

WORLD ROBOT OLYMPIAD™



SWITZERLAND

**ROBOTIK**  
**FÜR FORTGESCHRITTENE**  
**VARIABLEN UND**  
**EIGENE BLÖCKE**

© 2023 Verein World Robot Olympiad Schweiz  
Offizielle Ausrichterin der World Robot Olympiad in der Schweiz

---

## INHALTSVERZEICHNIS

1	Einleitung: Schleifen .....	3
2	Variable .....	5
3	Eigene Blöcke .....	7
4	Übungsaufgabe zur Vorbereitung auf einen WRO-Wettbewerb .....	11
	Material: .....	11
	Startaufstellung: .....	12
	Teilaufgabe 1: .....	12
	Teilaufgabe 2: .....	12
	Bewertungstabelle: .....	12
	Überlegung zum Lösen dieser Aufgabe .....	13
	Struktur .....	13
	Schritt 1: Aus dem Startbereich fahren bis zum Personenfeld 1 .....	14
	Schritt 2 und 5: Person einladen und Farbe erkennen .....	14
	Schritte 3 und 6: Zu einem der drei Zielbereiche fahren und die Person ausladen .....	15
	Teilaufgabe 2 .....	16
	Ergebnis Hauptstapel: .....	16

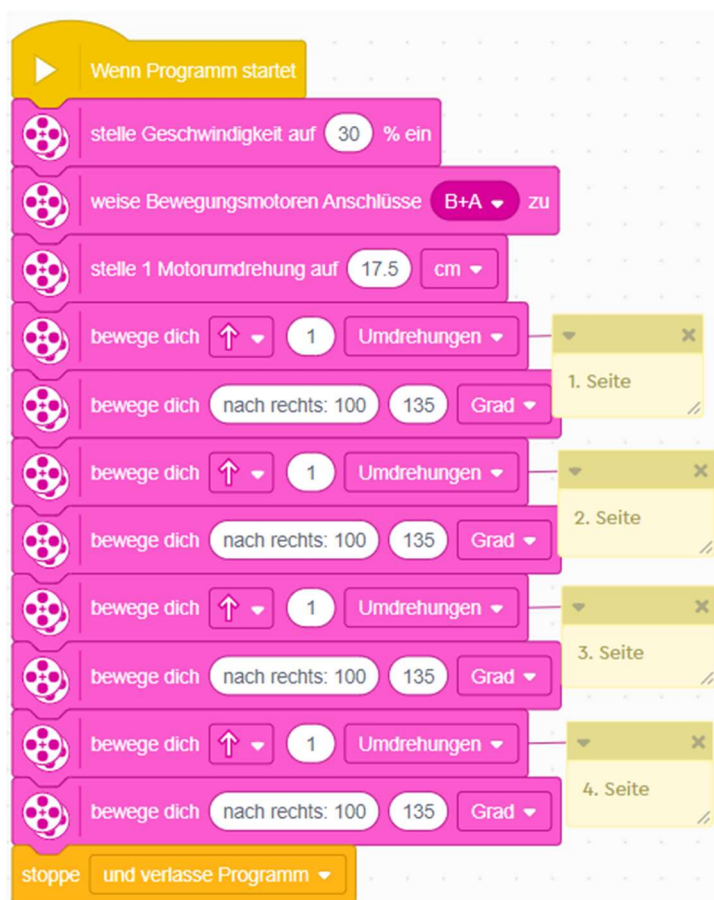
## 1 EINLEITUNG: SCHLEIFEN

Eine Grundregel beim Programmieren lautet:

«Schreibe nie etwas zweimal, wenn du es einmal schreiben kannst.»

Schauen wir uns ein ganz einfaches Beispiel an:

Du willst mit deinem Roboter ein Quadrat fahren. Ohne diese Grundregel sieht der Code so aus:



Wenn du jetzt beim Testen merkst, dass der Block «**Bewege dich nach rechts:100 – 135 Grad**» keinen rechten Winkel ergibt, sondern entweder etwas zu viel oder etwas zu wenig<sup>1</sup>, dann musst du an 4 Stellen korrigieren. Da schleichen sich schnell Fehler ein, und es ist schwierig, die zu finden.

Deshalb ist es sinnvoll, zuerst zu analysieren, welche Teile vom Code sich wiederholen, und diese zu testen ohne den ganzen Rest des Codes.

