Wir programmieren einen Roboter
Eine Unterrichtsreihe für die 4. Klasse

Manuel Lammers
(manuellammers@icloud.com)

Material

Schwerpunkte nach MKR NRW:
1.1, 1.2, 6.1, 6.2, 6.3, 6.4

Benötigte Materialien:

- Kopien dieser Arbeitsblätter für jeden Schüler
- Weiße Blätter
- Lernroboter: ozobot Bit oder Evo

Das Material wurde mit der Software „Worksheet Crafter“ erstellt.
Die Abbildungen der „Fahrbahnen“ wurden von Manuel Lammers für dieses Material erstellt.
Bei den Abbildungen von Programmierblöcken und Schaltflächen handelt es sich um Screenshots aus der Programmieroberfläche ozoblockly.com.
Alle anderen Abbildungen entstammen dem Worksheet-Crafter oder den mit ihr verknüpften freien Bildersammlungen „pixabay“ oder „OpenClipart“.

Name:	Willkommen	1

Willkommen zum zweiten Teil!

Huch, du bist ja immer noch da! Dann kann der erste Teil ja nicht sooo schlimm gewesen sein!
Aber jetzt wird´s richtig fies und ENGLISCH! - „Oh nein, Englisch! Das kann ich nicht! Ich habe keine Lust! MAMA!“
Ok, Spaß beiseite, so schlimm wird es auch jetzt nicht. Du wirst aber einige neue Wörter benötigen. Die englischen Anweisungen für Bit kommen dann ganz langsam hinzu und ich erkläre dir alles genau!
Du hast jetzt schon einiges über Ozobot Bit oder seinen großen Bruder Evo gelernt. Du weißt, wie man ihn mit Hilfe von Farbcodes Anweisungen erteilen kann.
Im ersten Teil des Kurses habe ich dir gesagt, dass Ozobot - genau wie die meisten Roboter - nicht sprechen kann. Dabei bleibe ich auch. Was Bit und Evo aber können: Sie können Sprache verstehen!
Leider verstehen sie aber kein Deutsch oder Englisch. Menschliche Sprachen wären für Computer viel zu kompliziert. Aber sie verstehen eine Computersprache, die aus Nullen und Einsen besteht. Leider wäre die aber für uns Menschen viel zu kompliziert. Und genau darum gibt es Programmiersprachen.
In einer Programmiersprache kann man für Menschen verständliche Anweisungen eingeben - meistens auf Englisch. Die Programmiersprache übersetzt unser Programm dann in eine für den Computer verständliche Computersprache aus Nullen und Einsen. Dieses fertig übersetzte Programm geben wir dann Bit oder Evo.

In diesem Kurs gehe ich nicht auf die besonderen Fähigkeiten von Evo ein. Darum spreche ich hier immer nur von „Bit“. Alle Übungen und Experimente kannst du aber auch mit Evo erledigen.

In diesem Kurs wird es also darum gehen, Bit nicht mehr mit Hilfe von Farbcodes Anweisungen zu geben. Stattdessen programmieren wir ihn mit „Ozoblockly“.

Name:	Ozoblockly kennenlernen	2

Woher bekomme ich dieses „Ozoblockly“?

Ozoblockly funktioniert über eine Internetseite. Du kannst Ozoblockly also auf jedem PC, Mac, Tablet oder jedem anderen aktuellen Gerät benutzen, mit dem du Internetseiten aufrufen kannst. Dazu gehst du einfach auf die Webseite:

ozoblockly.com

Du bist genauso schreibfaul wie ich? Dann scanne halt diesen QR-Code, er bringt dich auch auf die richtige Webseite:

Klicke auf die Schaltfläche „**Get Startet**“.
Wenn du das erste Mal hier bist, erscheint das Fenster „Welcome to OzoBlockly“, das dich freundlich begrüßt. Entferne den Haken bei „**Show at startup**“, dann wird dieses Fenster zukünftig nicht mehr erscheinen. Jetzt noch schnell auf das Kreuzchen oben rechts geklickt, dann ist das Fenster verschwunden.

Da sind wir schon. Auf der nächsten Seite zeige ich dir, wie der Bildschirm ungefähr aussehen sollte. Je nachdem welchen Computer du gerade benutzt, kann das Bild etwas anders aussehen. Im Prinzip bleibt aber alles gleich:

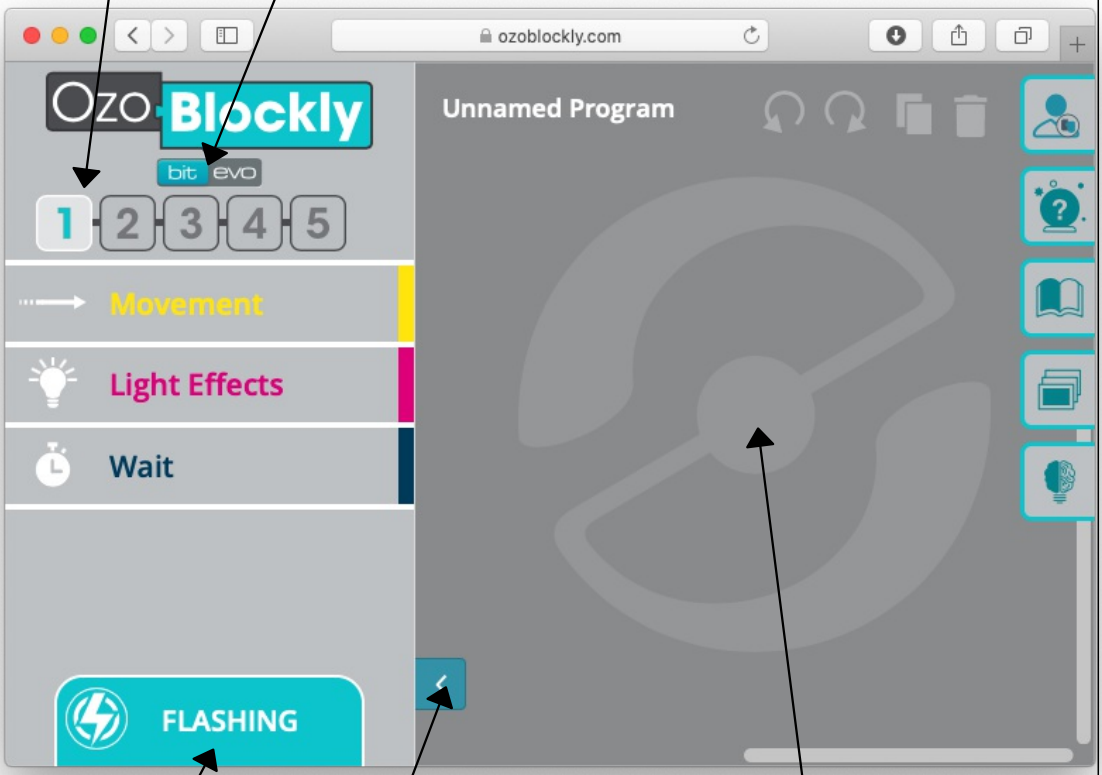
Name:	Ozoblockly kennenlernen	3

Ozoblocklys Startbildschirm

Hier kannst du auswählen, wie gut du dich schon mit Ozoblockly auskennst. Je höher die Zahl ist, desto mehr Anweisungen hast du zur Verfügung, das Programmieren wird aber auch schwieriger. Auf Stufe 1 benötigst du nicht einmal englische Wörter!

2

3 Hier wählst du aus, ob du mit „Bit“ oder mit „Evo“ arbeitest!



1

Anweisungen:

Hier stehen dir Anweisungen zum Programmieren zur Verfügung. Sie sind in zusammengehörige Bereiche sortiert:

- „Movement“ -> Bewegung
- „Light Effects“ -> Lichteffekte
- „Wait“ -> Warteanweisungen

6

Falls dein Bildschirm sehr klein ist und du Platz benötigst, kannst du hier die linke Spalte verkleinern oder vergrößern.

7

Arbeitsbereich: Hier entsteht dein Programm.

5

Programm übertragen:

Hier wird dein fertiges Programm an Bit oder Evo geschickt.

Name:	Das erste Programm	4

Das erste Programm

Oh je, schon drei Seiten gelesen und noch nichts programmiert. Dabei gibt es noch ein bisschen was zu erklären. Das lassen wir aber jetzt erst einmal weg und programmieren!

Falls im Arbeitsbereich (der große graue Bereich) schon etwas steht, dann müssen wir das entfernen. Wir benötigen einen leeren Arbeitsbereich:
Klicke oben rechts auf den Mülleimer!

Es ist Tradition, dass das erste Programm, das man in einer neuen Programmiersprache schreibt, die Wörter „Hello World!“ auf dem Bildschirm anzeigen soll. Leider hat Bit aber weder einen Bildschirm noch einen Lautsprecher. Darum lassen wir ihn in unserem ersten Programm einfach ein bisschen fahren:

Übung 1:

- Wähle Schwierigkeitslevel 1 aus!

- Wähle bei den Anweisungen den gelben Bereich „Movement“ (Bewegung) aus!

Das musst du immer machen, wenn du eine Anweisung aus dem Bereich „Bewegung“ hinzufügen möchtest.

- Du siehst jetzt sehr viele gelbe Anweisungen. Sie alle haben etwas mit Bewegungen zu tun.
- Wähle diese Anweisung aus:

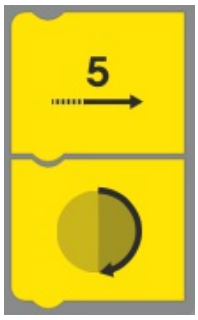
Diese Anweisung bedeutet: „Gehe 5 Schritte vorwärts!“

Name:	Das erste Programm	5

Herzlichen Glückwunsch! Dein erstes Programm ist fertig! - Es ist allerdings noch etwas langweilig. Bevor wir es testen, fügen wir noch zwei Anweisungen hinzu:
Füge den Befehl „umdrehen“ hinzu. Bit wird sich dann um einen halben Kreis drehen.

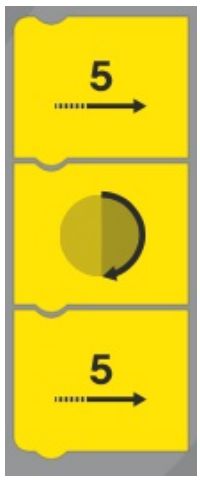
Auf deinem Arbeitsbereich sind jetzt beide Anweisungen zu sehen. Aber jetzt wird Bit nicht wissen, welche Anweisung er zuerst ausführen soll. Darum kannst du die Anweisungen im Arbeitsbereich verschieben und hintereinander sortieren:

Sortiere die Anweisungen so, dass Bit erst fünf Schritte läuft.
Danach soll er sich umdrehen.
Das Ganze sollte dann so aussehen:



- Alles verstanden?
  Super! Dann lassen wir Bit am Ende unseres ersten Programms wieder zurücklaufen. Nicht, dass wir ihm noch hinterherrennen müssen!
  Dein fertiges Programm müsste jetzt so aussehen:

- 5 Schritte vor
- umdrehen
- 5 Schritte vor



- Auf der nächsten Seite zeige ich dir, wie du Bit das fertige Programm geben und dann testen kannst.

Name:	Das erste Programm	6

Bit das Programm geben

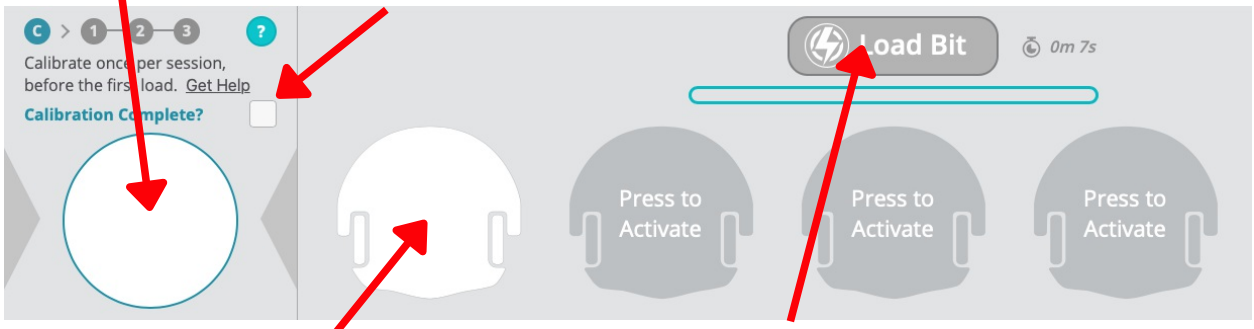
Dein erstes Programm ist jetzt fertig, nun muss Bit es noch irgendwie bekommen. Man sagt dazu „Das Programm wird zu Bit übertragen." Bit hat kein Bluetooth und der USB-Anschluss ist nur zum Aufladen des Akkus da. Wie kann man Bit also ein Programm übertragen? - Über seine Sensoren an der Unterseite! Ozoblockly kann den Bildschirm so blinken lassen, dass Bit dein Programm über diese blinkenden Farbcodes sozusagen „sehen" kann.
Und das geht so:

- Unten befindet sich ein Schaltfläche mit der Aufschrift „FLASHING" (blitzen). Drücke sie!

- Unten klappt ein neues Fenster auf:

1 Hier musst du Bit „kalibrieren", Bit lernt so die Farben deines Computers genau kennen und funktioniert dann besser. Diesen Schritt musst du nur beim ersten Mal machen, wenn du deinen Computer zum Programmieren benutzt. Wenn du den Computer oder das Tablet wechselst, musst du Bit neu kalibrieren.
Zum Kalibrieren:
  - schalte Bit an
  - halte die Taste 2-3 Sekunden lang gedrückt. Wenn Bit einmal weiß blinkt, dann stelle ihn auf das weiße Feld. Er beginnt zu blinken. Wenn er am Ende grün blinkt, hat alles geklappt.

2 Hake das Feld ab, um zu bestätigen, dass du Bit kalibriert hast. Schalte Bit aus.



3 Schalte Bit wieder ein! Stelle ihn nun auf das weiße Programmierfeld.

4 Drücke die Schaltfäche „Load Bit". Dein Programm wird nun zu Bit übertragen. Wenn der Balken unter der Schaltfläche hinten angekommen ist, dann ist dein Programm fertig übertragen.

Du startest dein Programm, indem du Bits Knopf zweimal schnell hintereinander drückst. (Bit muss dafür ausgeschaltet sein!)

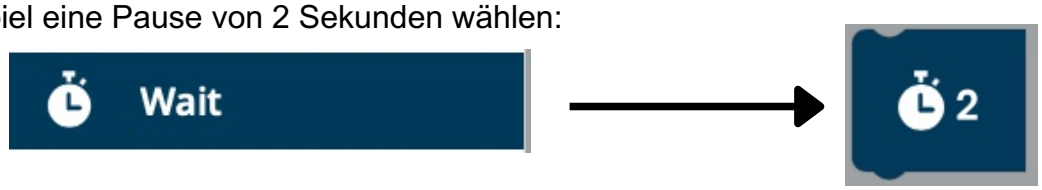
Name:	Das erste Programm	7

Das Programm verbessern

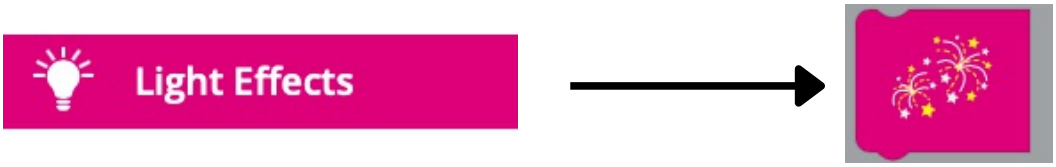
Ist dir aufgefallen, dass Bit sofort gestartet ist, nachdem du zweimal den Taster gedrückt hast? Das ist unpraktisch. Vermutlich hattest du Bit noch in der Hand, als er schon losfahren wollte. Aber genau die Anweisung hast du ihm schließlich gegeben. Das sollten wir mit einer zusätzlichen Anweisung verbessern:
Im Bereich „wait“ gibt es die Anweisungen, die Bit warten lassen.

