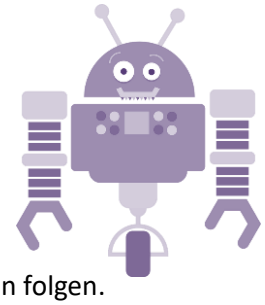
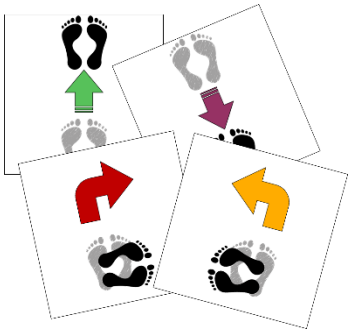
Menschlicher Roboter



Zeitrahmen:	mehrere Vormittage mit kurzen Einheiten
Lehrplan:	Die Kinder … • können formale Anleitungen erkennen und diesen folgen. • können durch Probieren Lösungswege für einfache Problemstellungen suchen und diese auf Korrektheit prüfen. • können die Schritte, um ein Problem zu lösen, in einer Folge von Anweisungen beschreiben. • können einfache Anleitungen und Programme erstellen. • verstehen, dass Informationstechnologie nur vordefinierte Anweisungen ausführen kann [muss aber besprochen werden!]. • verstehen, dass Software/Programme Abfolgen von vordefinierten Anweisungen sind [muss besprochen werden!]. • können ein Programm in der Form von ausführbaren Schritt-für-Schritt Anweisungen erstellen.
Informatikkonzept:	Algorithmen und Programme
Art des Materials:	Spiel
Benötigte Dateien:	Minitafeln (Legekärtchen für die Kinder) Symbole (Symbolkarten für das Labyrinth)
Utensilien:	Wollschnur/Isolierband, Klebeband, Ausdrucke (Kärtchen und Symbole) rote und gelbe Sticker (oder ein rotes und ein gelbes Armband)
Sozialform:	Teamarbeit
Autorin:	Nina Lobnig
Lehr-/Lernziel:	Die Kinder lernen Handlungsabläufe aus vorgegebenen Anweisungen zusammenzusetzen und dadurch das Konzept der Algorithmen spielerisch kennen. Fähigkeiten, die gelernt werden sollen, sind das Lesen, sowie das Erstellen von Anweisungsfolgen. Die Kinder entwickeln bei diesem Spiel ein Programm, bestehend aus kleineren Anweisungen und schnuppern somit in die Welt der Programmierung. *Bei weiteren Erweiterungen kann durch Besprechung des Informatikbezugs auch etwas über Codierung und Methoden (eine Anweisung bestehend aus mehreren anderen) gelernt werden.*
Quellen:	*lizenzfrei von Pixabay und Abbildungen der Informatik-Werkstatt*

Vorbereitung

Vor dem Start der Unterrichtseinheit sollten die Symbole, die später im Labyrinth verteilt werden, sowie die Legekärtchen, welche von den Kindern später auflegen werden, ausgedruckt und laminiert werden (sonst nutzen sie sich sehr schnell ab und können nicht gut wiederverwendet werden). Bei den Legekärtchen (u. a. Vorwärts/Rückwärts/Rechts-/Linksdrehung) muss die Anzahl passend variiert werden, je nach der geplanten Gruppeneinteilung (wie viele Kinder gleichzeitig legen) und der Größe der Aufgaben (Wege, die gelegt werden müssen).

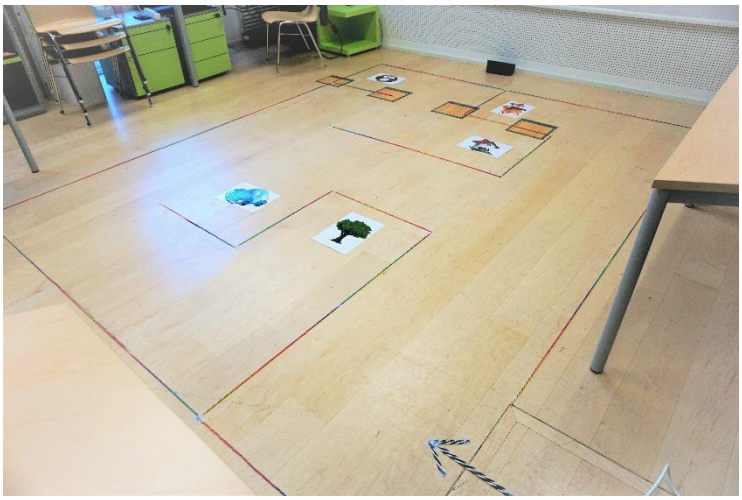


Anweisungskärtchen

Es empfiehlt sich, die Seiten (1-4) für die aufzulegenden Kärtchen getrennt zu drucken, da davon unterschiedliche Anzahlen benötigt werden. Ein **Programmierer*innen-Team**, also jene, die zusammen an einem Programm legen, benötigt mind. 15-20 Vorwärtskärtchen, dies hängt aber vom Labyrinth und den zu gehenden Wegen ab.