<sup>1</sup> Welcher Wert hier richtig ist hängt davon ab,

- wie gross deine Räder sind
- wie gross der Abstand zwischen den Rädern sind
- wie schnell du fährst (wegen der Trägheit musst du bei langsamen Geschwindigkeiten tendenziell einen etwas grösseren Wert nehmen als bei höheren Geschwindigkeiten).

Der Teil, der wiederholt wird, ist bei unserem Quadrat:



Wir können diese beiden Zeilen direkt unter den Kopfteil<sup>2</sup> des Codes hängen und testen und, falls nötig, noch korrigieren. Wenn wir zufrieden sind, können wir diese beiden Zeilen wiederholen. Wir verwenden dazu den Schleifen-Block. Unser Code wird jetzt viel kürzer:



<sup>2</sup> Der Kopfteil ist der Teil, der bei allen Fahrprogrammen gleichbleibt:



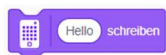
## 2 VARIABLEN

Eine **Variable** ist ein Wert, der beim Programmieren verwendet wird. Eine Variable kann beliebig oft verwendet werden. Eine Variable kann im Laufe des Codes verändert werden.

Schauen wir uns das mit dem Beispiel unseres Quadrates an:

Wir wollen, dass die Lichtmatrix auf unserem Roboter immer anzeigt, die wievielte Seite er gerade fährt. Also soll entlang der ersten Seite eine «1» dort stehen, entlang der zweiten Seite eine «2», und so weiter.

Der Block, den du hierfür brauchst, ist der «**Schreiben**»-Block:



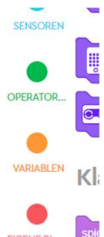
Du könntest also am Start der ersten Seite den Block so einfügen:



Aber jetzt können wir die Schleife nicht mehr brauchen, da sich die Nummer der Seite ja ändern soll, der Rest des Codes aber gleichbleiben!

Hier hilft die Variable.

1. Gehe in der Block-Palette zu den dunkel-orangen «Variable»-Blöcken.



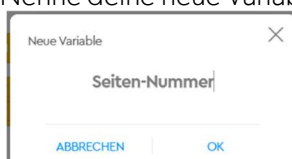
2. Klicke auf «Neue Variable»


### Variablen

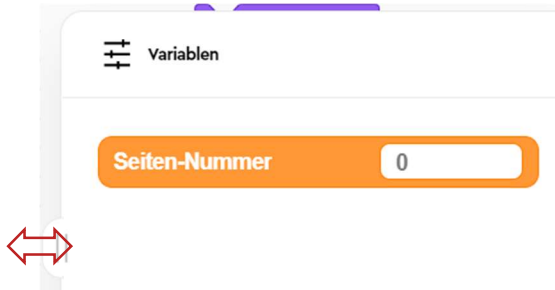
Neue Variable

Neue Liste

3. Nenne deine neue Variable «Seiten-Nummer» und klicke dann auf OK.



4. Am rechten Rand geht jetzt ein kleines Fenster auf, in dem alle Variable, die du definiert hast, angezeigt werden. Du kannst dieses Fenster auf- und zu-klicken, damit es dich nicht stört. Klicke dazu auf die linke Kante des «Griffs» von diesem Fenster (hier dargestellt durch den roten Doppelpfeil ).

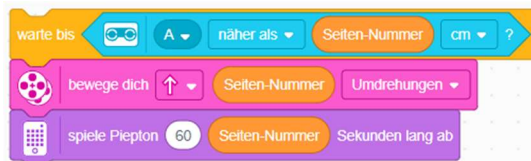


5. Hier siehst du immer den aktuellen Wert deiner Variablen. Wenn die App mit dem Roboter verbunden ist, kannst du beim Ausführen des Programmes immer genau sehen, welchen Wert die Variable gerade hat und wie er sich verändert. Das ist nützlich zum Kontrollieren, ob alles funktioniert.
6. Du siehst nun in der Variablen-Palette drei neue Blöcke, die vorher nicht da waren:

### Variablen



- «**Seiten-Nummer**»: Du kannst diesen Block überall einfügen, wo es ein rundes Eingabefeld hat. Hier sind ein paar Beispiele. Die sind nicht unbedingt immer sinnvoll, aber du siehst, was möglich ist:



Überall, wo die Variable «Seiten-Nummer» eingetragen ist, wird der aktuelle Wert dieser Variable verwendet.