Übung 2:

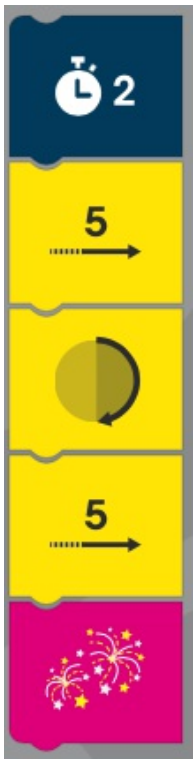
Füge deinem Programm vor dem Start eine kleine Pause hinzu. Ich werde für mein Beispiel eine Pause von 2 Sekunden wählen:



Und wenn Bit am Ende eine kleine Lightshow für uns hinlegen würde, dann wäre das doch auch ganz nett, oder? Ich wähle am Ende meines Programms aus der Gruppe „Light Effects“ (Lichteffekte) das Feuerwerk aus:



Führe am Ende deines Programms auch einen Lichteffekt hinzu.



Mein fertiges Programm sieht jetzt so aus:

- 2 Sek. warten
- 5 Schritte vorwärts gehen
- umdrehen
- 5 Schritte vorwärts gehen
- Feuerwerk zeigen

Wenn dein Programm fertig ist, musst du es wieder neu zu Bit übertragen. So, wie wir es im letzten Beispiel auch gemacht haben.

- Bit kalibrieren, falls dies dein erstes Programm heute ist oder du einen anderen Computer benutzt.
- Das Feld abhaken, um zu bestätigen, dass Bit kalibriert ist.
- Bit auf das Programmierfeld stellen und „Load Bit“ drücken.

Um dein Programm zu testen, drücke wieder 2 mal schnell hintereinander Bits Taste. (Bit muss dafür ausgeschaltet sein!)

Name:	Jede Menge Anweisungen	8

Du weißt jetzt, wie man Bit programmieren kann, wie man Bit die Programme übergibt (überträgt) und wie man sie startet.
In den nächsten Beispielen werde ich nur noch die eigentlichen Programmaufgaben mit dir besprechen. Natürlich sollst du sie immer zu Bit übertragen und testen, auch wenn ich dies nicht mehr jedes Mal dazuschreibe.

Jede Menge Anweisungen

Übung 3:

Auf der ersten Stufe sind die Anweisungen noch sehr einfach zu verstehen. Teste doch einfach mal alle anderen Anweisungen und schaue genau, was sie bewirken. Die Symbole für die Anweisungen sind so einfach, dass du bestimmt bei einigen schon vor dem Testen erraten kannst, wie Bit auf sie reagieren wird.
Wenn du jede Anweisung in einem eigenen Programm testest, dann wird das sehr lange dauern. Erstelle doch einfach einige Programme mit mehreren Anweisungen.
Tipp: Wenn du zwischen den Anweisungen immer eine Pause einfügst, kannst du sie beim Testen besser voneinander unterscheiden.

Was bewirken diese Anweisungen:

Name:	Jede Menge Anweisungen	9

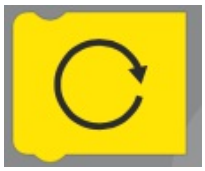
Lösungen:

schnell

langsam

10 Schritte vorwärts fahren

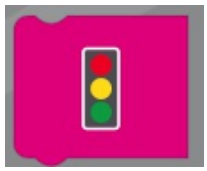
10 Schritte rückwärts fahren



einen großen Kreis fahren

einen kleinen Kreis fahren

Bit leuchtet hintereinander in allen Regenbogenfarben



Bit leuchtet hintereinander in den Ampelfarben

Bit macht 5 Sek Pause.

Übung 4:

Male eine Kreuzung auf ein Blatt Papier.
Programmiere Bit so, dass er an der Kreuzung links abbiegt!

Für Profis:
Lass Bit an einer aufgemalten Kreuzung so links abbiegen, als würdest du mit dem Fahrrad links abbiegen: links einordnen, bis zur Kreuzung vorfahren, kurz anhalten, links abbiegen.
Sich umschauen und das Handzeichen geben kann Bit leider nicht. Darauf müssen wir verzichten. Oder du denkst dir Leuchtsignale aus, die das Handzeichen ersetzen.

Tipp: Bit fährt leider nicht sehr genau. Je länger die Strecke ist, desto mehr weicht er von deiner programmierten Strecke ab. Male die Kreuzung also nicht unnötig groß!

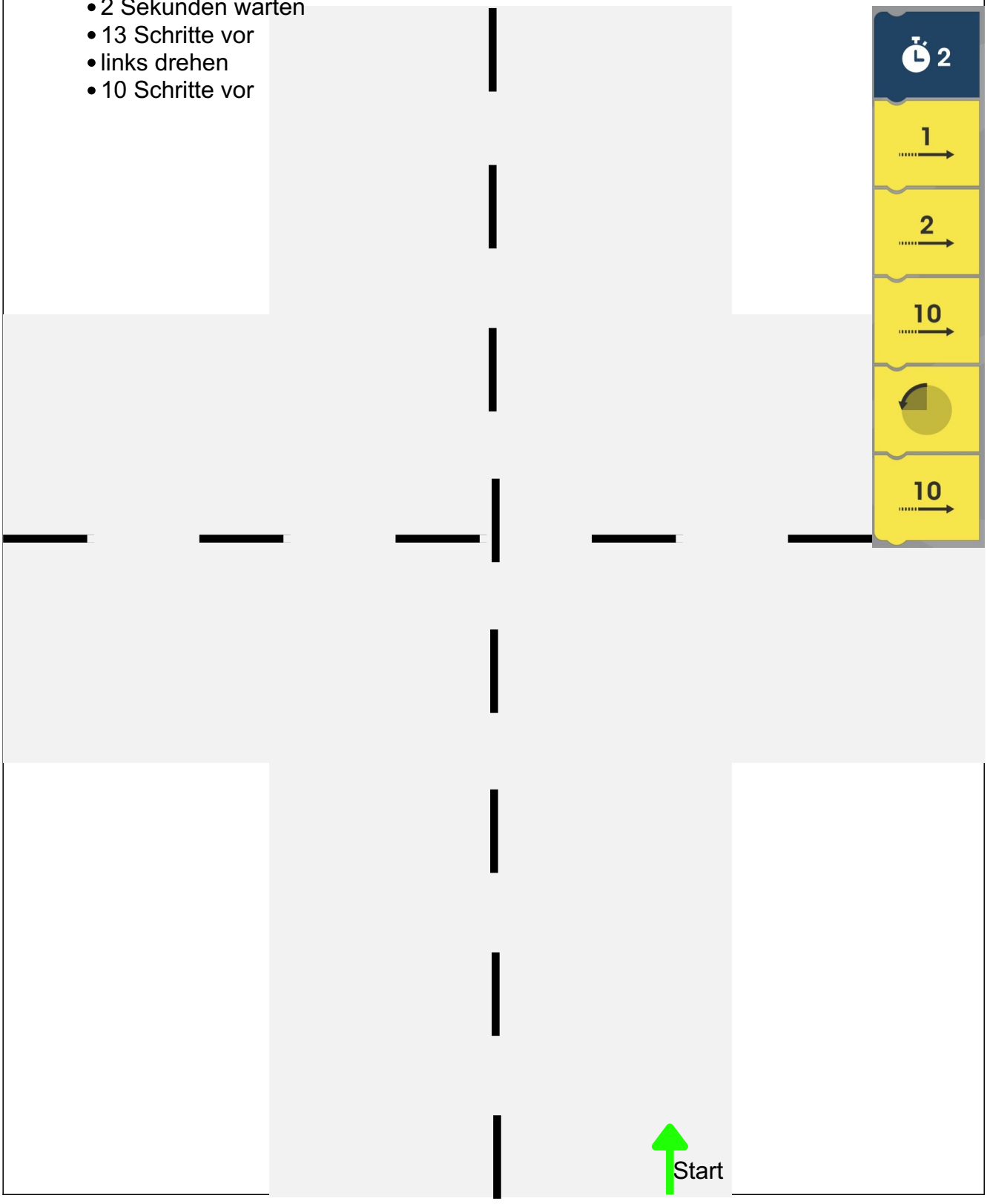
Falls Bit sehr große Probleme hat, eine lange Strecke geradeaus zu fahren, dann müssen die Motoren neu kalibriert werden. Lasse dir von deinen Eltern oder deinem Lehrer helfen!

Name:	Übungen Stufe 1	10

Lösung (einfach):

Deine Kreuzung kann auch ganz anders aussehen. Hier zeige ich dir meine Kreuzung und das passende Programm dafür:

- 2 Sekunden warten
- 13 Schritte vor
- links drehen
- 10 Schritte vor

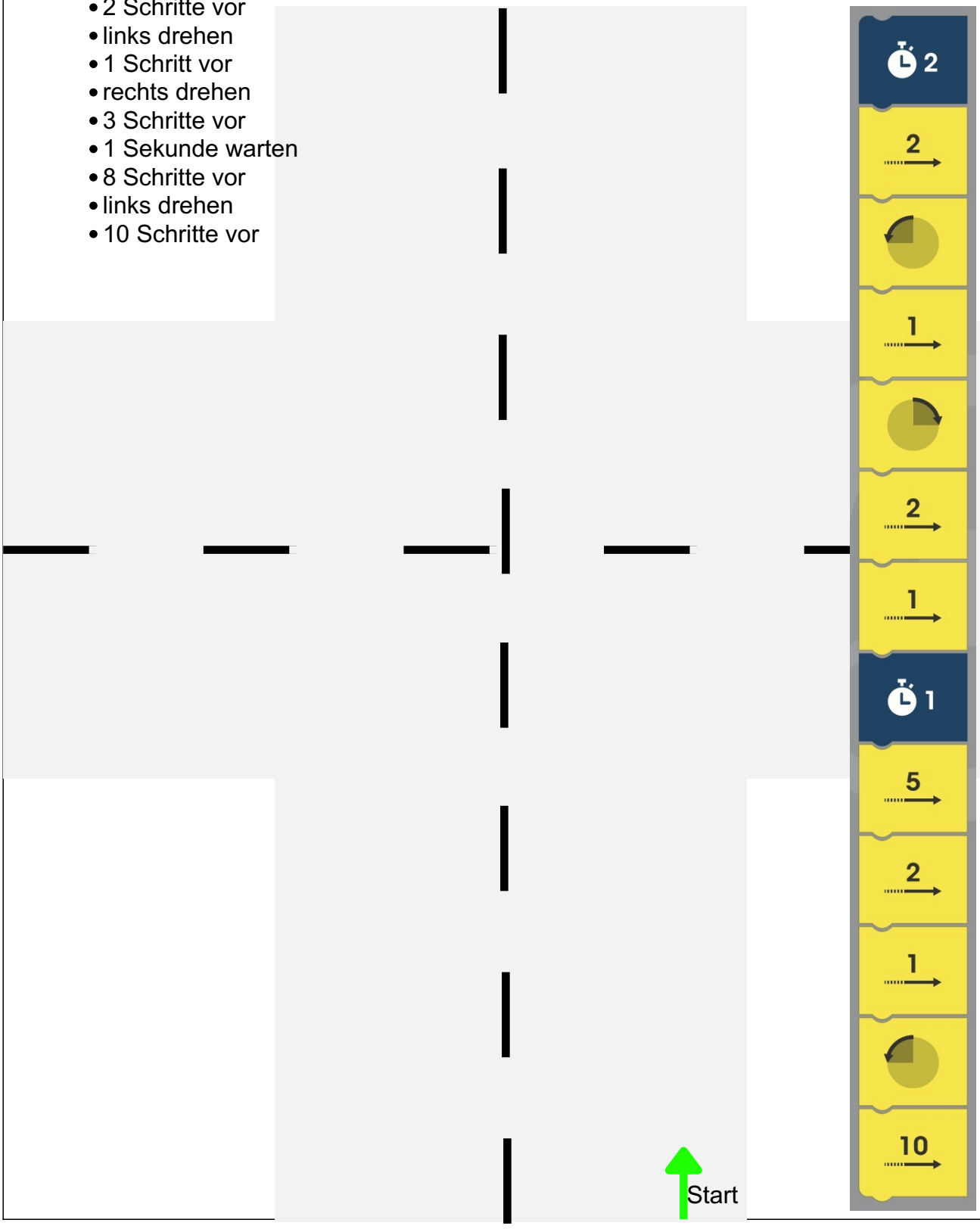


Name:	Übungen Stufe 1	11

Lösung (Profi):

Deine Kreuzung kann auch ganz anders aussehen. Hier zeige ich dir meine Kreuzung und das passende Programm dafür:

- 2 Sekunden warten
- 2 Schritte vor
- links drehen
- 1 Schritt vor
- rechts drehen
- 3 Schritte vor
- 1 Sekunde warten
- 8 Schritte vor
- links drehen
- 10 Schritte vor



Übung 5:

Nimm dir ein Blatt Papier!
Male einen kleinen Startpunkt und einen größeren Zielbereich auf!
Lege einige Hindernisse (Papierkügelchen, Würfel, Stifte...) zwischen Start und Ziel.
Programmiere Bit jetzt so, dass er das Ziel erreicht, ohne ein Hindernis zu berühren.

Tipp: Lasse genug Abstand zwischen den Hindernissen, weil Bit nicht so genau fahren kann!

Wenn du die Übung geschafft hast, gib deinen Hindernisparcours einem anderen Kind. Jetzt muss das andere Kind versuchen, ein Programm zu schreiben, mit dem Bit durch deinen Parcours zum Ziel kommt.

Für Profis:
Bastle dir anstatt Hindernissen kleine Tore aus Papier, durch die Bit zum Ziel fahren muss.

Tipp: Die Tore sollten 5-8 cm breit sein, weil Bit nicht so genau fahren kann!

Übung 6:

Erstelle ein Programm für Bit mit 5 Anweisungen. Ein Mitschüler schaut sich an, was Bit macht und muss versuchen, dein Programm nachzubauen. Für diese Übung benötigt ihr 2 Bit, damit ihr das Verhalten eurer Roboter vergleichen könnt!

Tipp: Es wird einfacher, wenn ihr nach jeder Anweisung eine kurze Pause einfügt.

Wiederholt die Übung so oft, dass jeder 2 mal raten und nachbauen durfte!

Übung 7:

Teste alle Anweisungen, die du noch nicht ausprobiert hast. Ich bin mir sicher, dass du schnell herausfinden wirst, was sie bewirken.

Name:	Stufe 2	13

Do you speak english?

