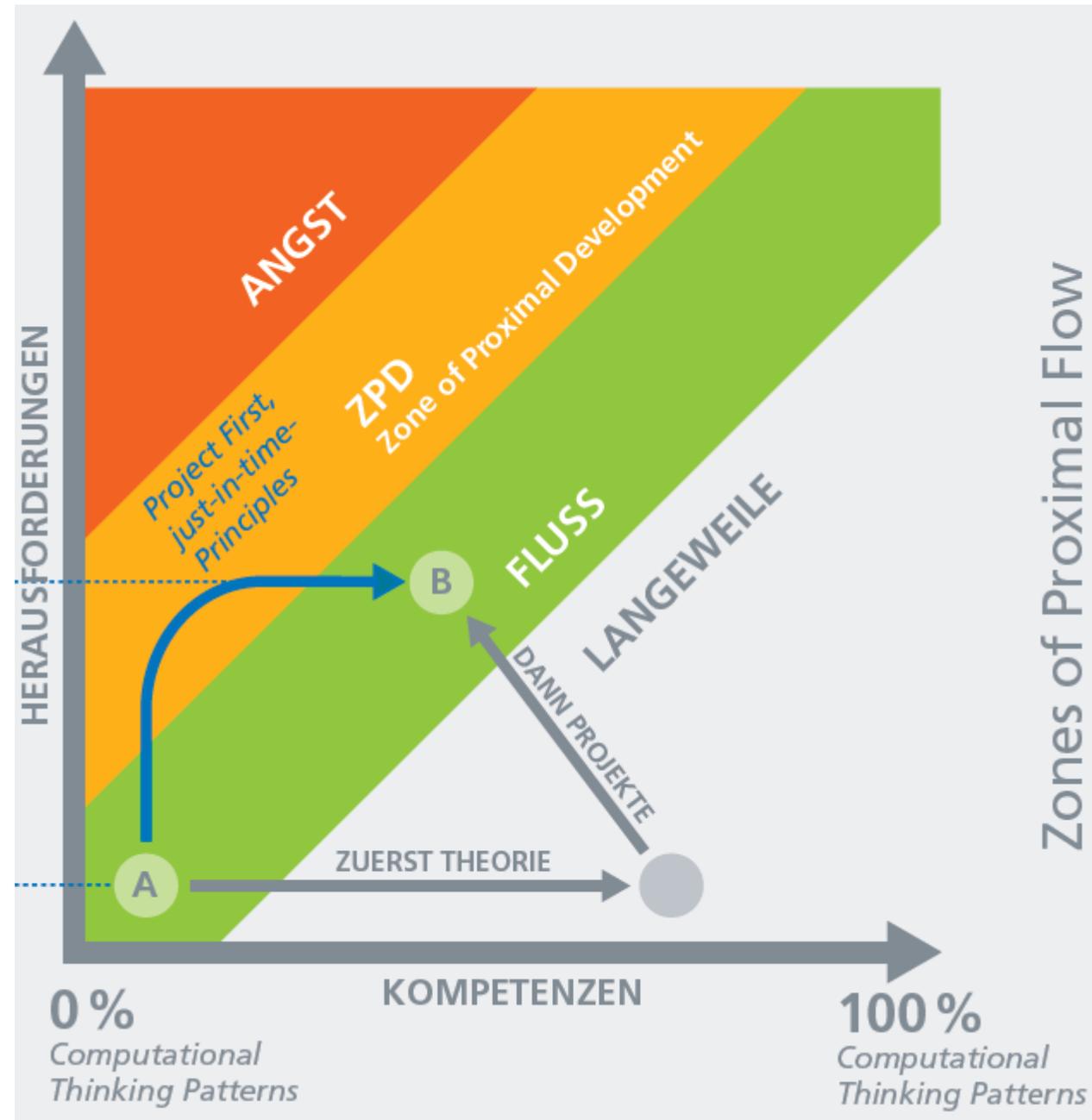
Ideen



<https://mebis.bycs.de/beitrag/robot-karol>



Vorgehen



Scratch



- 2007 veröffentlicht
- 3-Klausel BSD-Lizenz (OpenSource)
- Windows, macOS, Linux
- läuft seit 2019 auch mit JavaScript im Webbrowser

- ca. 90.000.000 registrierte Accounts
- ca. 102.000.000 Projekte veröffentlicht

- wenige Features, für Kinder gedacht
 - ❖ z.B. Scratch Jr. Ab 5 Jahren



&

Snap!