- «**Setze Seiten-Nummer auf ...**»: Mit diesem Block gibst du der Variablen einen festen Wert. In der Regel wird hier der Startwert der Variable festgelegt. Diesen Wert hat die Variable ganz am Anfang. Meistens ist 0 der richtige Wert hier, so auch bei unserem Quadrat.
- «**Ändere Seiten-Nummer um ...**»: Diesen Block brauchen wir für unseren Seiten-Zähler. Er setzt nicht einen fixen neuen Wert, sondern er gibt eine Regel an, wie der Wert automatisch verändert werden soll: «**Ändere Seiten-Nummer um 1**» bedeutet: zähle zum bisherigen Wert +1 hinzu. Übrigens: Wenn du eine Minus-Zahl in

das Feld schreibst, wird der Wert der Variable entsprechend kleiner. Das ist z.B. für einen Countdown nützlich.

- Füge direkt nach dem Kopfteil deines Codes, noch vor der Schleife, den Block «**Setze Seiten-Nummer auf 0**» ein.
- Hänge nun den Block «**Ändere Variable um 1**» vor deine zwei Blöcke, mit denen du die Seite und die Ecke des Quadrats beschreibst.



- Füge dann direkt danach den «Schreiben»-Block ein, mit unserer Variablen «Seiten-Nummer» im Eingabefeld.



- Jetzt musst du das ganze nur noch in deine Schleife packen, und fertig ist das Programm!



### 3 EIGENE BLÖCKE

Schleifen sind super, wenn du eine Aktion mehrmals direkt hintereinander wiederholen willst. Du hast auch gelernt, wie du Variablen nutzen kannst. Doch manchmal wird eine Aktion mehrfach gebraucht, aber dazwischen gibt es ganz verschiedene Sachen, die der Roboter tun soll.

Probieren wir das mit unserem Quadrat aus. Wir verwenden den Code von vorhin weiter, dann müssen wir nicht so viel neu schreiben.

Der Roboter soll weiterhin ein Quadrat fahren, aber nach jeder Drehung soll er etwas anderes tun:

- Nach der ersten Ecke soll er dreimal kurz tief piepsen (Ton 60).
- Nach der zweiten Ecke soll er kurz auf der Lichtmatrix ein Smiley anzeigen.
- Nach der dritten Ecke soll er beim Einschaltknopf dreimal rot blinken.
- Nach der vierten Ecke soll dreimal kurz und einmal lang hoch piepsen (Ton 72).

Du siehst: diese Aufgaben sind so verschieden, dass du sie nicht in eine Schleife packen kannst. Bedeutet das, dass wir den Code für die Seite und die Ecke wieder jedes Mal schreiben müssen, mit dem Risiko, dass sich Fehler einschleichen? Nein! Es gibt eine viel cleverere Lösung:

Wir können die Seite und die Drehung sowie den Seitenzähler in einen «**Eigenen Block**» auslagern. Oft werden diese «Eigenen Blöcke» auch mit ihrer englischen Bezeichnung «**MyBlocks**» benannt.

1. Gehe in der Palette zum Bereich «Eigene Blöcke».



2. Klicke dann auf «**Neuer Block**».

### Eigene Blöcke

Neuer Block

3. Es erscheint ein Fenster, bei dem du den Namen deines neuen Eigenen Blocks angeben kannst. Schreibe hier «Seite des Quadrats» hinein und klicke auf «Speichern».

Block erstellen



  
**Eingabe  
hinzufügen**  
Zahl oder Text

  
**Eingabe  
hinzufügen**  
Wahr oder falsch

  
**Kennzeichnung  
hinzufügen**

ABBRECHEN

SPEICHERN

Mit den drei Feldern darunter kannst du Eingabefelder zu deinem Feld hinzufügen, in die du Werte schreiben kannst wie in alle anderen Blöcke auch. Das brauchen wir im Moment noch nicht und schauen es uns später an.

4. Auf deiner Arbeitsfläche erscheint jetzt ein neuer Stapel-Kopf mit dem Namen deines MyBlocks:



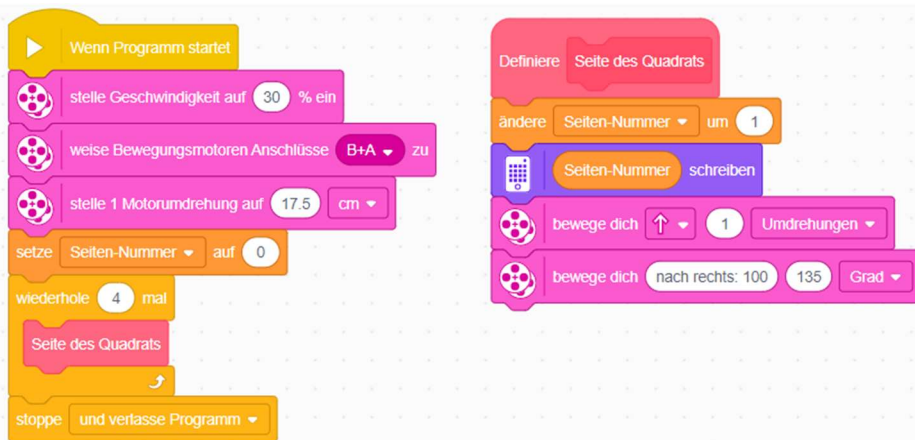


Und in der MyBlocks-Palette ist ein neuer Block aufgetaucht:

### Eigene Blöcke



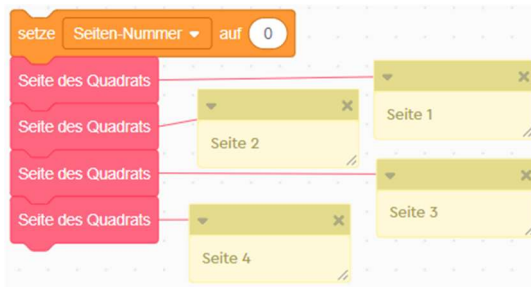
5. Ziehe jetzt alle Blöcke aus der Schleife heraus und hänge sie unter den MyBlock-Kopf. Lass für den Moment die leere Schleife noch im Stapel. Füge dann den «Seite des Quadrats»-MyBlock in die Schleife hinein. Dein Code sieht jetzt so aus:



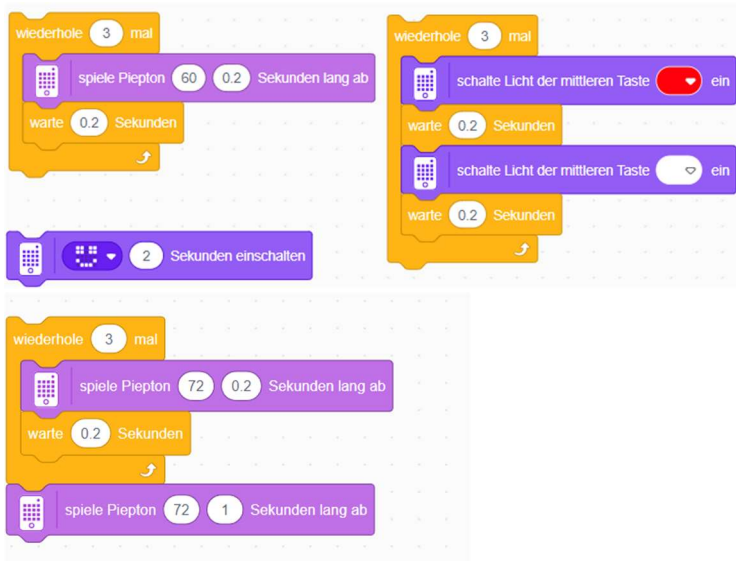
Wenn du das Programm jetzt startest, sollte es genau gleich funktionieren wie vorhin, wo wir nur die Schleife hatten.

6. Im Moment sieht das Programm komplizierter aus als vorhin. Wo ist dann der Nutzen, dass wir den Seitenzähler und die Seite mit Drehung auslagern? Ganz einfach: Schau noch mal in der Aufgabe oben: Jetzt soll der Roboter nach jeder Ecke eine andere Aktion ausführen. Die Schleife können wir dann nicht mehr brauchen. Wir löschen sie darum.
7. Jetzt überlegen wir uns die erforderliche Logik unseres Programms: Wir brauchen immer die Seite des Würfels, dann die Spezial-Aktion. Am einfachsten fügen wir unter dem Block «Setze Seiten-Nummer auf 0» zunächst vier Mal unseren «Seite des Quadrats»-MyBlock ein. Und um es später ein bisschen übersichtlicher zu haben, können wir jeweils noch einen Kommentar<sup>3</sup> dranhängen:

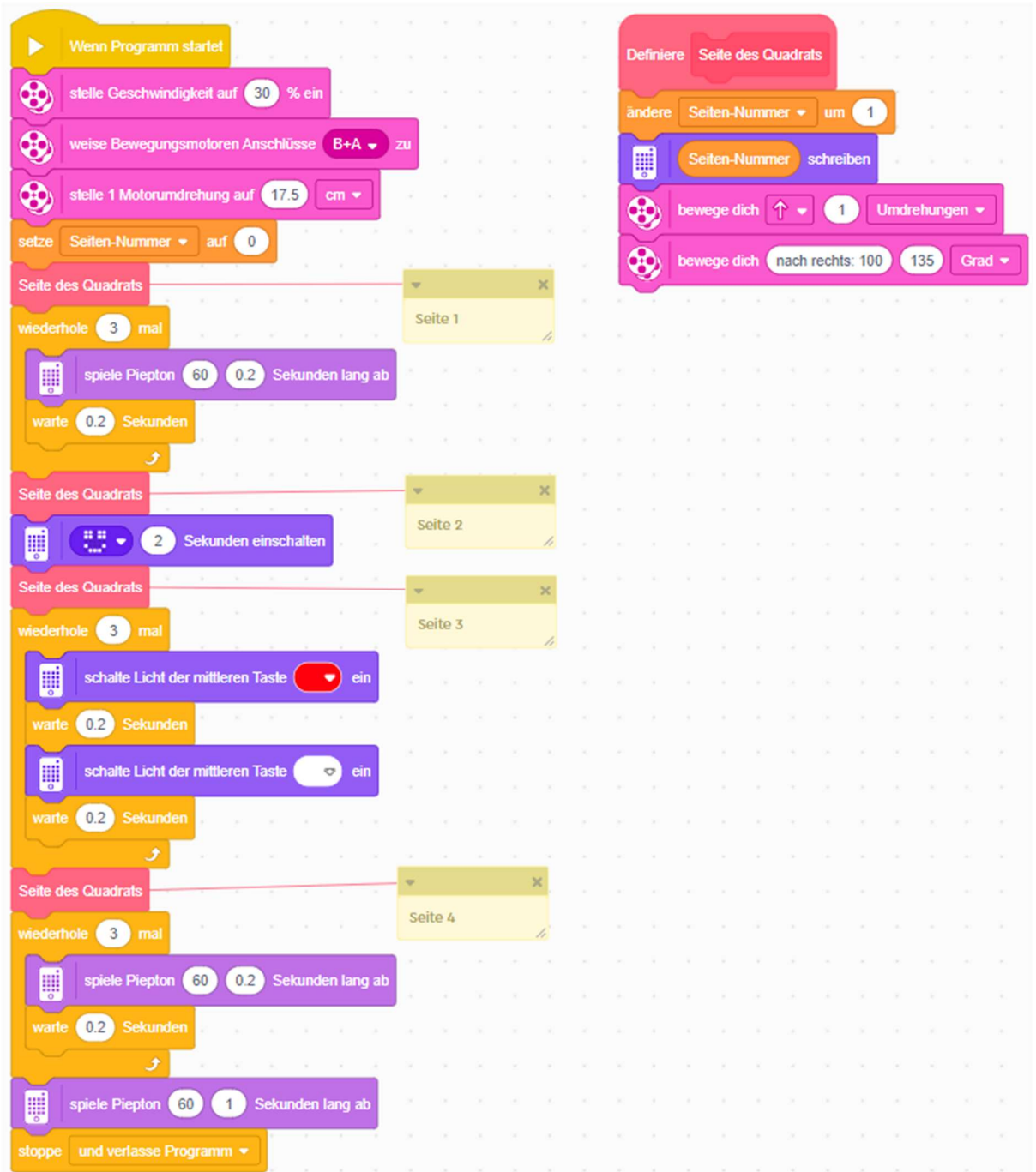
<sup>3</sup> Bei Informatikern gehört es zur Best Practice (dieser Fachausdruck bedeutet: das allgemein als vorbildlich anerkannte Verhalten), Code ausführlich zu kommentieren. Dies hilft dir selbst für die Übersichtlichkeit, damit langer Code etwas besser gegliedert wird. Es ist auch nützlich, damit du später noch weisst, was du da warum gemacht hast. In der SPIKE-App kannst du Kommentare hinzufügen, indem du mit der rechten Maustaste auf einen Block klickst und im Kontextmenü «Kommentar hinzufügen» wählst. Der Kommentar ist dann fest mit diesem Block verbunden. Wenn du im Kommentarfeld das X anklickst, wird der Kommentar gelöscht.