Auf der ersten Seite hatte ich schon angedeutet, dass es im zweiten Teil der Programmieranleitung für Ozobot Bit „englisch“ wird.
Echte Programmiersprachen benutzen englische Anweisungen. Falls du später mal richtig professionell programmieren möchtest, wirst du schnell feststellen, dass auch fast alle guten Fachbücher englischsprachig sind. Das kann man gut oder doof finden - es ist einfach so.
Ab jetzt werden die Anweisungen mit den Symbolen gegen echte englische Anweisungen ersetzt.
Das sieht dann zum Beispiel so aus:

Aus 

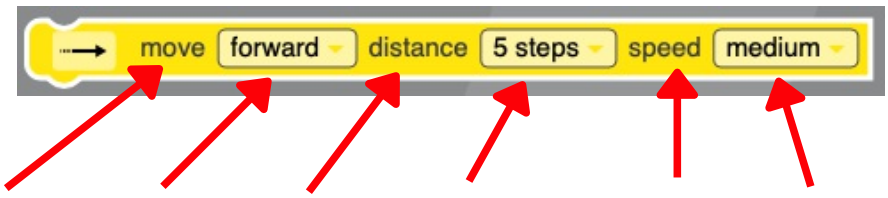
in Stufe 1

wird 

in Stufe 2

move = bewege dich
forward = vorwärts
distance = Entfernung
5 steps = 5 Schritte
speed = Geschwindigkeit
medium = mittel

Übersetzt bedeutet das also:



“bewege dich: vorwärts, Entfernung: 5 Schritte, Geschwindigkeit mittel

Wer also absolut keine Lust auf Englisch hat, für den ist an dieser Stelle Schluss. Ohne Englischkenntnisse wirst du niemals über die Grundlagen der Programmierung hinauskommen!
Aber es wird nicht nur schwieriger, du bekommst auch etwas für deine Mühen: Mit einer einzigen Anweisung kannst du nun viel mehr machen: Hier wird Bit nicht nur gesagt, dass er sich bewegen soll. Es wird ihm auch gesagt: wohin, wie weit und wie schnell er dies tun soll.
Diese zusätzlichen Angaben nennt der Fachmann **„Parameter“**. „Move“ ist also die **Anweisung**, der Rest sind die **Parameter** zu dieser **Anweisung**.

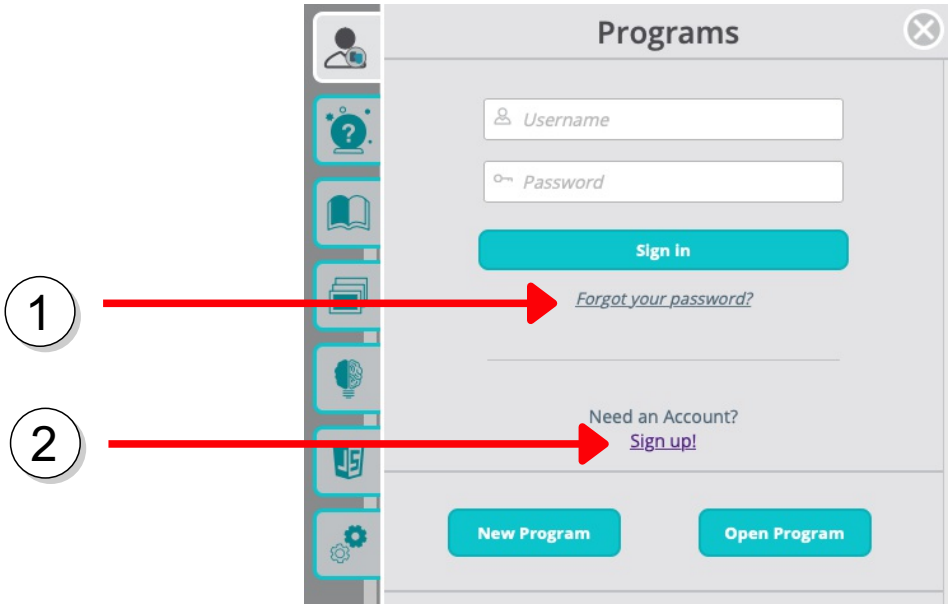
Name:	Stufe 2	14

Englisch schreckt dich nicht ab? Sehr gut! Dann kann es ja jetzt weitergehen!

Als Erstes musst du in Ozoblockly die 2. Stufe aktivieren. Das machst du oben links bei den Zahlen von 1 bis 5:

Ab jetzt werden die Programme länger, oder zumindest dauert es länger, bis man sie erstellt hat. Da lohnt es sich, wenn man seine Programme abspeichert. So kann man sie bei Bedarf jederzeit wieder laden.
Damit du deine Programme speichern und laden kannst, benötigst du einen Benutzernamen und ein Kennwort für Ozoblockly. Dies nennt man einen **„Account“**.
Damit du dir einen A**ccount** erstellen kannst, benötigst du eine **Emailadresse**. Über diese kann man sich auch ein neues Kennwort zuschicken lassen, falls man es einmal vergisst.

Bitte deine Eltern oder deine Lehrer dir dabei zu helfen, einen A**ccount** einzurichten. Das geht an der Schaltfläche oben rechts:



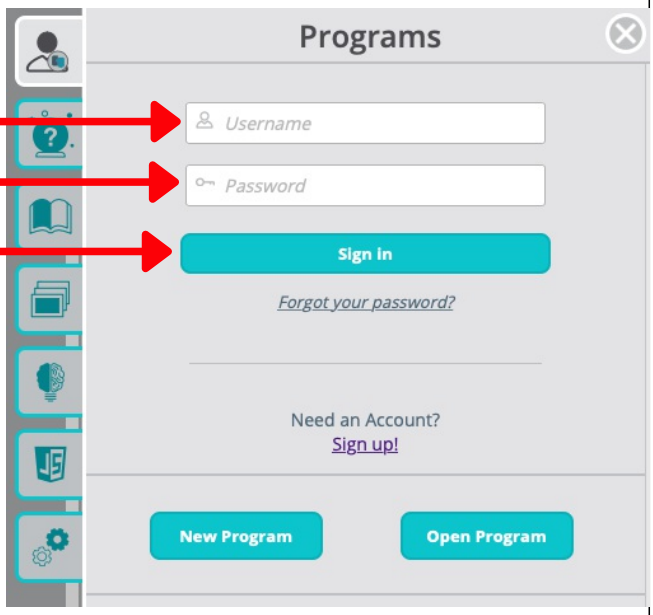
Du kannst hier auch ohne einen **Account** weiterarbeiten. Dann kannst du deine Programme aber nicht abspeichern und wieder laden!

Name:	Laden und Speichern	15

Sobald du einen Account hast, kannst du dich anmelden (“Sign in“), bevor du mit deiner Arbeit beginnst. Und du musst dich wieder abmelden (logout), wenn du mit deiner Arbeit fertig bist. Wenn du dich nach deiner Arbeit nicht ausloggst und auch andere Personen den Computer benutzen, dann können sie auf deine Daten zugreifen!

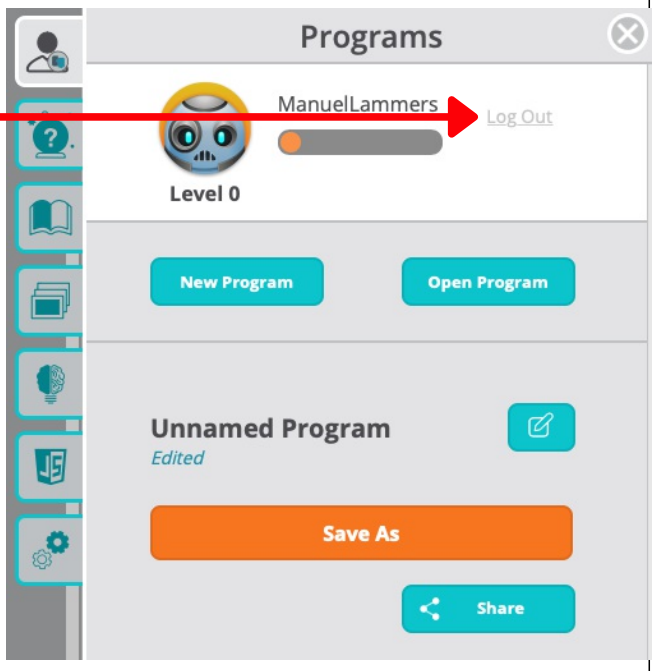
anmelden:

1 dein Benutzername
2 dein Kennwort
3 anmelden



abmelden:

1 abmelden



Wenn du angemeldet bist, kannst du mit der Schaltfläche „Save AS“ oder „Save“ dein Programm speichern.
Mit „Open Program“ kannst du ein gespeichertes Programm aussuchen und laden.
„New Program“ startet ein neues leeres Blatt für ein neues Programm.

Hiermit kannst du dein Programm umbenennen.

Name:	Neue Anweisungen	16

Wenn du dir nun in Ozoblockly die verfügbaren Anweisungen anschaust, dann wirst du feststellen, dass es weniger sind als vorher auf der ersten Stufe. Dafür kannst du aber vielen Anweisungen zusätzliche **„Parameter“** mitgeben. **„Parameter“** sind zusätzliche Einstellungen zu einer **Anweisung**.

Es gibt zum Beispiel nicht mehr mehrere unterschiedliche Anweisungen, um die Farbe der Leuchtdiode einzustellen. Es gibt nur eine: „set top light color“.
Dafür muss man aber hinter der Anweisung die Wunschfarbe einstellen. Die Farbe ist also der „Parameter“ zur Anweisung „set top light color“.

Der Profi sagt: „Der **Anweisung** 'set top light color' wird der **Parameter** 'Farbe' übergeben.“

Andere Anweisungen benötigen keine Parameter: Die Anweisung „firework“ startet noch immer ein voreingestelltes Feuerwerk. Die Farben und die Länge des Feuerwerks kannst du nicht einstellen.

Übung 8:

Erstelle ein Programm, das Bit die Farben

- Rot,
- Gelb,
- Grün und dann
- Blau

anzeigen lässt.

Starte dann dein Programm. Was passiert?

Name:	Mach mal eine Pause!	17

Hat es geklappt?

Vermutlich sieht dein Programm so aus:

Ich vermute, dein Programm funktioniert nicht so, wie du es dir ausgedacht hast und das Licht bleibt einfach aus? Wie kommt das?
Die Antwort ist ganz einfach:
Dein Programm funktioniert, du kannst es nur nicht sehen:
Bit arbeitet so schnell, dass du keine Chance hast, die Farben zu sehen Dann ist Bit mit seiner Aufgabe fertig und schaltet sich aus. Für dich sieht das so aus, als wenn Bit nichts getan hätte.

Übung 9:

Versuche es doch mal so:
Füge nach jeder Anweisung zum Farbwechsel eine Pause ein!
Mit der Anweisung „wait“ (warte) kannst du Pausen einfügen. Der Parameter gibt an, wie lange Bit warten soll.

Jetzt solltest du die Farben gut erkennen können!

Name:	Neue Anweisungen (Movement)	18

Die Anweisungen aus dem Bereich „Light Effect“ (Lichteffekte) entsprechen weitgehend den Anweisungen aus der ersten Stufe. Neu ist nur, dass es jetzt nicht mehr mehrere Anweisungen zum Einstellen des Lichtes gibt. Es gibt nur noch eine. Der Parameter bestimmt nun, welche Farbe Bit einstellen soll.
Im Bereich „Movement“ (Bewegung) hat sich aber einiges getan. Hier lohnt sich ein genauer Blick:

Übung 10:

Finde heraus, was die Anweisungen und Parameter im Bereich „Movement“ bedeuten!

1

2

3

4

Name:	neue Anweisungen (Movement)	19

1

1

1

3

2

Name:	neue Anweisungen (Movement)	20

1

4

3

1

2

Name:	neue Anweisungen (Movement)	21

Meine Lösungen

1 bewegen

Richtung:
2
- vorwärts
- rückwärts

3 Entfernung:
1 Schritt - 10 Schritte weit

Geschwindigkeit:
4
- langsam
- mittel
- schnell
- sehr schnell

drehen:
1
- slight left -> leicht nach links
- left -> nach links
- slight right -> leicht nach rechts
- right - nach rechts
- u-turn left -> umdrehen linksherum
- u-turn right -> umdrehen rechtsherum

In einer Zick-zack-Linie fahren:
1
- langsam
- mittel
- schnell
- sehr schnell

Einen Slalom fahren:

Geschwindigkeit:
1
- langsam
- mittel
- schnell
- sehr schnell

Richtung:
2
- vorwärts
- rückwärts

Herumschleudern:Richtung
1
- links herum
- rechts herum

Name:	Übungen	22

Eine Kreislinie fahren:
small -> klein
big -> groß

Geschwindigkeit:
1
- langsam
- mittel
- schnell
- sehr schnell

Richtung:
2
- vorwärts
- rückwärts

Drehrichtung:
3
- links herum
- rechts herum

4 Dauer:
1 - 5 Sekunden

Die nächsten beiden Übungen kennst du schon. Jetzt wiederholen wir sie auf Stufe 2!

Übung 11:

Nimm dir ein Blatt Papier!
Male einen kleinen Startpunkt und einen größeren Zielbereich auf!
Lege einige Hindernisse (Papierkügelchen, Würfel, Stifte...) zwischen Start und Ziel.
Programmiere Bit jetzt so, dass er das Ziel erreicht, ohne ein Hindernis zu berühren.

Tipp: Lasse genug Abstand zwischen den Hindernisse, weil Bit nicht so genau fahren kann!

Wenn du die Übung geschafft hast, gib deinen Hindernisparcours einem anderen Kind. Jetzt muss das andere Kind versuchen, ein Programm zu schreiben, mit dem Bit durch deinen Parcours zum Ziel kommt.

Für Profis:
Bastle dir anstatt Hindernisse kleine Tore aus Papier, durch die Bit zum Ziel fahren muss.

Tipp: Die Tore sollten 5-8 cm breit sein, weil Bit nicht so genau fahren kann!

Name:	Übungen	23

Übung 12:

Erstelle ein Programm für Bit mit 5 Anweisungen. Ein Mitschüler schaut sich an, was Bit macht. Er muss versuchen, dein Programm nachzubauen. Für diese Übung benötigt ihr 2 Bit, damit ihr das Verhalten eurer Roboter vergleichen könnt!

Tipp: Es wird einfacher, wenn ihr nach jeder Anweisung eine kurze Pause einfügt.

Wiederholt die Übung so oft, dass jeder 2 mal raten und nachbauen darf!

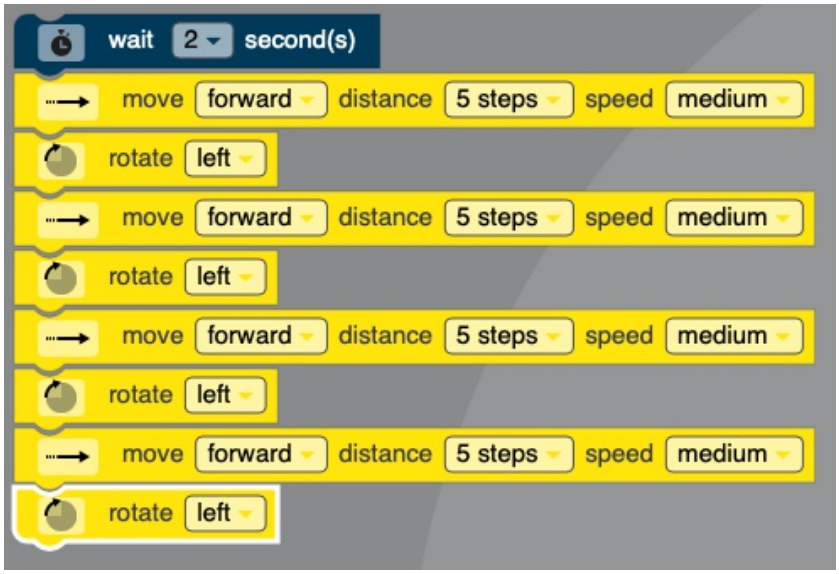
Anders als auf Stufe 1 ist es jetzt viel schwieriger, das Programm eurer Mitschüler genau nachzubauen, weil es sehr viele Parameter gibt. Man kann auch nicht gut sehen, ob Bit beispielsweise 9 oder 10 Schritte geht.

Seid also gnädig! Es reicht aus, wenn das Programm ungefähr nachgebaut wurde!