- Erscheinungsjahr 2011
- OpenSource-Lizenz
- Windows, macOS, Linux
- läuft als BYOB im Webbrowser (JavaScript)

- ca. 600.000 registrierte Accounts
- ca. 5.000.000 Projekte

- Weiterentwicklung von Scratch mit mehr Features
 - ❖ BYOB = eigene Funktionen; Rekursion möglich
 - ❖ Listen, Sprites und Funktionen sind first-class, d.h. können in anderen Listen gespeichert werden
 - ❖ externe Ansteuerung möglich Lego, Arduino, ...
 - ❖ ausgefeilte Parameterübergabe – mächtiger als manche Text-Programmierungsumgebung

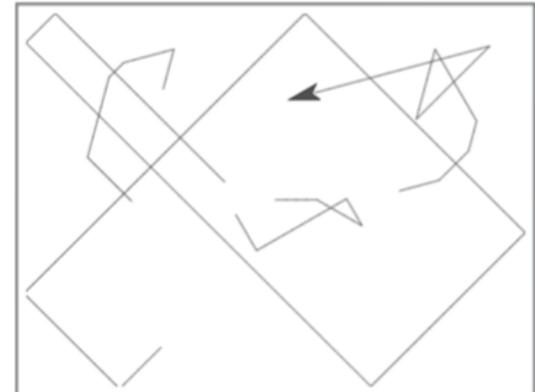


Direktmodus bei beiden Sprachen möglich

The image displays the Scratch Malstift extension interface. On the left, a sidebar lists categories: Bewegung, Aussehen, Klang, Ereignisse, Steuerung, Fühlen, Operatoren, Variablen, Meine Blöcke, and Malstift. The Malstift category is selected, showing a list of green code blocks: lösche alles, hinterlasse Abdruck, schalte Stift ein, schalte Stift aus, setze Stiftfarbe auf (with a color picker), ändere Stift Farbe um 10, setze Stift Farbe auf 50, ändere Stiftdicke um 1, and setze Stiftdicke auf 1. The main workspace shows a sequence of blue code blocks: gehe 10 Schritte, drehe dich um 15 Grad, drehe dich um 15 Grad, pralle vom Rand ab, and schalte Stift ein. A Scratch cat character is visible in the workspace. Below the workspace, an 'Erweiterung auswählen' (Select Extension) menu is open, showing options for Musik, Malstift, Video-Erfassung, Text zu Sprache, Übersetzen, and Makey Makey.

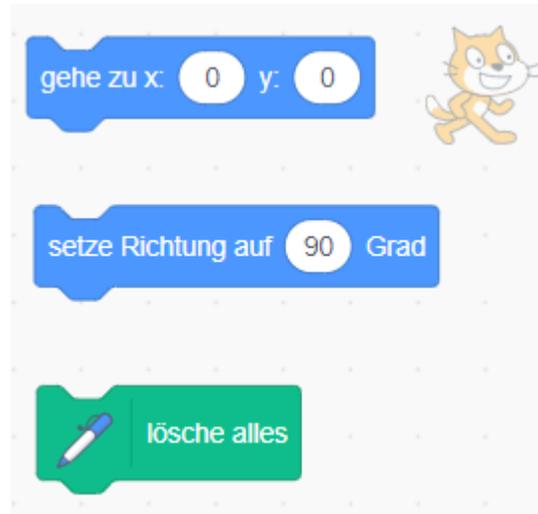
Stift runter

Stift hoch



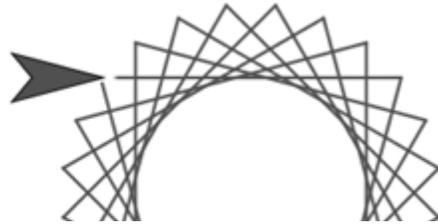
Malstift = Erweiterung (microbit, Lego, ...)

Zurücksetzen

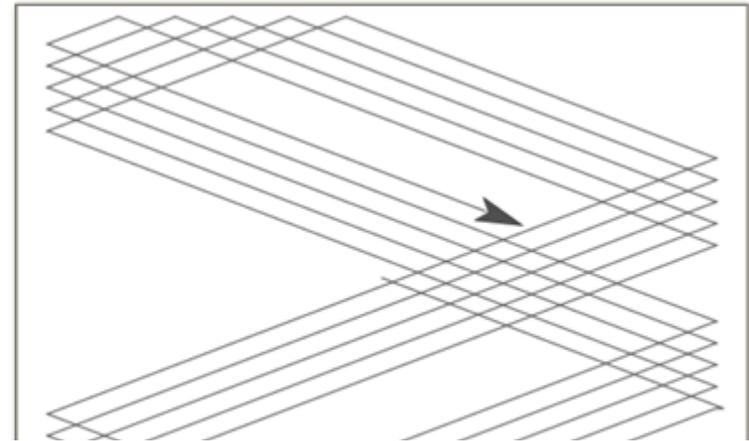
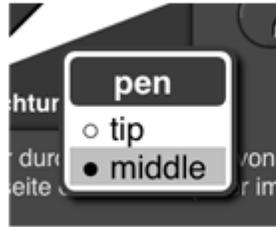