8. Und jetzt schreibst du nach jedem «Seite des Quadrats»-MyBlock den Code der jeweiligen Spezialaktion. Diese vier Aktionen sehen so aus:



9. Geschafft! Dein fertiges Programm sollte jetzt so aussehen:

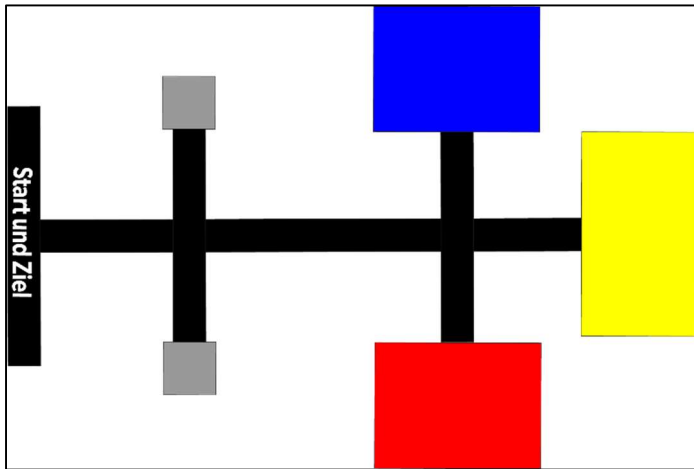


## ÜBUNGSAUFGABE ZUR VORBEREITUNG AUF EINEN WRO-WETTBEWERB

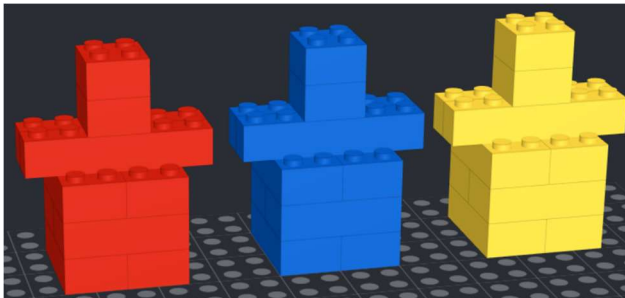
### MATERIAL:

- Euer Roboter (baut einen möglichst kleinen Roboter für diese Aufgabe!)

- Ein Spielfeld im Format A2. Klebt es am besten mit etwas Klebeband am Tisch oder Boden fest, damit es euch nicht verrutscht.



- Drei Lego-Personen:



### STARTAUFSTELLUNG:

Wählt zufällig zwei der drei Personen aus und platziert sie auf den beiden grauen Quadraten. Die dritte Person ist nicht im Spiel.

Euer Roboter startet hinter der Start-/Ziel-Linie ausserhalb des Spielfelds.

### TEILAUFGABE 1:

Bringt die beiden Personen in das farblich passende Feld. Es gibt nur Punkte für Personen, die unbeschädigt sind.

### TEILAUFGABE 2:

Fahrt den Roboter zurück hinter die Start-/Ziel-Linie. Für diese Teilaufgabe gibt es nur Punkte, wenn mindestens für eine Person aus Teilaufgabe 1 Punkte vergeben werden.

### BEWERTUNGSTABELLE:

	Punkte	Anzahl	Gesamt
<b>Teilaufgabe 1:</b> Bringe die beiden Personen in das farblich passende Feld.			
Die Person ist vollständig im richtigen Feld.	20		

Die Person ist teilweise im richtigen Feld und berührt kein falsches Feld.	15		
Die Person ist teilweise oder ganz in einem falschen Feld.	5		
<b>Teilaufgabe 2:</b> Fahre den Roboter zurück hinter die Start-/Ziel-Linie. (Nur wenn in Teilaufgabe 1 Punkte erzielt wurden.)			
Der Roboter hat die Ziellinie vollständig überquert.	20		
Der Roboter hat die Ziellinie teilweise überquert.	10		
	<b>Gesamtpunktzahl</b>		

## ÜBERLEGUNG ZUM LÖSEN DIESER AUFGABE

### Hinweis:

Für diese Aufgabe gibt es nicht eine einzelne richtige Lösung. Viele Wege führen nach Rom, und genauso gibt es auch viele Möglichkeiten, diese Aufgabe zu lösen. Es wird aber ein Weg aufgezeigt, der funktionieren kann. Er muss allerdings noch mit vielen Details «gefüttert» werden, um zu funktionieren. Diese Aufgabe soll für dich eine Anregung sein, selbst zu experimentieren, nicht mehr eine Anleitung, die 1:1 umgesetzt werden kann!

Wir arbeiten hier mit Pseudo-Code: Er zeigt an, welche Schritte wir brauchen, aber es fehlen noch viele wichtige Details.

Wir gehen für die folgenden Überlegungen davon aus, dass unser Roboter einen Farbsensor hat, mit dem er die Figuren scannen kann, sowie einen «Einlademechanismus», mit dem jeweils eine Person transportiert werden kann. Wenn wir den Roboter so konstruieren, dass beide Personen gleichzeitig transportiert werden können, dann ändern sich natürlich auch diese Überlegungen.