Name:	Schleifen	24

Schleifen (Loops)

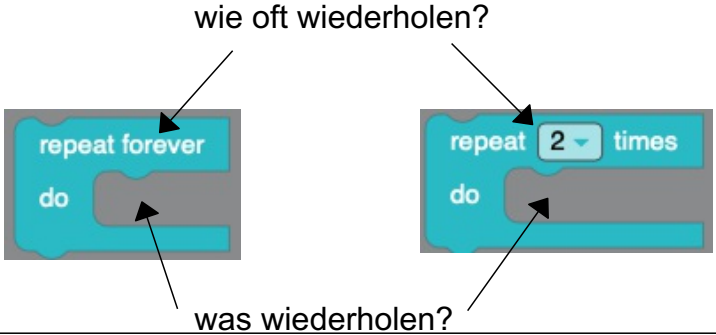
Manchmal kommt es vor, dass Anweisungen oft wiederholt werden. Wenn Bit in Form eines Quadrates fahren soll, dann müssten wird beispielsweise so ein Programm schreiben:



Erklärung:
Bit fährt 5 Schritte und biegt dann nach links ab. Diese beiden Befehle wiederholen sich dann insgesamt 4 mal. Wenn Bit fertig ist, ist er in Form eines Quadrates gefahren.
Das Ganze ist aber eine ganz schöne Klickerei, bis man alle Anweisungen untereinander gezogen und die Parameter eingestellt hat.

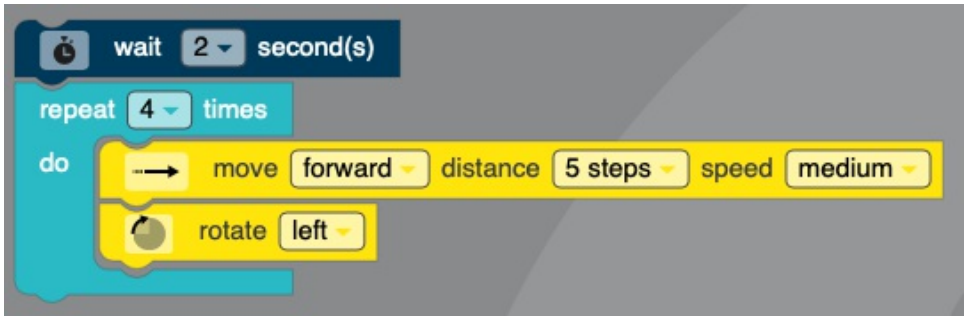
Das bekommen wir auch schöner und mit weniger Arbeit hin. Und zwar mit einer Schleife - oder auf Englisch „loop“.

Bei den Anweisungen findest du in türkiser Schrift auch den Bereich „Loops“. Es gibt dort die beiden Anweisungen „repeat forever“ und „repeat 2 times“.
„Forever“ bedeutet „für immer“. Mit dieser Anweisung kannst du Bit andere Anweisungen immer wieder wiederholen lassen. Bei der zweiten Anweisung ist die „2“ der Parameter. Hier kannst du auch andere Zahlen eingeben und so bestimmen, wie oft andere Anweisungen wiederholt werden sollen.
Die Anweisungen, die du wiederholen lassen möchtest, müssen hinter „do“ (“tue“ oder „mache“) eingesetzt werden.
Klingt kompliziert? Ist es aber gar nicht! Auf der nächsten Seite zeige ich dir ein Beispiel.



Name:	Schleifen	25

Und noch einmal das Ganze!
Jetzt zeige ich dir wieder, wie man Bit ein Quadrat fahren lassen kann. Aber nun ist das Programm viel kürzer, weil ich nicht alle Anweisungen vier mal hintereinander einbauen muss. Stattdessen habe ich die Anweisungen „move“ und „rotate“ aber nur jeweils einmal benutzt. Sie werden aber vier mal wiederholt.



Und wieder fährt Bit in Form eines Quadrates.

Übung 13:

Lass Bit ein Quadrat mit der Seitenlänge „zehn Schritte“ fahren.

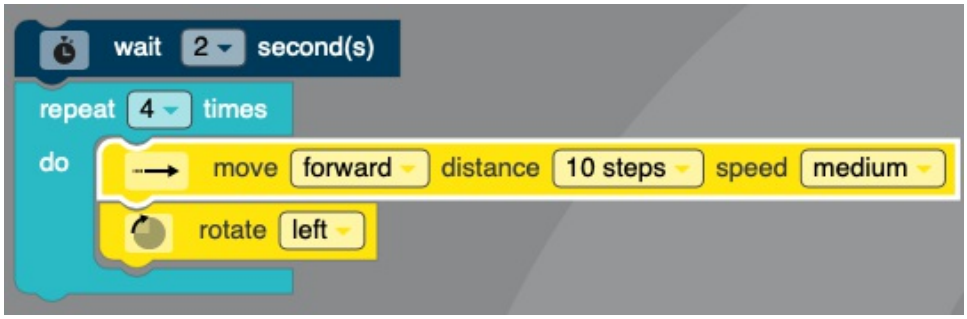
Übung 14:

Lass Bit ein Achteck mit der Seitenlänge „fünf“ Schritte fahren.

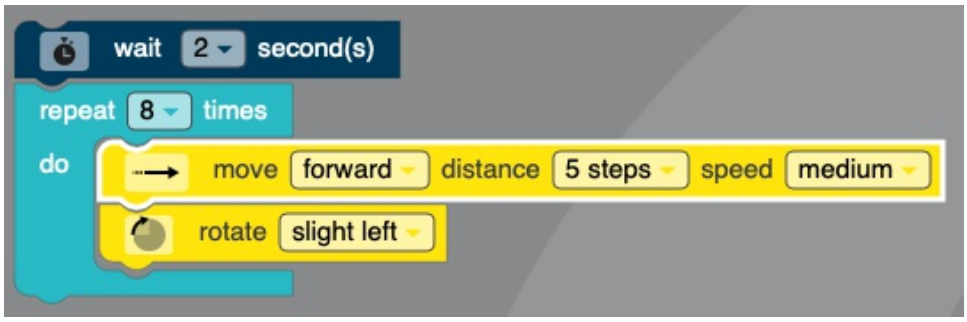
Tipp: Achte auf den richtigen Parameter für die Anweisung „rotate“! Mit „left“ oder „right“ dreht sich Bit im rechten Winkel. Das ist zu viel! Mit „slight left“ oder „slight right“ dreht er sich nur um die Hälfte, also um 45°.

Name:	Schleifen	26

Lösung Übung 13:



Lösung Übung 14:



Schleifen in Schleifen in Schleifen in Schleifen ...

Eine Schleife (loop) ist schon etwas Feines. Sie erspart dir jede Menge Arbeit. Man kann auch mehrere Schleifen hintereinander in einem Programm benutzen. Aber noch interessanter wird es, wenn man eine Schleife in eine andere Schleife packt!

Übung 15:

Jetzt versuche mal, eine Lösung für diese Aufgabe zu finden:
Bit soll wieder ein Quadrat fahren. Die Kantenlänge beträgt 10 Schritte.
Er soll das Quadrat insgesamt 5 mal fahren. Am Ende jedes Quadrates soll er ein Feuerwerk abfeuern.

Übung 16 (für Experten):

Jetzt erstellen wir mal ein größeres Programm:
Bit soll erst 2 mal ein Quadrat fahren, dann 2 mal ein Achteck. Am Endes jedes Quadrates soll Bit für eine Sekunde rot leuchten. Am Ende jedes Achteckes soll er für eine Sekunde grün leuchten. Wenn Bit ganz fertig ist, soll er ein Feuerwerk abschießen.

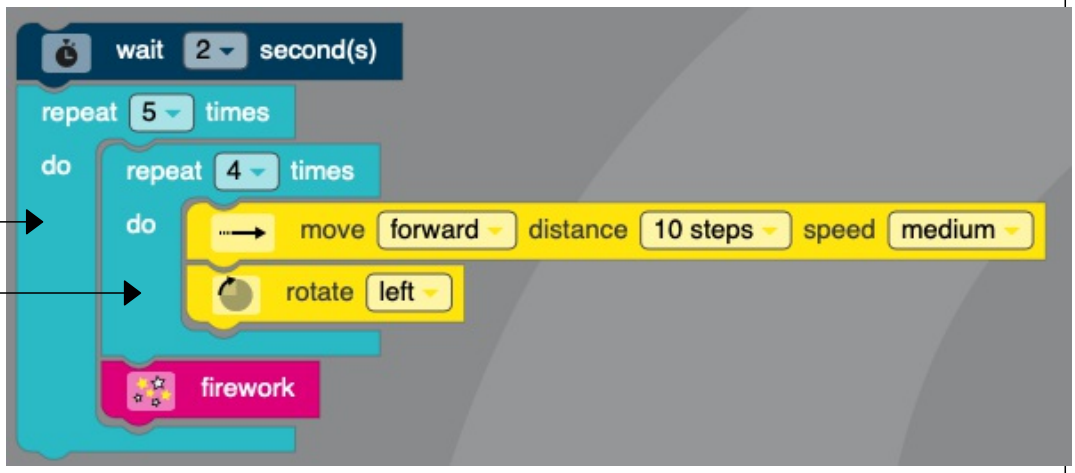
Tipp: Mit der Anweisung „turn top light off“ schaltest du Bits Lampe aus.

Name:	Schleifen: Lösung 15 und 16	27

Lösung Übung 15:

äußere Schleife

innere Schleife



Erklärungen:

Eine Schleife ist hier für das eigentliche Quadrat zuständig. Du kennst sie schon aus Übung 12. Sie enthält die Anweisungen „move“ und „rotate“. Diese Schleife ist zu einer „inneren Schleife“ geworden, denn sie ist jetzt selber eine Anweisung innerhalb einer anderen Schleife. Die andere Schleife wird fünf mal wiederholt, weil Bit ja 5 mal in Form eines Quadrates fahren sollte. Diese Schleife wird „äußere Schleife“ genannt, weil sie eine andere Schleife enthält. Neben der inneren Schleife enthält sie auch das Feuerwerk. Bit feuert es so jedes Mal ab, wenn ein Quadrat fertig ist.

Lösung Übung 16:

Puh, das war schon eine kleine Herausforderung, oder?
(Das Programm ist auf der nächsten Seite abgedruckt!)

Erklärungen:
1: Hier wird die Lampe ausgeschaltet.

2: Diese Schleife lässt Bit ein Quadrat fahren (innere Schleife)

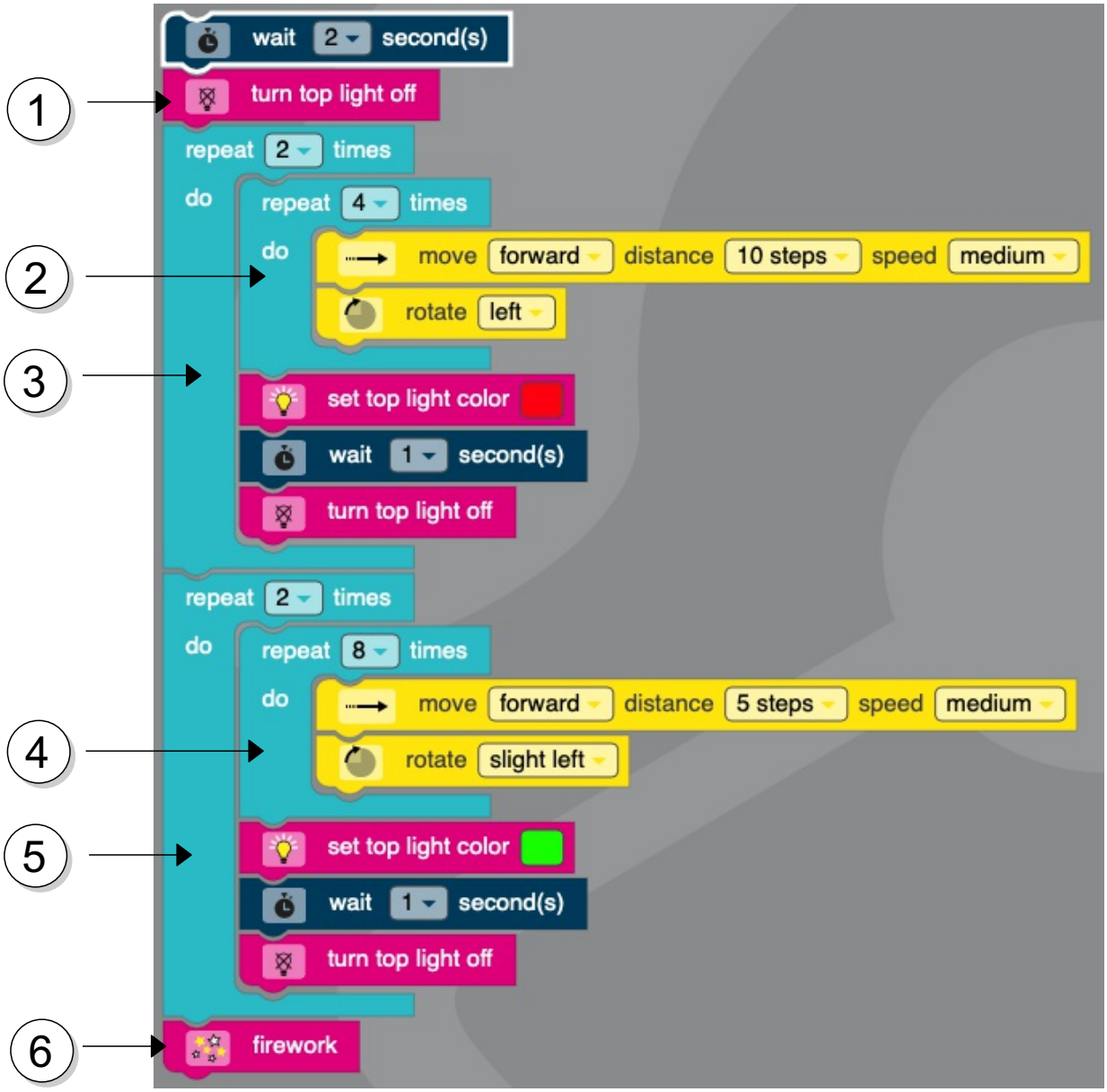
3: Diese Schleife sorgt dafür, dass die innere Schleife (2) 2 mal ausgeführt wird. Danach lässt sie das Licht eine Sekunde lang rot leuchten und schaltet es wieder aus. Sie ist eine äußere Schleife.

4: Diese Schleife lässt Bit ein Achteck fahren. (innere Schleife)

5: Diese Schleife sorgt dafür, dass die innere Schleife (4) 2 mal ausgeführt wird. Danach lässt sie das Licht eine Sekunde lang grün leuchten und schaltet es wieder aus. Sie ist eine äußere Schleife.

6: Nach der ganzen Arbeit haben wir uns ein Feuerwerk verdient!

Name:	Schleifen: Lösung 16	28



Name:	Stufe 3	29

Stufe 3!

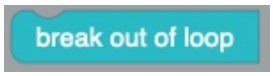
Weiter geht´s mit Stufe 3!

Wenn du die Stufe 3 aktivierst, dann siehst du bei den Anweisungen gleich zwei neue Bereiche:

- Line Navigation
- Logic

Außerdem ist bei „Movement“ eine neue Anweisung mit einem super langen Namen hinzugekommen:

Bei den Loops, die du im letzten Kapitel kennen gelernt hast, ist auch eine neue Anweisung hinzugekommen.



Das ist eine ganze Menge Neues auf einmal. Wir beginnen mit dem neuen Bereich „Line Navigation“:

Zurück zu Linien und Farben

Im ersten Teil hast du Bit programmiert, indem du Linien gezeichnet hast. Anweisungen hast du Bit über Farbcodes gegeben. Das hatte den Vorteil, dass Bit sehr genau über die Linien fahren konnte. Du konntest Bit aber zum Beispiel nicht sagen, dass er an einer Kreuzung abwechselnd links oder rechts abbiegen sollte.
Woran das liegt: Ganz einfach! Wenn du einmal einen Code gemalt hattest, dann ist er auch dann noch gültig, wenn Bit zum zweiten Mal über ihn fährt. Du kannst deine Anweisung also nicht ändern!
Ab dem zweiten Teil des Kurses haben wir keine Linien und Farben mehr benutzt. Stattdessen hast du Bit Anweisungen über OzoBlockly gegeben. Aber du hast bestimmt schon gemerkt, dass Bit nicht immer sehr genau gefahren ist. Falls nicht, lass Bit doch einfach mal eine sehr lange Strecke geradeaus fahren. Vermutlich wirst du feststellen, dass Bit eine leichte Kurve fährt. Das ist bei jedem Bit etwas anders.