Sind die teilnehmenden Kinder noch nicht so geübt, was links und rechts bedeutet, können Sticker in gelb und rot sehr hilfreich sein. Man klebt dem **‚Roboterkind'** auf die linke Hand einen gelben und auf die rechte Hand einen roten Sticker – die Farben entsprechen den Farben auf den Anweisungskärtchen. Die **‚Computer'** lesen dann „drehe dich nach rot/gelb" anstatt „drehe dich nach rechts/links" vor. Anstatt der Sticker können aber auch zwei Armbänder – in gelb und rot – verwendet werden.

Vor Beginn der Durchführung muss zudem überlegt werden, wie das Labyrinth aussehen soll und dann dieses mittels Wollschnüren (für temporäre Verwendung) oder Isolierband (länger haltbar) am Boden aufgeklebt werden. Am Schluss legt man noch die groß ausgedruckten Symbole (wie Hexe, Baum usw.) als Ziele in das Labyrinth (ggf. klebt man diese auch fest). Ein solches Labyrinth könnte wie folgt aussehen:



Beispiel eines solchen Labyrinths,
CC-BY-SA Informatik-Werkstatt

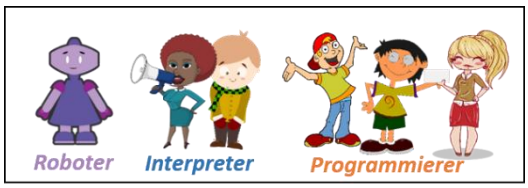
Tipp: Um die Felder recht genau und gerade zu kleben, kann man z.B. 5 Seiten A3-Papier zu Quadraten schneiden und als Messgrundlage verwenden. Für eine Labyrinthseite legt man die fünf Quadrate dann aneinander und zieht die Linie (mit Wollschnur oder Isolierband) daneben, für Wandteile verwendet man einzelne Quadrate.

Umsetzung

Am Beginn erklärt am besten die Lehrperson den Ablauf der Unterrichtseinheit. Es wird erklärt, dass es darum geht, einen Roboter durch Legen von Anweisungen (Vorwärts, Rechtsdrehung, Vorwärts, Vorwärts, …) zu einem zuvor festgelegten Ziel zu steuern. Dabei gibt es **drei verschiedene Rollen**: jene der ‚Programmierer*innen', welche sich die Anweisungen überlegen und dementsprechend die Kärtchen legen, jene der ‚Computer', welche die gelegten Anweisungen nach dem „Programmieren" vorlesen und jene des ‚Roboters', welcher die vorgelesenen Anweisungen ausführt. *Es kann hierbei mehrere ‚Programmierer*innen'-Teams geben, welche im Team zusammen parallel (evtl. auch in Konkurrenz) zu den anderen Teams programmieren.*

Des Weiteren wird besprochen, was ein Schritt und eine Drehung bedeuteten: Ein Schritt entspricht einem Feld, man steigt immer von der Mitte eines Feldes zur Mitte des nächsten (siehe auch Abbildung ‚Ausführung des Beispielprogramms' auf Seite 4). Bei den Drehungen dreht man sich am Stand – man bleibt also am gleichen Punkt am Boden stehen (siehe auch Fuß-Abbildungen auf den Anweisungskärtchen). Es wird auch angemerkt, dass die Wollfäden die Wände des Labyrinths darstellen und diese nicht übertreten werden sollten.

Zuerst teilt man die Mitspielenden ihren Rollen zu (oder lässt sie sich selbst zuteilen). Dazu wird, wie oben erklärt, eine Gruppe an ‚Programmierer*innen ', ein paar ‚Computer' und ein ‚Roboter' benötigt. Diese Rollenverteilung wird in jeder Runde verändert. Hat man z.B. sechs Kinder, so könnte eine Aufteilung wie folgt aussehen:

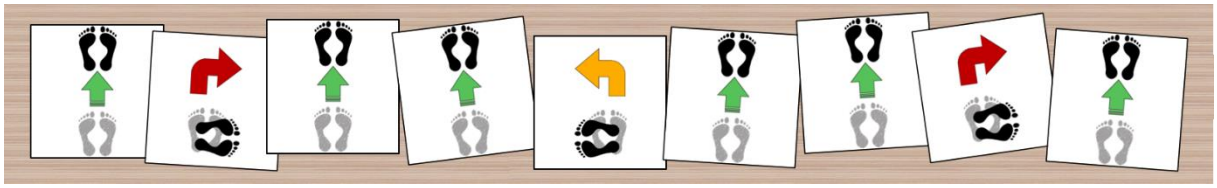


Beispiel einer Gruppeneinteilung,
Grafik adaptiert, Quelle: CC-BY-SA Informatik-Werkstatt