Wir schauen uns zunächst Teilaufgabe 1 an. Diese ist recht komplex. Es gibt drei mögliche Objekte, aber nur zwei davon sind im Spiel, und wir wissen nicht, auf welcher der beiden Startpositionen sie stehen. Und sie müssen in das farblich passende Zielfeld gebracht werden.

### STRUKTUR

Wir können nun die Aufgabe in mehrere kleine Abschnitte herunterbrechen:

1. Aus dem Startbereich fahren bis zum Personenfeld 1
2. Person einladen und Farbe erkennen
3. Zu einem der drei Zielbereiche fahren und die Person ausladen
4. Zurückfahren bis zum Personenfeld 2
5. Person einladen und Farbe erkennen
6. Zu einem der drei Zielbereiche fahren und die Person ausladen

**Überlege:** Gibt es hier Dinge, die sich wiederholen, und die man daher in einen MyBlock auslagern kann?

Es fällt sofort auf: Schritt 2 und Schritt 5 sind identisch, und Schritt 3 und Schritt 6 sind ebenfalls identisch.

Es bietet sich also an, diese beiden Teile auszulagern.

Es macht also Sinn, mit dieser Struktur zu starten:



Unter dem ersten Hauptstapel (**Wenn das Programm startet**) kommt all das, was sich verändert, unter die anderen beiden Stapel das, was gleichbleibt.

### SCHRITT 1: AUS DEM STARTBEREICH FAHREN BIS ZUM PERSONENFELD 1

Hier ist Pseudocode für den ersten Schritt aus der Liste oben (Aus dem Startbereich fahren bis zum Personenfeld 1):

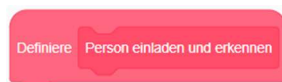


- Vorfahren bis zur ersten Kreuzung
- Nach links drehen

Das ist nur sehr wenig. Vielleicht denkst du, dass du ja jetzt noch nach vorne fahren musst. Aber schau dir das Spielfeld noch mal genau an: Der Abstand von der Kreuzung zum Personenfeld 1 und zum Personenfeld 2 ist genau gleich lang. Und wir haben gelernt: Gleiches wird ausgelagert! Darum kommt das nach vorne fahren bis zur Person in den MyBlock **Person einladen und erkennen**.

### SCHRITTE 2 UND 5: PERSON EINLADEN UND FARBE ERKENNEN

Schauen wir uns nun an, was in diesen MyBlock reingehört:



- Vorfahren bis zur Person (möglicherweise muss der «Einsammel-Mechanismus» schon vorher geöffnet werden, um die Person nicht umzustossen! Der Platz hier ist sehr begrenzt, probiere also aus, wann es am besten ist.)
- Person einsammeln («Einsammel-Mechanismus» schliessen)
- Wert des Farbsensors auslesen und in eine Variable **«Personen-Farbe»** schreiben
- Zurückfahren bis zur Kreuzung.

Jetzt können wir wieder im **Hauptstapel** weitermachen.

- Wir müssen jetzt wieder nach rechts drehen, damit unser Roboter richtig steht. Diese Drehung ist nicht mehr im MyBlock, da wir beim Einsammeln der zweiten Person ja andersrum drehen müssen.

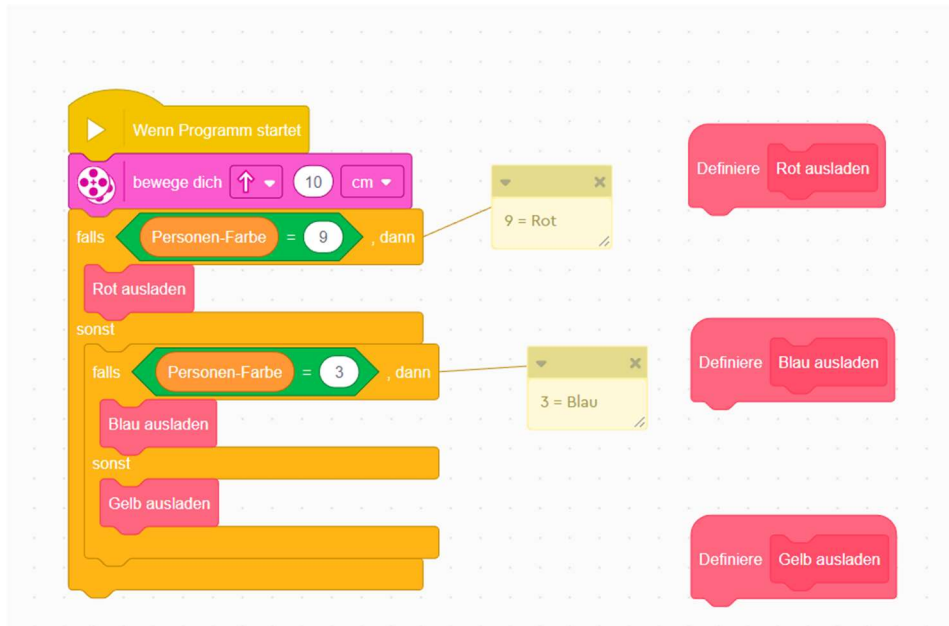
### SCHRITTE 3 UND 6: ZU EINEM DER DREI ZIELBEREICHE FAHREN UND DIE PERSON AUSLADEN