Es wäre doch toll, wenn man beides miteinander verbinden könnte! Also:

- Bit fährt an den Linien entlang.
- Bit erhält die Anweisungen aber über OzoBlockly programmiert, nicht über Farbcodes.

Und genau das geht! Und zwar mit den Anweisungen aus dem Bereich:

Name:	Stufe 3	30

Dann testen wir jetzt doch mal die Anweisungen aus dem Bereich „Line Navigation“:

Diese drei neuen Anweisungen stehen uns zur Verfügung, um Bit über Linien zu steuern:

Folge der Linie bis sie unterbrochen oder beendet wird!

Biege an der nächsten Kreuzung ab! Mit dem Parameter teilst du Bit mit, in welche Richtung er abbiegen soll.

Folge Linien in dieser Geschwindigkeit! Der Parameter gibt an, in welcher Geschwindigkeit Bit der Linie folgen soll.

Übung 17:

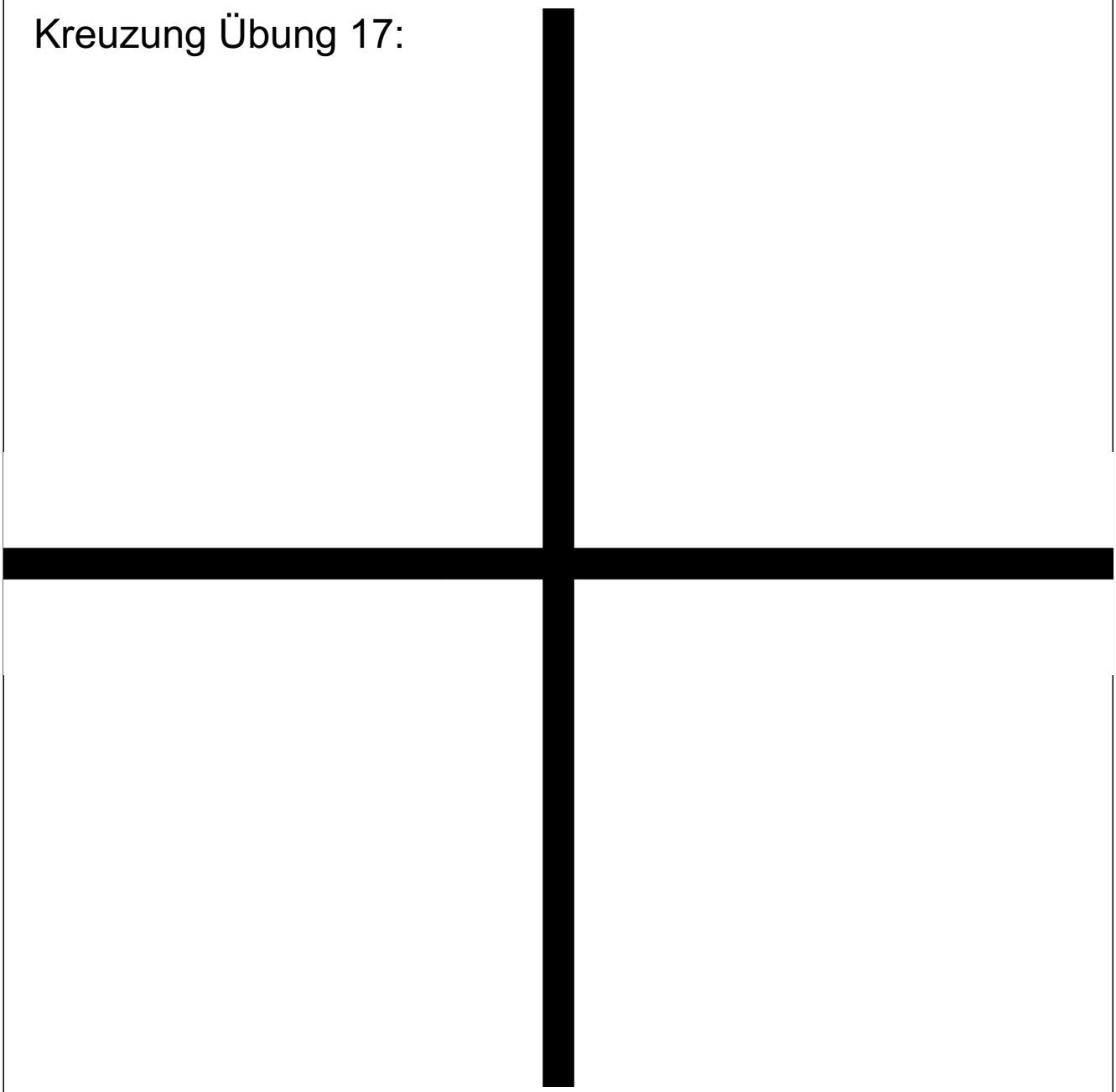
Auf der nächsten Seite habe ich dir eine einfache Kreuzung eingefügt. Du kannst natürlich auch eine eigene Kreuzung malen.

Versuche ein Programm zu erstellen, so dass Bit der Linie folgt und an der Kreuzung links abbiegt.

Die Anweisung „set-line-following speed to“ benötigst du hier noch nicht.

Name:	Line Navigation	31

Kreuzung Übung 17:

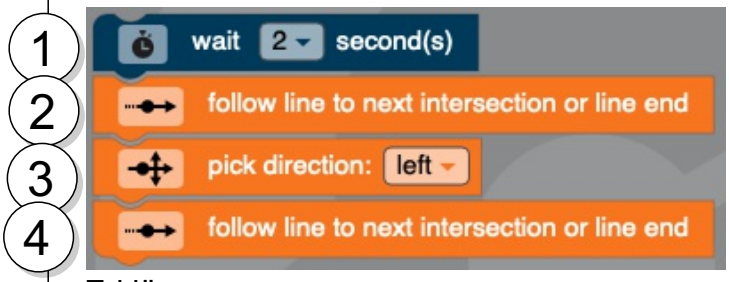


Tipp 1:
Denke daran, Bit mit einem „Doppelklick“ (zweimal schnell hintereinander drücken) anzuschalten. Wenn du den Schalter nur einmal drückst, dann startet Bit im normalen Modus wie aus Teil 1. Er folgt dann einfach der Linie und sucht nach Farbcodes.

Tipp 2:
Bit bleibt einfach stehen und fährt nicht los? Hast du Bit vielleicht direkt angewiesen, dass er links abbiegen soll? Das kann er aber nur, wenn er schon an der Kreuzung ist. Er muss also erst beginnen, der Linie zu folgen. Sonst kommt er nie an der Kreuzung an, um dort abzubiegen.

Name:	Line Navigation	32

Lösung Übung 17:



Erklärung:

1. Das Programm beginnt mit der üblichen Pause von 2 Sekunden, damit du genügend Zeit hast, Bit in Ruhe auf das Blatt zu stellen.

2. Hier soll Bit der Linie folgen, bis die Linie endet oder unterbrochen wird. Bit fährt daher genau bis zur Kreuzung. Dort führt er die nächste Anweisung aus.

3. Mit der dritten Anweisung sagen wir Bit, dass er an der Kreuzung links abbiegen soll.

4. Nachdem Bit sich in Anweisung 3 gedreht hat, soll er jetzt wieder der Linie folgen.

Übung 18:

In der Erklärung habe ich dir bei Punkt zwei gesagt, dass Bit bei der Anweisung

genau bis zur Kreuzung fährt. Kam dir das nicht komisch vor? Die Linie ist doch gar nicht unterbrochen! Hinter der Kreuzung geht sie schließlich weiter und es gibt keine Lücke in der Linie!

Aber es ist wirklich so! Für Bit ist auch eine Kreuzung eine Unterbrechung. Und das kann ich dir auch beweisen! Teste mal dieses Programm und lasse Bit damit über eine Kreuzung fahren!

Bit bleibt genau an der Kreuzung stehen. Damit habe ich dir bewiesen, dass für Bit auch eine Kreuzung eine Unterbrechung ist.

Name:	follow line to intersection or line end	33

Übung 19:

Was könnte für Bit denn noch alles eine Unterbrechung der Linie sein? Nutze das folgende Programm, um zu untersuchen, was für Bit noch alles eine Unterbrechung ist:

Was könnte für Bit denn noch alles eine Unterbrechung der Linie sein?

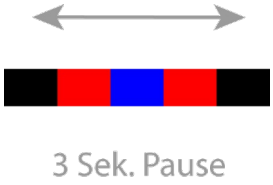
Kreuze deine Ergebnisse in der Tabelle an!
Wenn dir zusätzlich etwas einfällt, kannst du es in die letzten drei Zeilen einfügen.

zu testen:	Unterbrechung	keine Unterbrechung
ein T-Kreuzung		
ein Wechsel der Linienfarbe		
ein Farbcode (zum Beispiel: 3 Sekunden Pause“)  3 Sek. Pause		
eine andere Hintergrundfarbe		
eine Lücke in der Linie		
eine Ecke		
eine Kurve		

Name:	follow line to intersection or line end	34

Lösung Übung 19:

Ich bin beim Testen mit dem Programm zu diesen Ergebnissen gekommen:

getestet	Unterbrechung	keine Unterbrechung
eine T-Kreuzung	X	
ein Wechsel der Linienfarbe		X
ein Farbcode (zum Beispiel: 3 Sekunden Pause“)  		X (einige Farbcodes funktionieren)
eine andere Hintergrundfarbe	verursacht Fehler	
eine andere Linienfarbe		X
eine Lücke in der Linie	X	
eine Ecke		X
ein Kurve		X

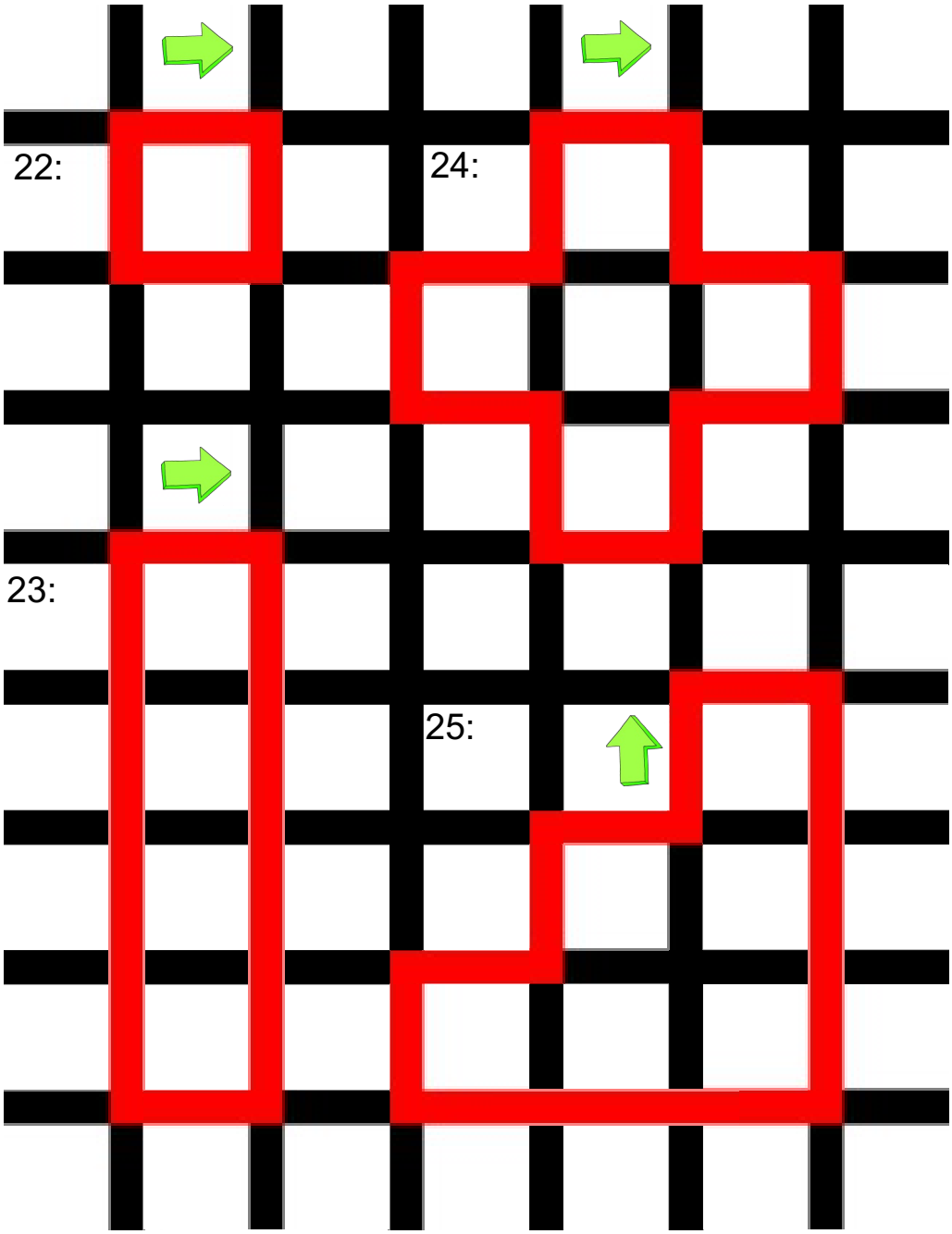
Name:	follow line to intersection or line end	35

Loops, Loops, Loops

Oft ist das Programmieren von Schleifen gar nicht das Schwierige. Manchmal fällt es schon schwer zu erkennen, was genau sich eigentlich wiederholen soll. Darum machen wir jetzt mehrere kleinere Übungen.

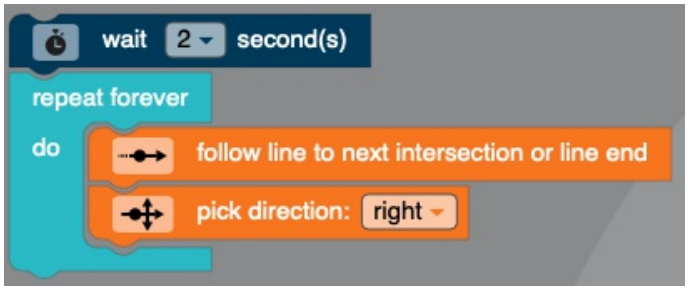
Übung 22-25:

Schreibe Programme, so dass Bit unendlich oft die rote Bahn fährt. Verwende dabei möglichst viele Loops!
Achte unbedingt darauf, dass du Bit beim grünen Pfeil beginnen lässt! Sonst passen die Lösungen nicht, die ich für dich erstellt habe!