Schleifen

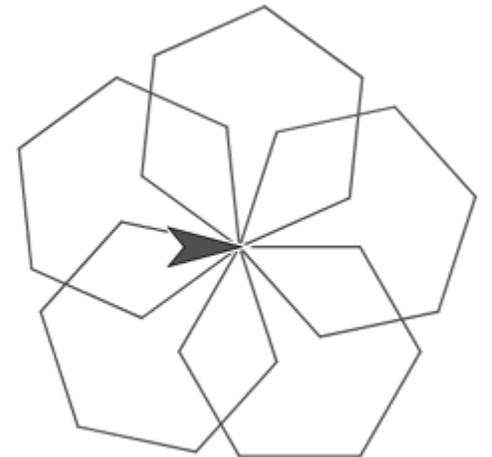
```
Stift runter
wiederhole 24 mal
  gehe 100 Schritte
  drehe 105 Grad
```



```
Stift runter
drehe 21 Grad
fortlaufend
  gehe 10 Schritte
  pralle vom Rand ab
```

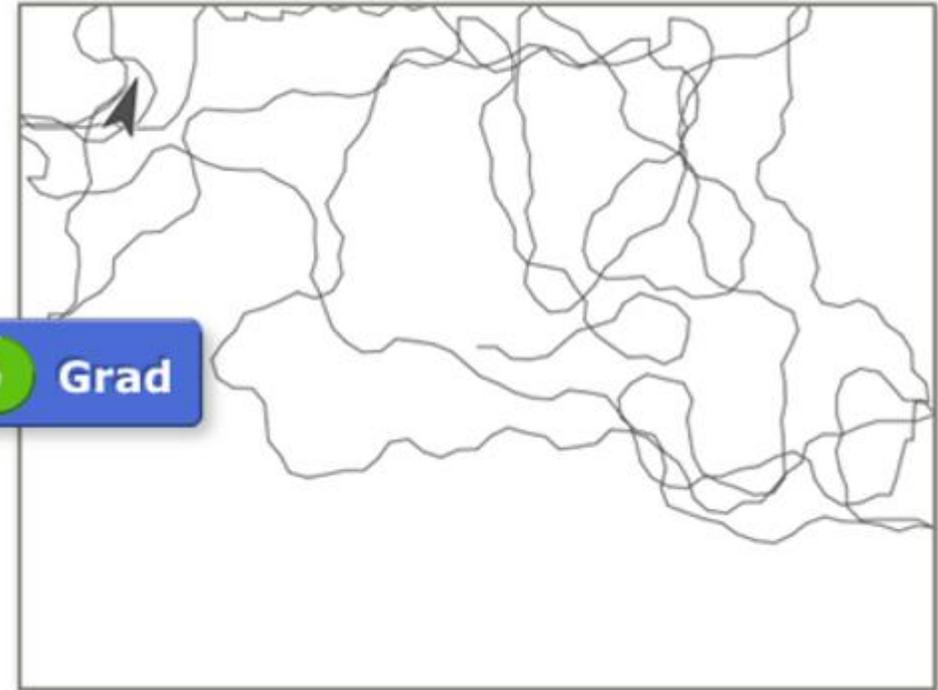


```
wiederhole 5 mal
  wiederhole 6 mal
    gehe 50 Schritte
    drehe 60 Grad
  drehe 72 Grad
```



Zufall

```
wiederhole 500 mal  
  gehe 10 Schritte  
  drehe Zufallszahl von -50 bis 50 Grad  
  pralle vom Rand ab
```



Wie funktioniert das?



setze Stiftdicke auf 1

wiederhole 60 mal

- gehe 5 Schritte
- drehe 5 Grad

ändere Stiftdicke um 1

In TSNmoodle ist Snap! direkt in der Aktivität „Aufgabe“ hinterlegt

Abgabetypen

- Abgabetypen
- Global credentials
- Mahara web services OAuth secret
- Archive when graded
- Snap! Distribution
- Snap! Cloud-Funktionen
- Snap! Sprache

Mahara ? Dateiabgabe ? Texteingabe online ?

Mahara credentials have been set globally

Snap! ?

Snap! Cloud aktivieren ?

Moodle-Sprache verwenden ?