Aber dann können wir schon mit dem anderen MyBlock «**Person zum Zielbereich bringen**» weitermachen.



- Fahre vor bis zur zweiten Kreuzung
- Jetzt wird es Zeit, die Variable **Personen-Farbe** auszulesen, die wir vorhin geschrieben haben.
- Nun müssen wir uns entscheiden: Wenn **Personen-Farbe = rot**
  - Fahre zum roten Zielbereich (drehe rechts, fahre gerade aus)
  - Lade deine Person aus
  - Fahre zurück zur zweiten Kreuzung und drehe dich nach links
- Sonst
  - Wenn **Personen-Farbe = Blau**
    - Fahre zum blauen Zielbereich (drehe links, fahre gerade aus)
    - Lade deine Person aus
    - Fahre zurück zur zweiten Kreuzung und drehe dich nach rechts
  - Sonst
    - Fahre zum gelben Zielbereich
    - Lade deine Person aus
    - Fahre zurück zur zweiten Kreuzung und drehe dich um 180 Grad
- Fahre zurück zur ersten Kreuzung  
Bemerkung: Dies ist eigentlich Schritt 4, da wir jedoch nach jedem Ausladen anders stehen, muss es an den Ausladeort angepasst werden und dementsprechend in den **Blau ausladen**, **Rot ausladen** und **Gelb ausladen** Block hinein. Er soll ja nach dem Ausladen immer gleich stehen, so dass er einen zweiten einladen kann oder dann über die Ziellinie fahren kann, wofür er ja immer gleich stehen muss.

Wir sehen schon: Selbst im stark verkürzten Pseudo-Code wird dieser MyBlock sehr lang. Darum ist es eleganter, wenn wir ihn wieder aufteilen in drei separate MyBlocks: **Blau ausladen**, **Rot ausladen** und **Gelb ausladen**. Im MyBlock «**Personen zum Zielbereich bringen**» ist lediglich die Struktur drin, in den drei «Ausladen»-Blöcken ist der Code von der ersten Kreuzung zum passenden Zielbereich und wieder zurück zur ersten Kreuzung.



### Hinweis:

Die SPIKE App hat für jede Farbe eine Nummer. Die wichtigsten Farben sind

- Weiss = 10
- Schwarz = 0
- Rot = 9
- Gelb = 7
- Blau = 3
- Grün = 5

Du kannst jetzt jeden Teil für sich programmieren, testen, und dann zusammenhängen, wenn es funktioniert.

### TEILAUFGABE 2

Als letztes müssen wir den Roboter noch zurück über die Start-/Ziel-Linie fahren.

Das ist gar nicht mehr schwer. Wir müssen einfach noch die letzten paar Zentimeter fahren, und schon sind wir «zu Hause»!

### ERGEBNIS HAUPTSTAPEL:

Wenn wir alles richtig gemacht haben, sieht unser Hauptstapel so aus:





Das sind 16 Textblöcke, inklusive den 4 Blöcken im Kopfteil und dem Block «Stoppe und verlasse das Programm» ganz am Schluss. Ausserdem haben wir 5 ziemlich kurze MyBlock-Stapel. Wir müssen zwar ein bisschen auf dem Bildschirm hin- und herschieben, damit alles passt, aber es ist übersichtlich.

Wenn wir dasselbe ohne MyBlocks machen wollten, wäre unser Code viel länger. Über 60 Zeilen Code in einem einzigen Stapel wären notwendig! Nicht ganz zu Unrecht hat ein solcher Code den Spitznamen «WC-Rollen-Code»...