Name:	follow line to intersection or line end	36

Lösung 22:



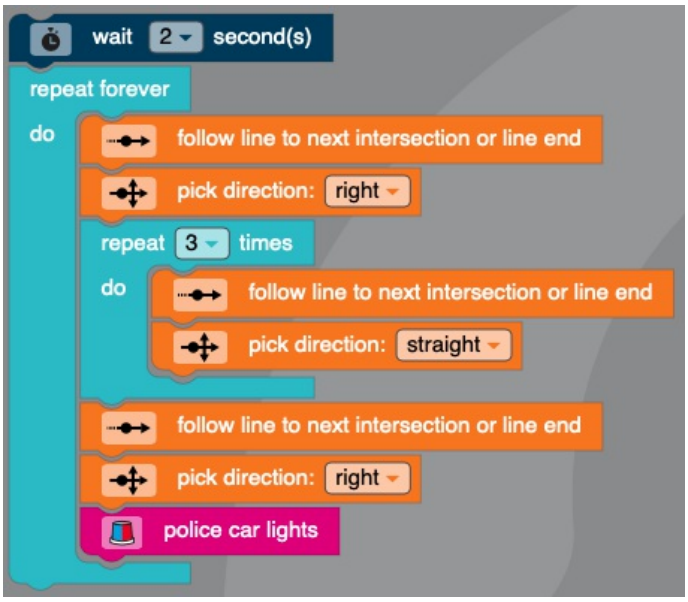
Lösung 23:

Innerhalb der Schleife fährt Bit eigentlich nur das halbe Rechteck ab. Weil die andere Hälfte aber genau gleich ist, reicht das aus. Füge am Ende der Schleife noch einen Lichteffekt ein. Dann kannst du das viel besser sehen.

Innerhalb der Schleife wiederholen sich die Anweisungen

- Follow line to ...
- pick direction: straight

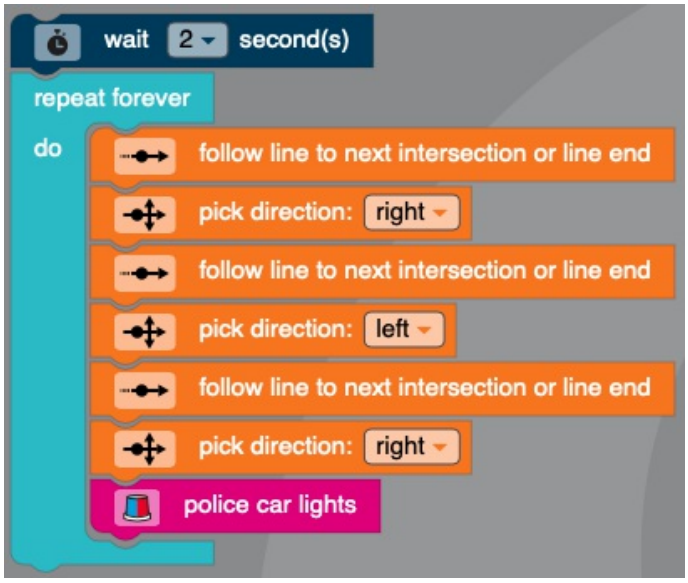
drei mal. Daraus könnte man noch eine innere Schleife basteln.



Mit der inneren Schleife und dem Lichteffekt sieht meine Lösung so aus.

Name:	follow line to intersection or line end	37

Lösung 24:



Am Ende der Schleife habe ich dir wieder einen Lichteffekt eingebaut, damit du besser sehen kannst, wann Bit einen Schleifendurchlauf beendet hat.

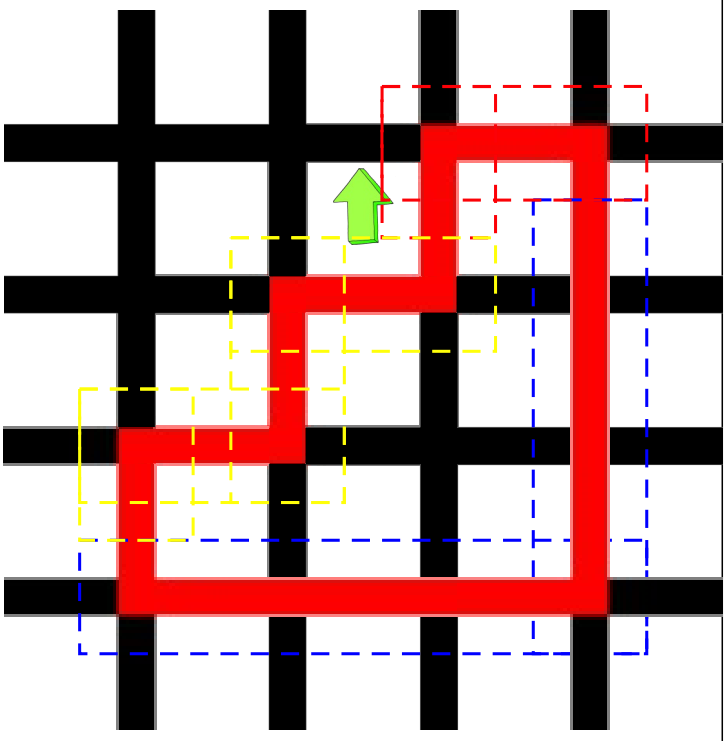
Lösung 25:

Huch! Mit Schleifen ist Übung 25 ganz schön kompliziert! Ich habe drei innere Schleifen benutzt. Damit du besser sehen kannst, wofür welche Schleife zuständig ist, habe ich dir die Bahn noch einmal mit Farben markiert:

- Die innere Schleife 1 ist für den roten Weg zuständig.
- Die innere Schleife 2 ist für den blauen Weg zuständig.
- Die dritte innere Schleife ist für den gelben weg zuständig!

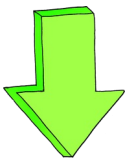
Mein Programm dazu siehst du auf der nächsten Seite.

Ich lasse Bit auch für jede Schleife in der passenden Farbe leuchten.



Name:	follow line to intersection or line end	38

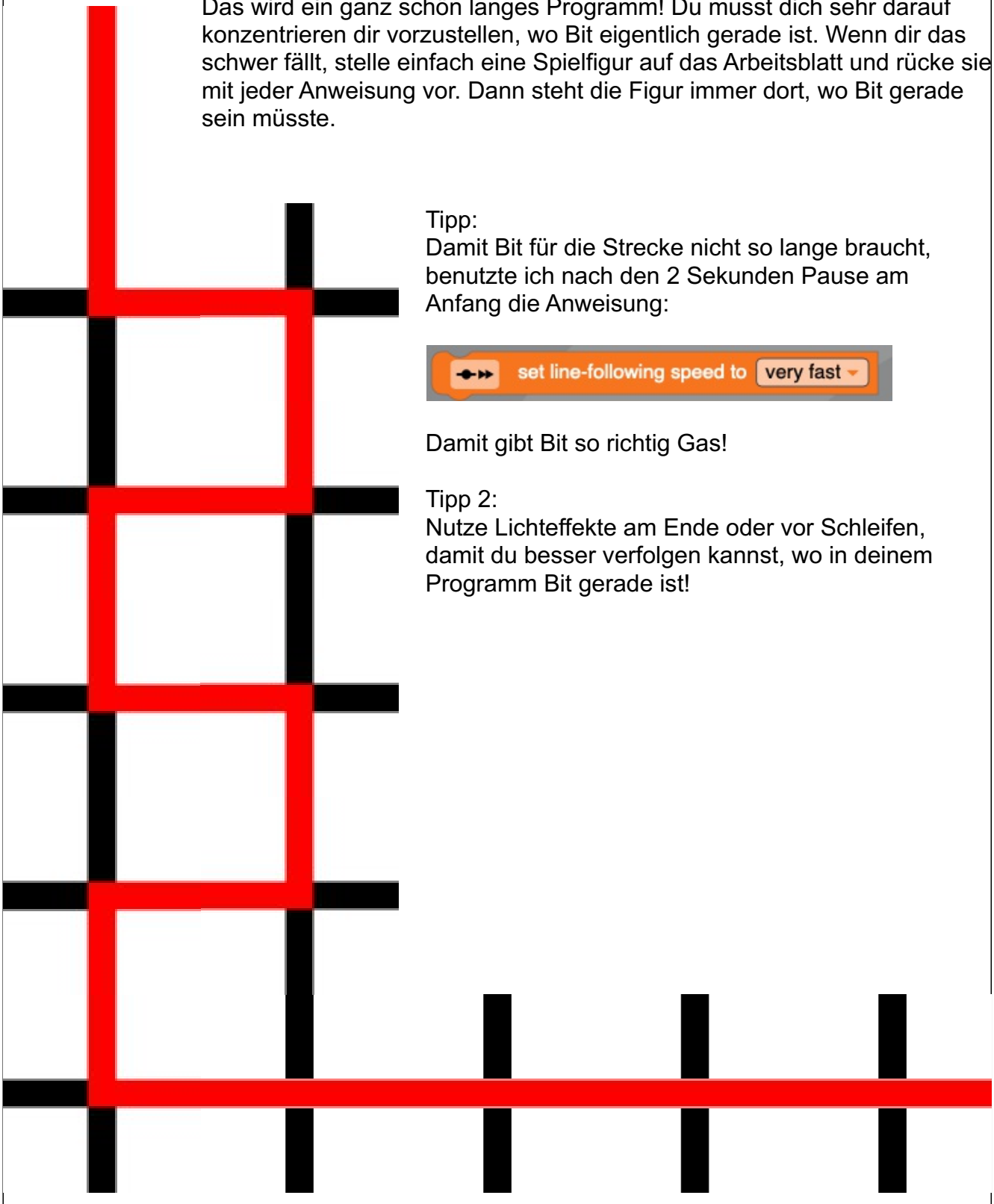
Name:	Line Navigation und Schleifen	39



Übung 26:

Hier habe ich dir eine längere Bahn gebastelt.
Schreibe ein Programm, so dass Bit der roten Strecke folgt!

Das wird ein ganz schön langes Programm! Du musst dich sehr darauf konzentrieren dir vorzustellen, wo Bit eigentlich gerade ist. Wenn dir das schwer fällt, stelle einfach eine Spielfigur auf das Arbeitsblatt und rücke sie mit jeder Anweisung vor. Dann steht die Figur immer dort, wo Bit gerade sein müsste.

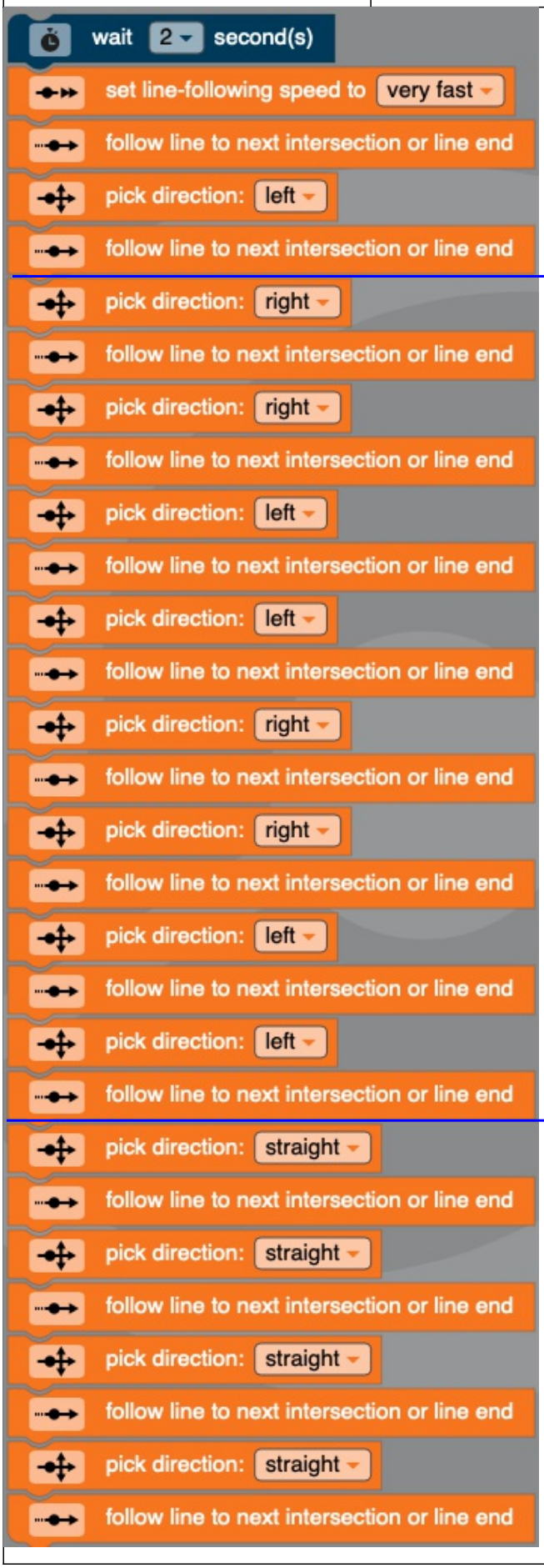


Tipp:
Damit Bit für die Strecke nicht so lange braucht, benutzte ich nach den 2 Sekunden Pause am Anfang die Anweisung:

Damit gibt Bit so richtig Gas!

Tipp 2:
Nutze Lichteffekte am Ende oder vor Schleifen, damit du besser verfolgen kannst, wo in deinem Programm Bit gerade ist!

Name:	Line Navigation und Schleifen	40



Lösung 26:

Start:
Bit macht seine übliche Pause, die Geschwindigkeit wird auf „sehr schnell“ eingestellt. Dann beginnt Bit die Linie entlang zu fahren.

Hin und her:
Jetzt wiederholt sich immer das Gleiche: Bit biegt immer zwei mal rechts und dann zwei mal links ab.
Und auch das wiederholt sich wieder je 2 mal.

Ende:
Am Ende muss Bit noch an 4 Kreuzungen geradeaus fahren.

Was für eine Quälerei! So viele Anweisungen! Aber es geht auch schöner! Sieh dir unser Programm mal genauer an. Viele Bereiche wiederholen sich immer wieder. Darum kann man das Programm viel kürzer schreiben.
Ganz klar! Mit Loop!
Denn natürlich kann du die Anweisungen aus „Loops“ auch mit den Anweisungen aus „Line Navigation“ zusammen benutzen.

Übung 27:

Versuche, das Programm aus Lösung 26 viel kürzer zu schreiben!
Benutzte dafür Loops (Schleifen)!

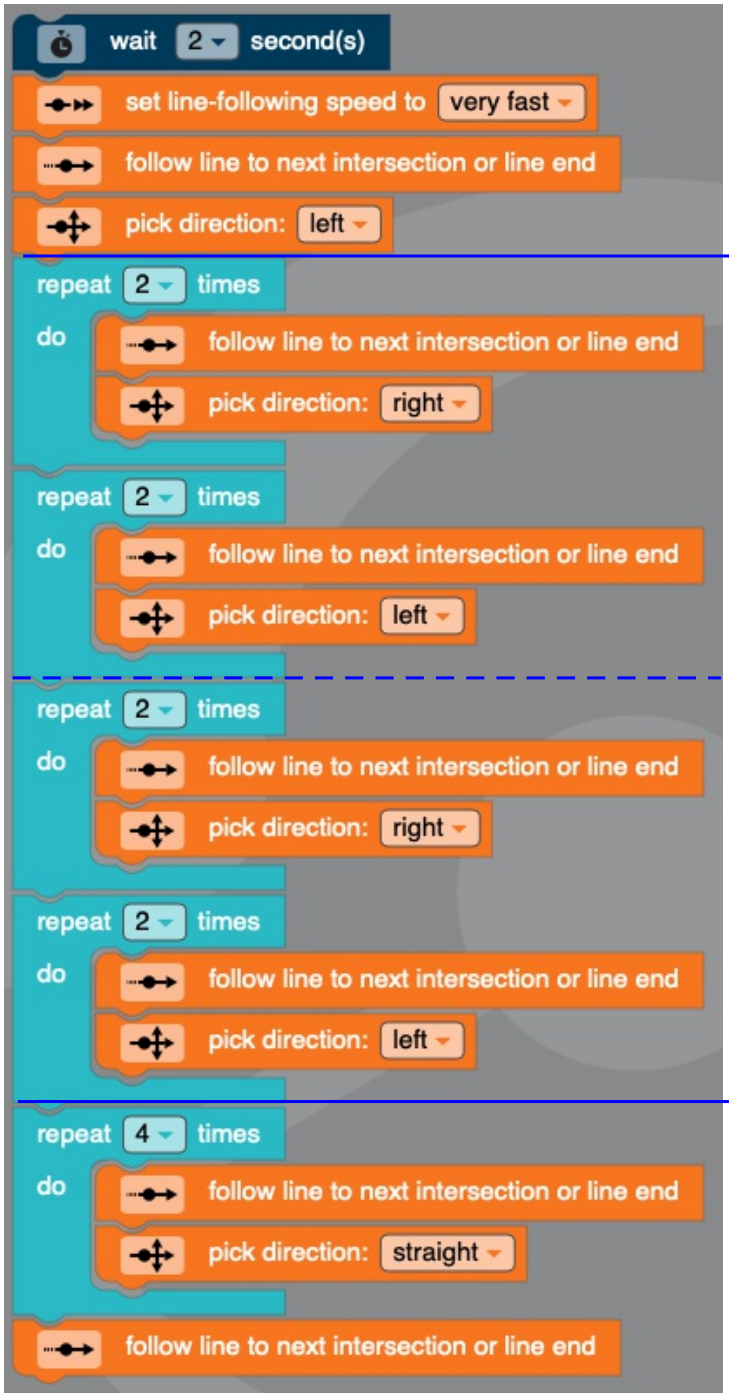
Tipp:

Beschrifte auf dem Blatt (Seite 39) jede Ecke, so dass du einfach ablesen kannst, ob Bit links oder rechts abbiegen muss. Nutze für „links“ und „rechts“ unterschiedliche Farben. So kannst du einfacher erkennen, was genau sich wiederholt.