Nein ⇅

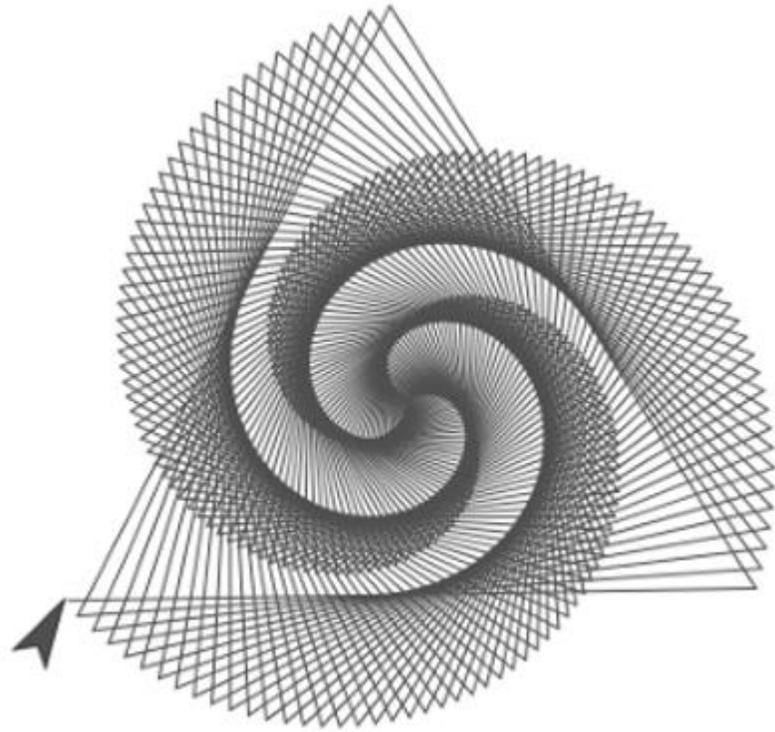
Snap! - <https://snap.berkeley.edu/snap/snap.html> ⇅



Farben



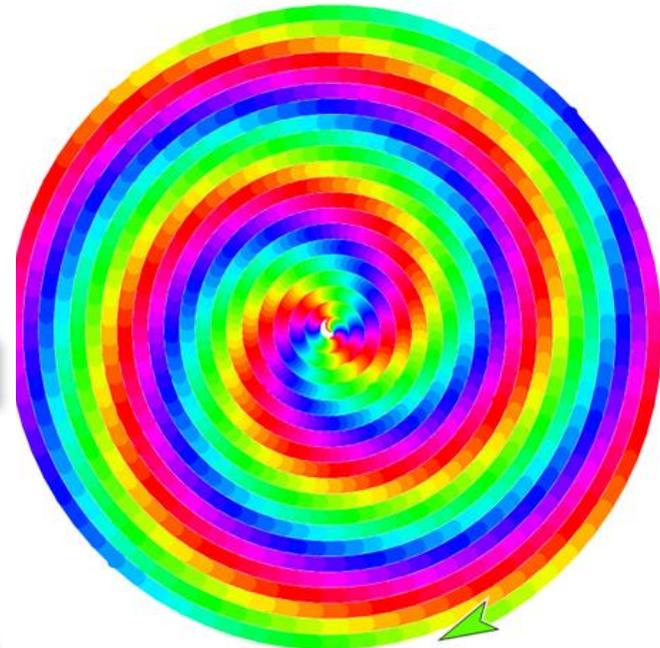
Mit Variablen – wie viele Zeilen?



```
for i = 1 to 300
  gehe i Schritte
  drehe 121 Grad
```



```
gehe zu x: 0 y: 0
zeige Richtung 90
wische
setze Stiftfarbe auf
setze Stiftdicke auf 8
Stift runter
Skriptvariablen Schritt
setze Schritt auf 3
wiederhole bis Schritt > 15
  gehe Schritt Schritte
  drehe 5 Grad
  ändere Schritt um 0.01
  ändere Stiftfarbe um 3
```