Auf der nächsten Seite zeige ich dir meine Lösungen.

Name:	Line Navigation mit Schleifen	42

Lösung 27:



Start:
Bit macht seine übliche Pause, die Geschwindigkeit wird auf „sehr schnell“ eingestellt. Dann beginnt Bit die Linie entlang zu fahren.

Hin und her:
Jetzt wiederholt sich immer das Gleiche: Bit biegt immer zwei mal rechts und dann zwei mal links ab.
Diese Anweisungen sind nun in den Schleifen.

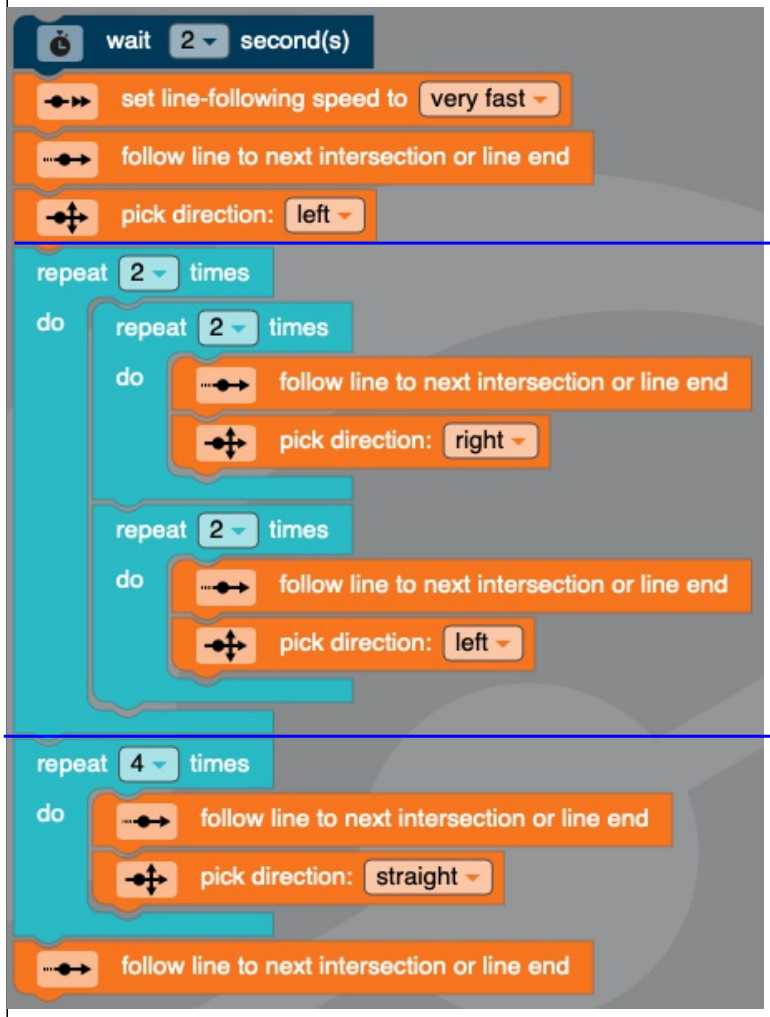
Und auch das wiederholt sich wieder je 2 mal! Die erste und die dritte Schleife sind genau gleich. Die zweite und die vierte Schleife sind auch genau gleich. Daraus können wir noch eine Schleife bauen!

Das Ergebnis siehst du auf Seite 41!

Ende:
Am Ende muss Bit noch an 4 Kreuzungen geradeaus fahren.

Name:	Line Navigation mit Schleifen	43

Lösung 27:



Jetzt werden die Schleifen, die auf Seite 42 noch zweimal hintereinander im Programm standen, durch eine weitere Schleife wiederholt.
Erinnerst du dich noch? So etwas nennt man eine „äußere Schleife“!

Das war jetzt schon ganz schön schwierig!

Wenn du das Programm mit den vielen Schleifen liest und nicht so recht verstehst, was Bit mit all den Anweisungen tun soll: **Das ist ganz normal:**
Schleifen sparen dir zwar immer eine Menge Arbeit beim Erstellen eines Programms, dafür wird das Lesen eines Programms mit vielen Schleifen umso schwieriger!

Wenn du mit Schleifen arbeitest, nutze Bits Licht in verschiedenen Farben. Oft kannst du dann besser sehen, an welcher Stelle in deinem Programm sich Bit gerade befindet! Wenn das Programm fertig ist, kannst du die Anweisungen für das Licht ja wieder entfernen.

Manchmal sind Programme sogar so kompliziert, dass man selbst bei einem eigenen Programm am nächsten Tag schon nicht mehr versteht, wofür man eigentlich all die Schleifen benutzt hat. Ich programmiere schon seit 30 Jahren und es passiert mir immer noch regelmäßig!
Überlege dir beim Schreiben deiner Programme also immer gut, ob du lieber viele Anweisungen benutzt oder wenige Anweisungen in Schleifen.
Beides hat Vorteile aber auch Nachteile!

Name:	Bedingte Anweisungen	44

Bedingte Anweisungen:

Herzlichen Glückwunsch! Du hast tatsächlich bis zum letzten Kapitel durchgehalten und das, obwohl das letzte Kapitel schon echt fortgeschritten war.
In diesem Kapitel wird es wieder etwas einfacher, versprochen!

Bisher haben unsere Programme immer so funktioniert, dass wir Bit gesagt haben, was er machen soll. Das hat er - ganz typisch für Computer - auch ganz genau gemacht. Bit musste darum keine Entscheidungen treffen. Wir wussten immer schon vor dem Programmstart genau, was Bit wann machen soll. Das nennt man „linear“.

Es gibt aber auch manchmal Dinge, die das Programm beeinflussen können, die wir nicht schon vor dem Programmstart wissen. Irgendwie müssen wir Bit dann sagen, was zu tun ist. Denn selber denken kann im Moment noch kein Computer.

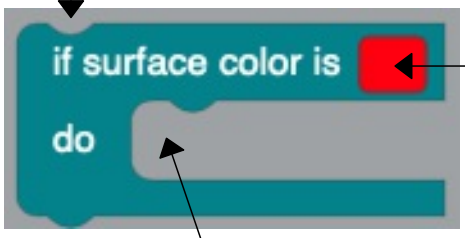
Für genau solche Fälle gibt es „bedingte Anweisungen“. OzoBlockly benutzt diesen Fachbegriff nicht und nennt diesen Bereich der Anweisungen einfach nur „Logic“.

Wenn du den Bereich in Ozoblockly anklickst, siehst du gleich sechs neue Anweisungen. Vier der Anweisungen werden benutzt, wenn man mit Linien programmiert. Zwei werden benutzt, wenn Bit sich frei bewegen kann.

Wir werden nur die beiden Anweisungen für die freie Bewegung benutzen. Die beiden anderen kann man nur sinnvoll mit vielen neuen Dingen nutzen, die im Moment noch zu schwierig wären. - Und ich habe schließlich versprochen, dass es wieder einfacher wird!

Dies sind die beiden Anweisungen aus dem Bereich Logic, die ich euch zeigen möchte:

Falls (if) die Farbe der Fläche unter Bit diese Farbe hat (rot), dann mache (do) dies.



Parameter: Hier kannst du die Farbe auswählen, auf die Bit reagieren soll (hier rot). Nur wenn die Farbe unter Bit gerade genau so (hier rot) ist, führt Bit die Anweisungen aus, die hinter „do“ stehen.

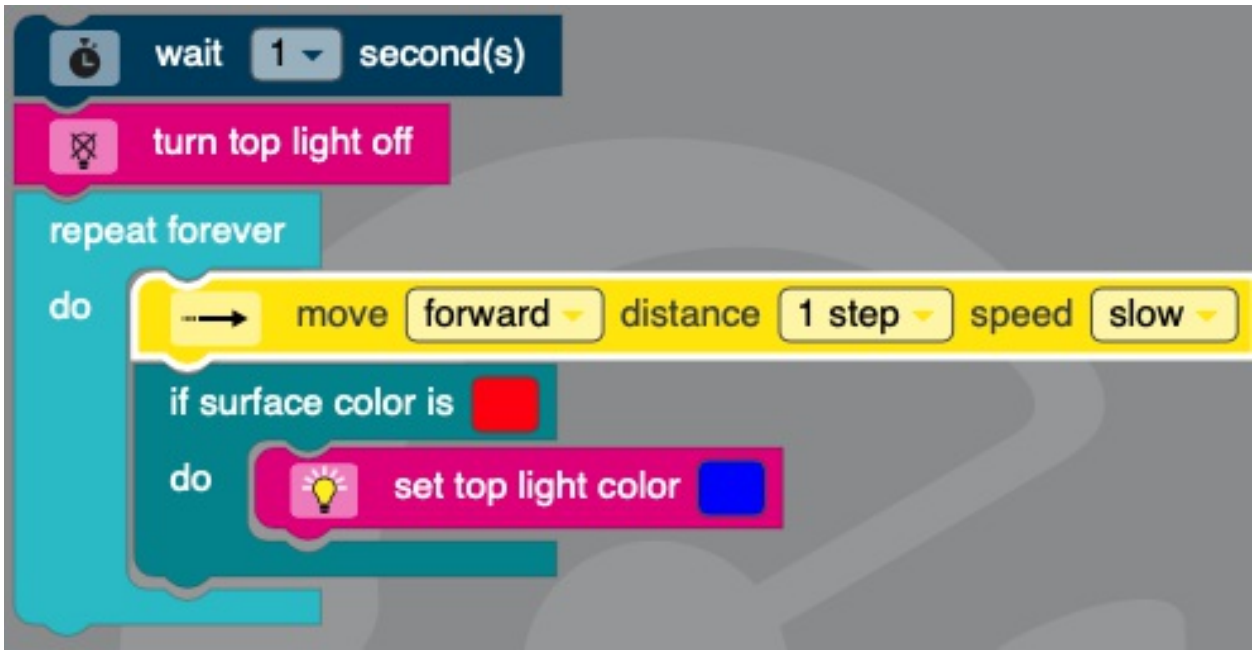
Hier musst du die Anweisungen eintragen, die Bit nur dann ausführen soll, wenn die Bedingung (Oberfläche ist rot) erfüllt ist. Danach geht dein Programm mit den nächsten Anweisungen weiter.
Wenn die Farbe nicht rot ist, wird Bit direkt deine nächsten Anweisungen ausführen.

Name:	Bedingte Anweisungen	45

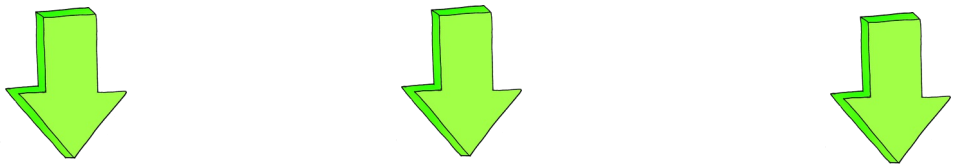
Übung 28:

Das nächste Programm schaltet Bits Lampe aus und lässt ihn in kleinen Schritten losfahren. Dann kommt die bedingte Anweisung. Sie sagt Bit, dass er die Lampe BLAU leuchten lassen soll. Aber nur, falls er über eine ROTE Fläche fährt.

Übergib Bit folgendes Programm:



Lass Bit über die verschiedenen roten Flächen fahren! Was passiert?
Teste verschiedene Geschwindigkeiten! Wiederhole den Versuch mehrmals!



Name:	bedingte Anweisungen	46

Was konntest du beobachten?

Meine Beobachtungen:

Bit erkennt nur große Flächen. Wenn Bit sehr schnell fährt, dann reicht manchmal auch die große Fläche nicht aus. Die Flächen müssen also sehr groß sein, damit Bit sie erkennen kann. Kleinere Flächen erkennt Bit nur, wenn er langsam fährt. Aber auch das gelingt ihm nicht zuverlässig.

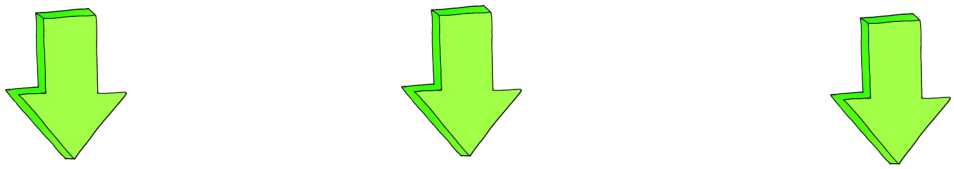
Wenn Bit über die Fläche gefahren ist und die Lampe blau leuchtet, dann bleibt sie auch dann an, wenn Bit nicht mehr über die roten Fläche fährt. Das ist auch kein Wunder. Du hast Bit schließlich an keiner Stelle die Anweisung gegeben, die Lampe wieder auszuschalten!

Name:	Bedingte Anweisungen	47

Übung 29:

Wiederhole den Test aus Übung 28! Aber jetzt sollen die Flächen schwarz sein.

Denke daran, im Programm den Parameter der bedingten Anweisung auf schwarz zu stellen:



Reagiert Bit anders als in Übung 28?

Name:	Bedingte Anweisungen 2	48

Meine Lösung:

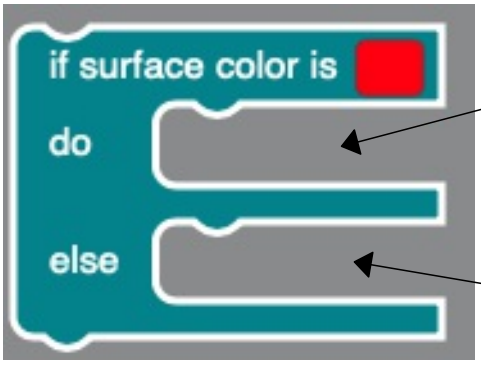
Ja! Bit reagiert anders! Es scheint so zu sein, dass Bit schwarze Flächen viel schneller und zuverlässiger erkennen kann als rote Flächen!

Du kannst ja noch testen, wie Bit auf die Farben Grün und Blau reagiert.

if ... else

Wir haben noch gar nicht über die zweite bedingte Anweisung gesprochen, die ich dir zeigen wollte:

Sie ist der ersten Anweisung sehr ähnlich. Jetzt kann man Bit aber nicht nur sagen, was er tun soll, falls die eingestellte Farbe (der Paramter) genau der Farbe unter ihm entspricht. Hinter „else“ - das bedeutet so viel wie „falls nicht“ - kannst du hier Anweisungen geben, was Bit machen soll, falls die Farben nicht gleich sind!