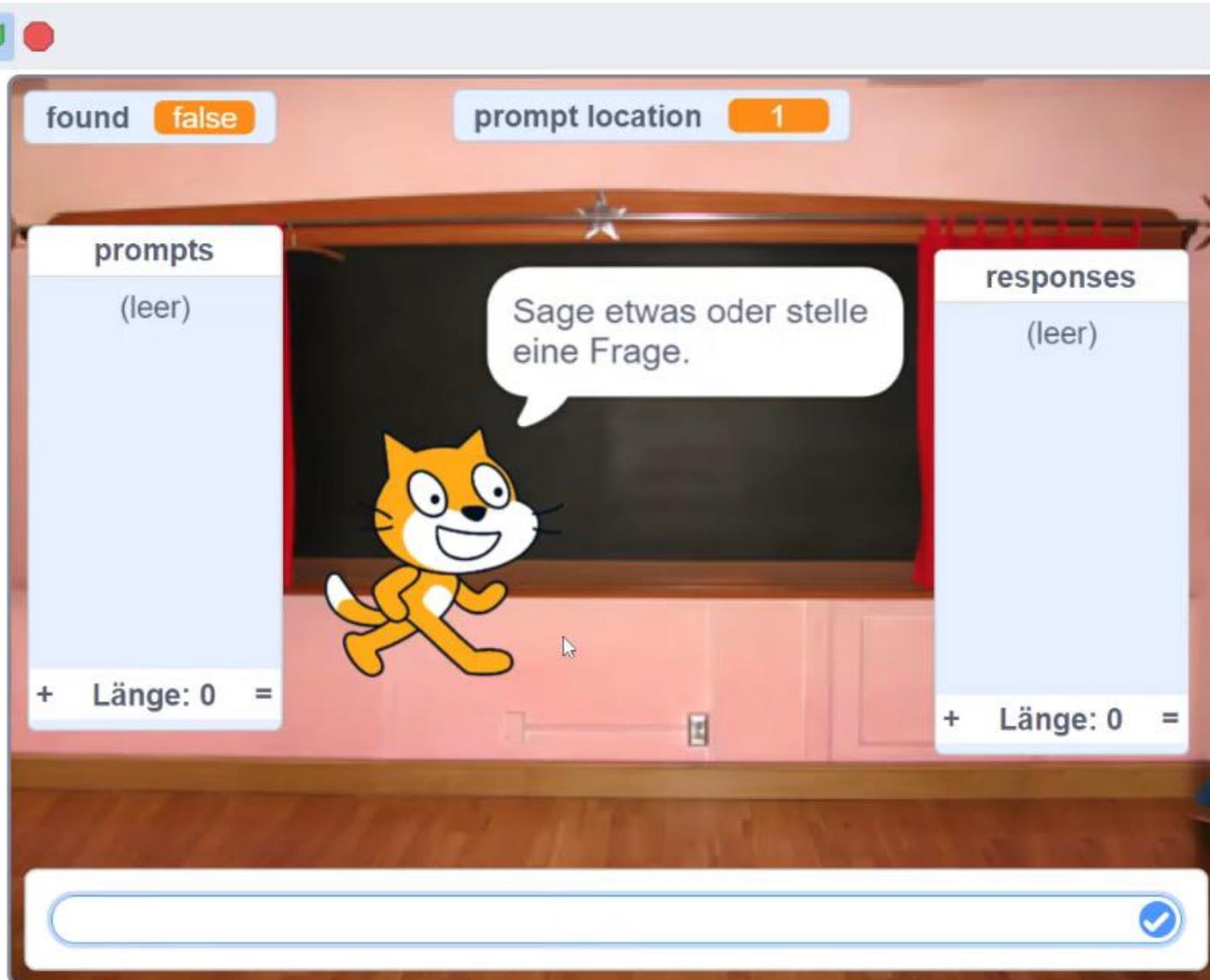
Stempeln



```
Wenn  angeklickt
  wische
  Skriptvariablen Abstand
  setze Abstand auf 30
  setze y auf 160
  wiederhole 6 mal
    setze Größe auf  $200 - y\text{-Position} \times 0.8$  %
    setze x auf -200
    wiederhole 20 mal
      falls Zufallszahl von 1 bis 5 > 1
        nächstes Kostüm
      stemple
      ändere x um Abstand
    ändere y um  $0 - \text{Abstand}$ 
  ändere Abstand um 20
```



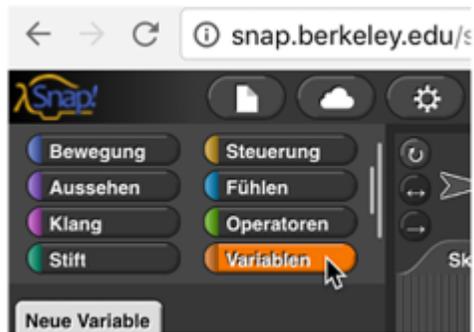
ChatBot in Scratch



```
Wenn angeklickt wird
wiederhole fortlaufend
  frage Sage etwas oder stelle eine Frage. und warte
  setze found auf false
  setze prompt location auf 1
  wiederhole Länge von prompts mal
    falls Antwort enthält Element prompt location von prompts ?, dann
      falls nicht prompts enthält Antwort ?, dann
        füge Antwort zu prompts hinzu
        füge Element prompt location von prompts zu responses hinzu
      setze found auf true
      sage Element prompt location von responses für 2 Sekunden
    sonst
      ändere prompt location um 1
  }
  falls found = false, dann
    füge Antwort zu prompts hinzu
    frage Entschuldigung. Ich weiß momentan keine Antwort. Was soll ich das nächste Mal darauf antworten? und warte
    füge Antwort zu responses hinzu
```

<https://www.create-learn.us/blog/make-chatbot-with-ai-in-scratch/>

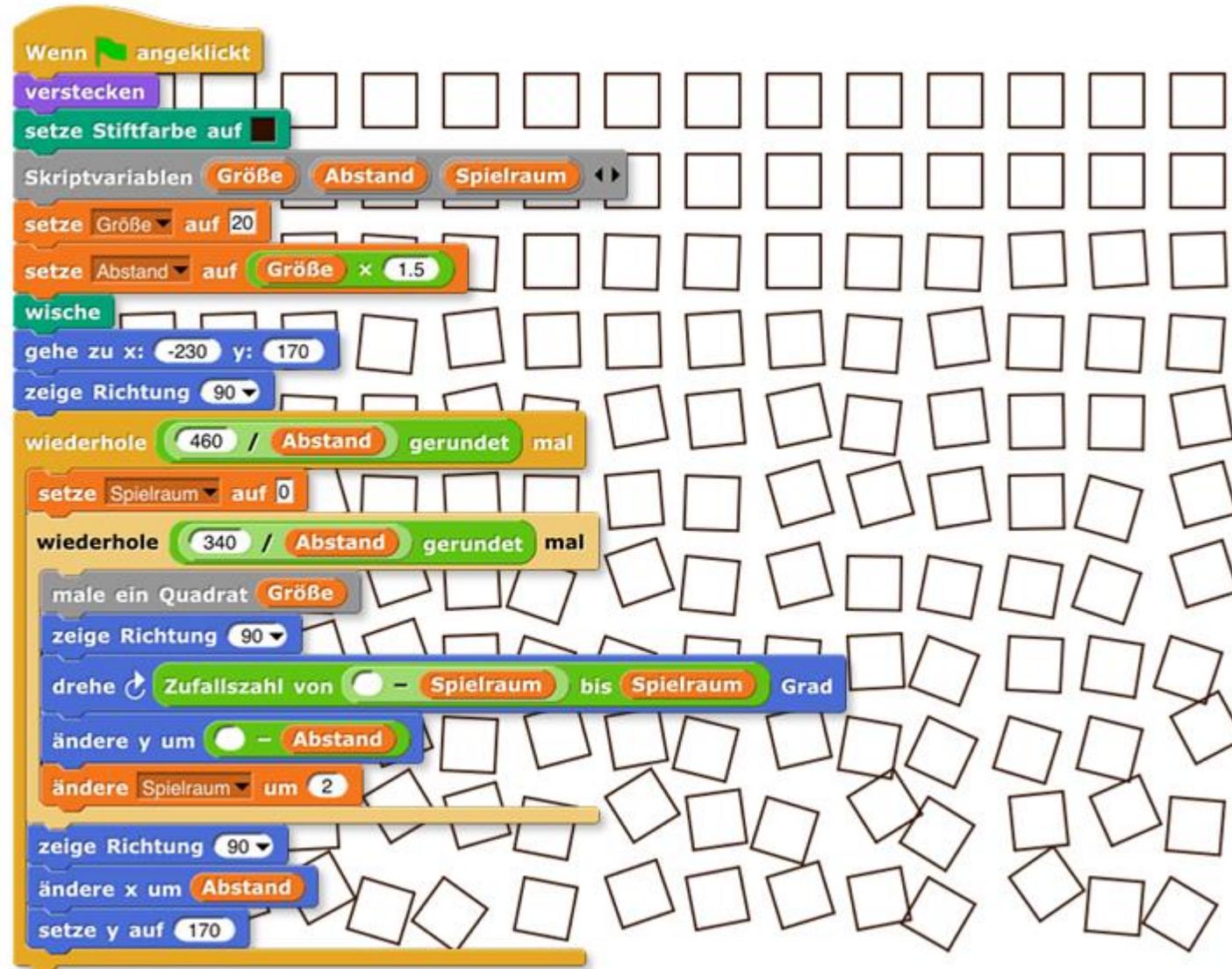
Eigene Blöcke



```
Wenn  angeklickt
zeige Richtung 90
gehe zu x: -200 y: 0
wische
Skriptvariablen Seiten
setze Seiten auf 1
wiederhole 7 mal
  ändere Seiten um 1
  Spirale Schritt: 25 Faltungen: Seiten
wiederhole 6 mal
  ändere Seiten um -1
  Spirale Schritt: 25 Faltungen: Seiten
```

```
Blockeditor
+Spirale+Schritt: Schritt # = 30 +Faltungen:+ Faltungen # = 6 +
Skriptvariablen delta
setze delta auf Schritt / Faltungen
wiederhole Faltungen mal
  gehe Schritt Schritte
  drehe 90 Grad
  ändere Schritt um delta x -1
  gehe delta Schritte
  wiederhole Faltungen mal
    ändere Schritt um delta
    drehe 90 Grad
    gehe Schritt Schritte
  drehe 90 Grad
  gehe Schritt Schritte
  drehe 90 Grad
OK Anwenden Abbrechen
```