Hier kommen die Anweisungen hin, die Bit ausführen soll, falls die Farbe stimmt!

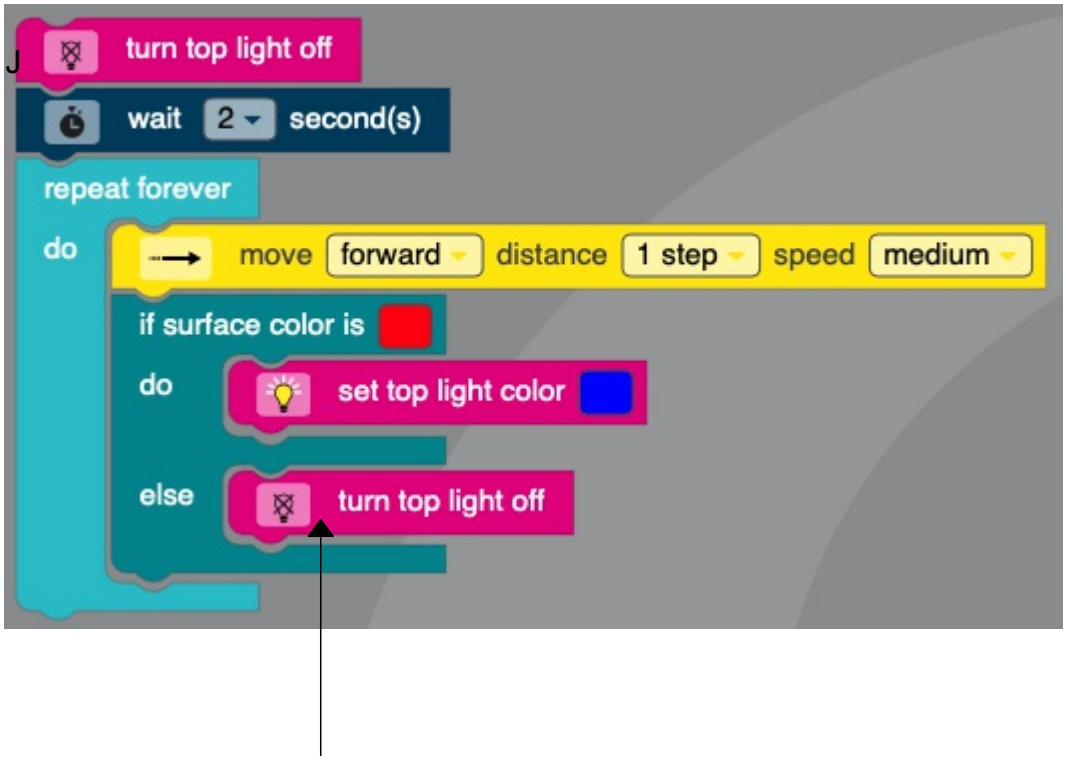
Hier kommen die Anweisungen hin, die Bit ausführen soll, falls die Farbe NICHT stimmt!

Übung 30:

Ändere das Programm aus Übung 28 so, dass Bits Lampe wieder ausgeht, wenn er nicht mehr über die roten Fläche fährt!

Name:	Der gefangene Bit	49

Meine Lösung:



Hier schaltet Bit die Lampe aus, falls die Fläche unter ihm nicht rot ist!

Übung 31

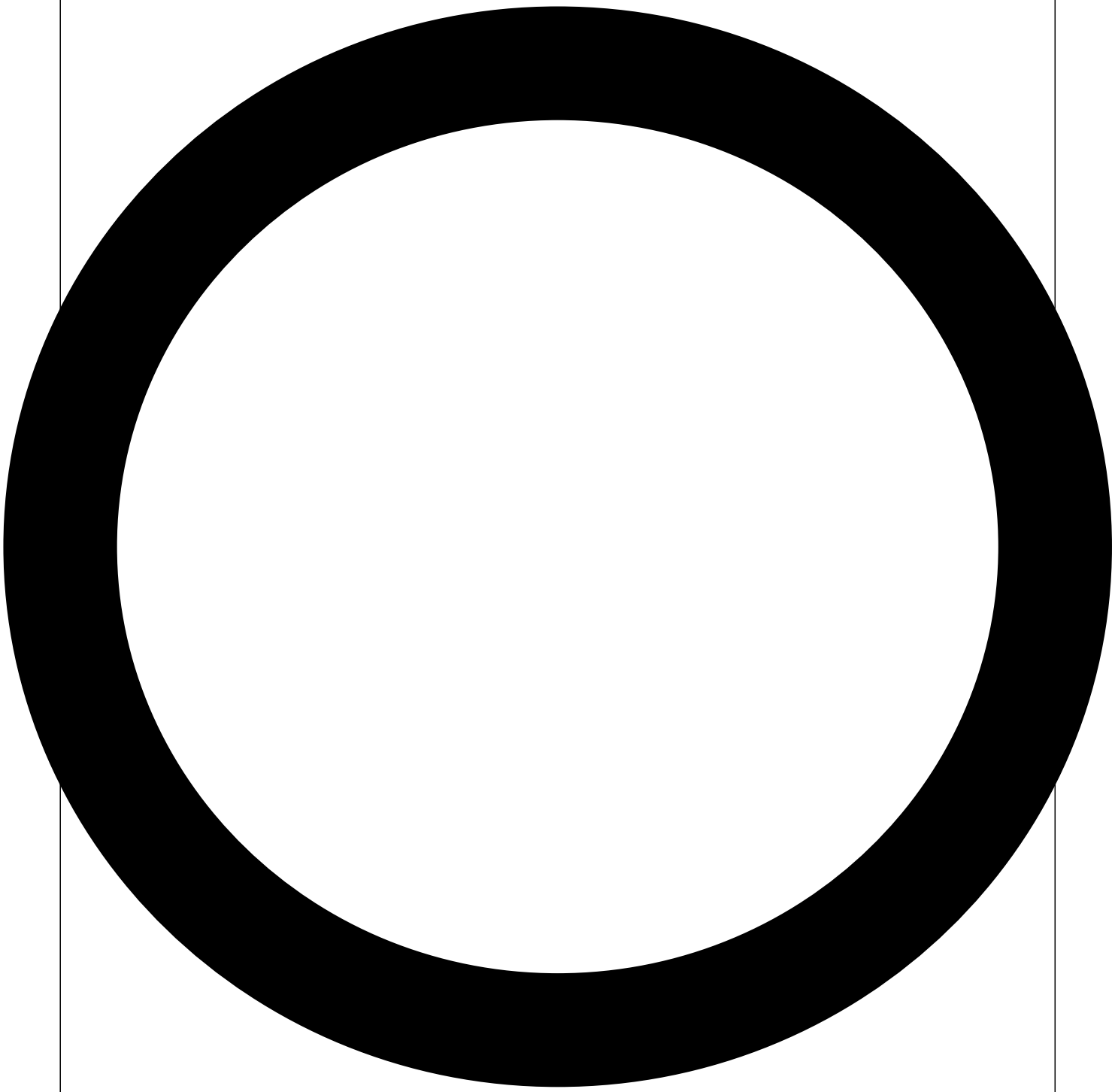
Der gefangene Bit:
Auf der nächsten Seite findest du einen großen schwarzen Kreis.

Versuche folgendes Programm zu schreiben.

Wenn du Bit in die Mitte des Kreises setzt, soll er immer umdrehen, wenn er auf die schwarze Kreislinie trifft!

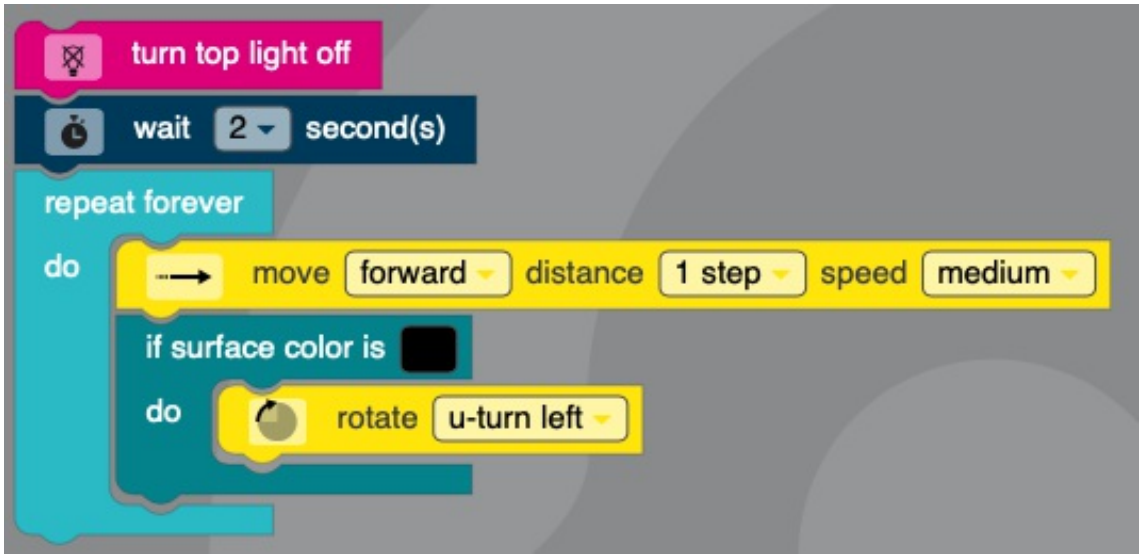
Bit fährt dann im Kreis hin und her!

Name:	Der gefangene Bit	50



Name:	Ein Spielplatz für Bit	51

Meine Lösung:



Übung 32:

Wenn Bit schon im Kreis gefangen ist, dann soll er wenigstens so richtig beschäftigt sein. Bauen wir ihm einen Spielplatz!

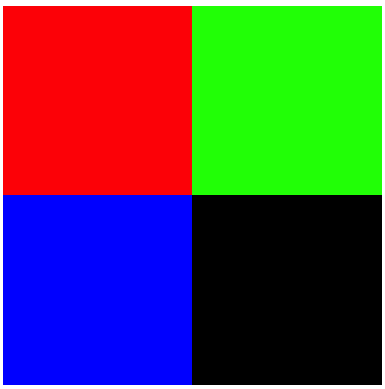
- Male noch weitere Flächen in die Mitte des Kreises hinein!
- Ändere das Programm so:
  Ergänze Anweisungen die Bit sagen, was er machen soll, wenn er über deine neu eingefügten Flächen fährt!

Aber Vorsicht:
Mit einigen Anweisungen kann es dir passieren, dass Bit den Kreis verlässt und dir entwischt!

Name:	ein Spielplatz für Bit	52

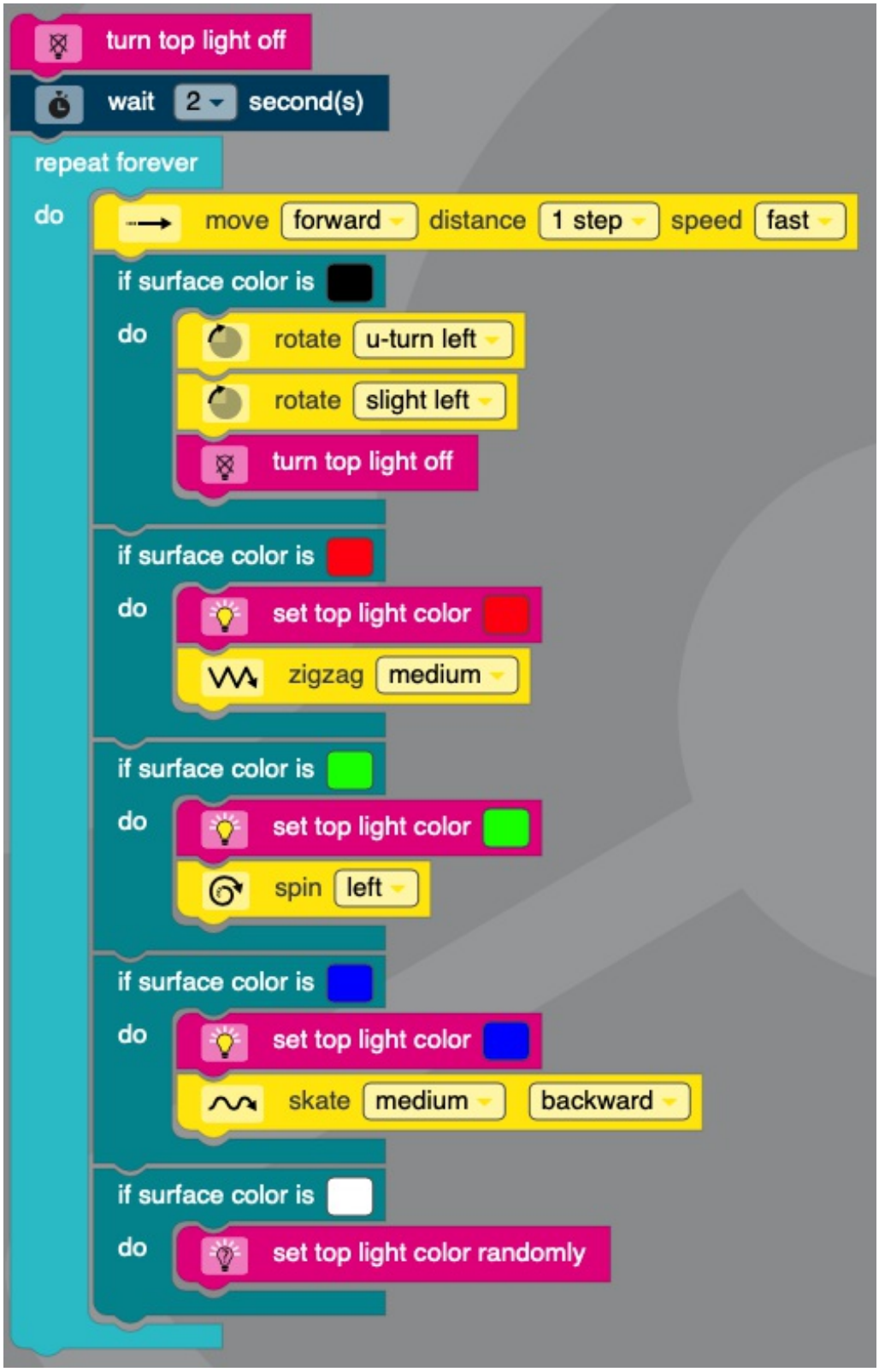
Meine Lösung:

So sieht mein Spielplatz für Bit aus:



Das passende Programm findest du auf der nächsten Seite.

Name:	ein Spielplatz für Bit	53



Name:	Ende	54

Toll! Super! Spitze! Prima!

Du hast bis zum Ende durchgehalten und hast dir ein Feuerwerk verdient!

Leider habe ich gerade keines hier.
Du weißt ja, wie du dir selber eines programmieren kannst!

Du willst noch immer programmieren?

- Auf den Schultablets ist die App „Ozobot Bit“ installiert. Hier kannst du noch einiges entdecken. Du kannst Bit anders einstellen oder Strecken direkt auf dem Ipad entwerfen und Bit auf dem Display des Ipads herumfahren lassen.
- Mit der App „OzobotBitGroove“ kannst du mehrere Bits auf dem Ipad tanzen lassen.
- Das ist dir alles zu einfach? Dann geht es für dich jetzt erst richtig los:
    - Für Ipads gibt es die App „Playgrounds“. Hier kannst du ohne einen echten Roboter zu haben weiter das Programmieren üben. Du steuerst hier keinen echten Roboter, sondern eine Figur auf dem Bildschirm.