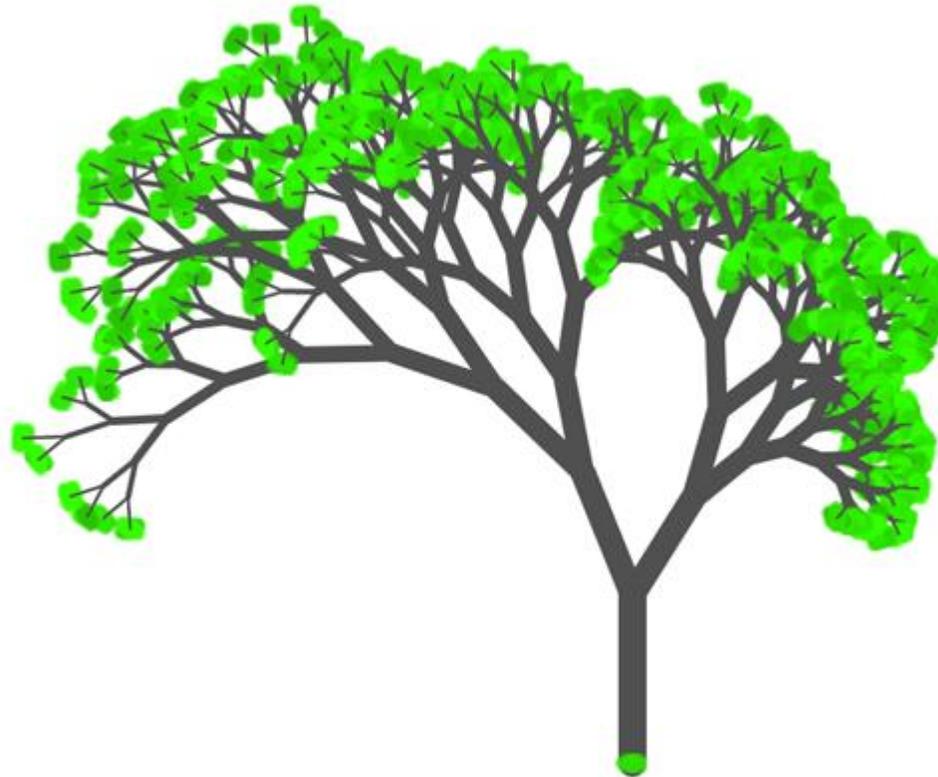
Eigene Blöcke



```

+ Baum + Level + Level # = 3 + Länge + Länge # = 80 +
falls Level > 0
  ziehe Kostüm Richtungszeiger an
  setze Stiftdicke auf Level
  Skriptvariablen links rechts
  setze links auf Zufallszahl von 15 bis 25
  setze rechts auf Zufallszahl von 35 bis 60
  gehe Länge Schritte
  drehe links Grad
  Baum Level Level - 1 Länge
  Länge x Zufallszahl von 0.7 bis 0.95
  drehe rechts Grad
  Baum Level Level - 1 Länge
  Länge x Zufallszahl von 0.5 bis 0.9
  drehe rechts Grad
  drehe links Grad
  gehe Länge x -1 Schritte
sonst
  ziehe Kostüm Blatt an
  setze Helligkeit -Effekt auf Zufallszahl von 1 bis 30
  stemple

```



```

Wenn Flagge angeklickt
  gehe zu x: 0 y: -150
  zeige Richtung 0
  wische
  Stift runter
  Baum Level 10 Länge 60

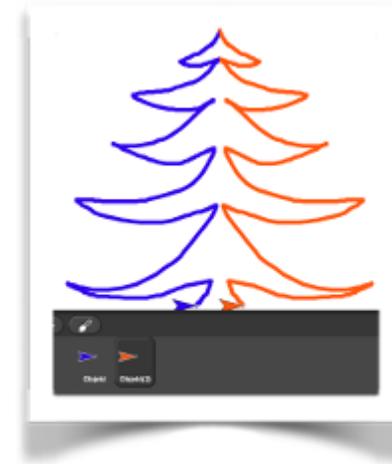
```

Interaktiv

```
Wenn  angeklickt  
wische  
fortlaufend  
gehe zu  Mauszeiger  
falls  Maustaste gedrückt?  
Stift runter  
sonst  
Stift hoch
```



```
Wenn  angeklickt  
wische  
fortlaufend  
gehe zu x:  - Maus x-Position y: Maus y-Position  
falls  Maustaste gedrückt?  
Stift runter  
sonst  
Stift hoch
```

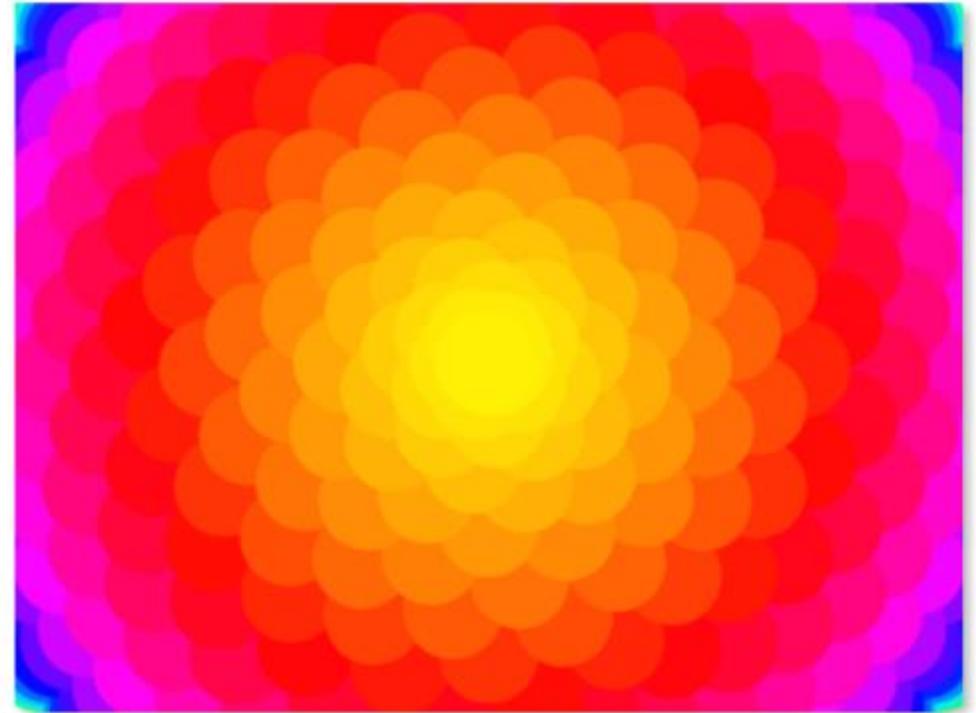


Klonen

```
Wenn  angeklickt  
entferne diesen Klon
```

```
Wenn  angeklickt  
schalte Grafikeffekte aus  
fortlaufend  
ändere Farbe -Effekt um   
drehe  Grad  
gehe zu x:  y:   
klone mich
```

```
Wenn ich geklont werde  
Skriptvariablen speed  
setze speed auf   
wiederhole bis speed <   
gehe speed Schritte  
setze speed auf speed ×  /   
entferne diesen Klon
```





V I E L E
N D A N K
F U E E R D

Bild- und Quellennachweis

Seymour Papert: Folie 2 - https://de.wikipedia.org/wiki/Seymour_Papert#/media/Datei:Papert.jpg

Jeannette Wing: Folie 2 -

https://de.wikipedia.org/wiki/Jeannette_Wing#/media/Datei:Jeannette_Wing,_Davos_2013.jpg

Wikipedia: Folie 3 - https://en.wikipedia.org/wiki/Computational_thinking

Swisscom.ch: Folie 5 - <https://www.swisscom.ch/de/schulen-ans-internet/computational-thinking.html>

chatGPT: Folie 4 - <https://chat.openai.com/chat>

ZPT: Folie 11 - Hasler Broschüre; Hasler Stiftung; Schriftenreihe Januar 2015; Computational Thinking in der Lehrerfortbildung

Kunst mit Snap! - https://eeducation.at/fileadmin/user_upload/Kunst_mit_Snap_JadgaHuegle.pdf

StarWars TextEngine: Folie 27 - <https://scratch.mit.edu/projects/364079313>

Robot Karol: <https://mebis.bycs.de/beitrag/robot-karol>