Vor dem Labyrinth stellt man in der Vorbereitung noch zwei oder drei Tische für das Legen der Kärtchen auf. Die ‚Programmierer*innen' stellen sich dann zu dieser „Tisch-Insel", die ‚Computer' stehen etwas abseits und geben das Ziel vor, z.B. das Auto. (Hat man mehrere Teams, wird entweder ein gemeinsames Ziel für alle Teams oder für jedes Team ein anderes gewählt.) Danach stehen die ‚Computer', gleich wie der ‚Roboter' seitlich und sehen den ‚Programmierer*innen' zu und warten. So sieht die Aufstellung aktuell aus:

Aufstellung beim Programmieren
CC-BY-SA Informatik-Werkstatt

Nun machen sich die ‚Programmierer*innen' an die Arbeit und legen auf ihren Tischen durch Absprache untereinander und gemeinsames Überlegen die Anweisungen der Reihe nach hin. Während dieser Phase haben die ‚Computer' und der ‚Roboter' nichts zu tun, können aber den ‚Programmierer*innen' zusehen. Eine von den ‚Programmierer*innen' gelegte Anweisungsfolge könnte z.B. wie folgt aussehen:

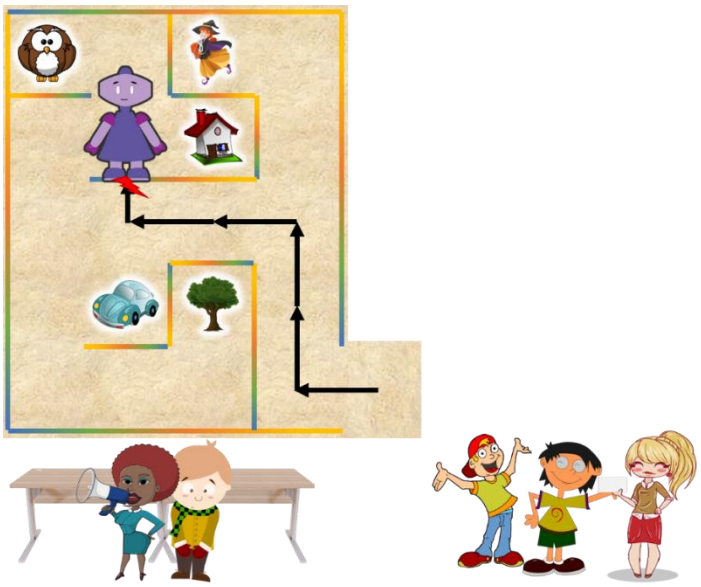


Beispiel einer Programmierung,
eigene Grafik angelehnt an eine von: CC-BY-SA Informatik-Werkstatt

Sind die ‚Programmierer*innen' mit ihrem Programm fertig, so rufen sie die ‚Computer' herbei und tauschen ihren Platz mit den ihnen. Es stehen also nun die ‚Computer' vorm Tisch und die ‚Programmierer*innen' seitlich und sehen dem weiteren Ablauf zu (sie dürfen ihr Programm ab sofort nicht mehr ändern). Die ‚Computer' rufen nun abwechselnd die vor ihnen liegenden Anweisungen dem ‚Roboter' zu, welcher diese ausführt (und idealerweise den gebastelten Roboterhelm trägt).

Sollte der „Code" (die Handlungsanweisungen) richtig sein, so darf gratuliert werden und die Rollen werden getauscht und das Spiel beginnt von vorne (mit Zieldefinition, Anweisungslegen usw.). Ist der Code fehlerhaft - so wie in unserem Beispiel - so stoppt das Programm, die ‚Programmierer*innen' nehmen ihren Platz wieder ein und versuchen den Fehler zu korrigieren (im Idealfall sagen die ‚Computer' aber nicht, wo sie zu lesen aufgehört haben).

Der Roboter in unserem Beispiel geht beim Zurufen folgenden Weg:

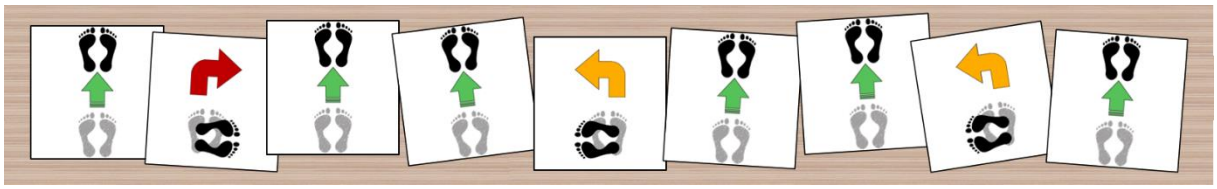


Ausführung des Beispielprogramms,
CC-BY-SA Informatik-Werkstatt

Man sieht: Leider führt die Anweisungsfolge nicht zum erwünschten Ergebnis, also nicht zum Auto. Der ‚Roboter' crasht gegen eine der Wände, dies bringt die ‚Computer' dazu aufzuhören zu lesen (auch, wenn danach noch weitere Anweisungen zum Vorlesen liegen würden). Der ‚Roboter' geht wieder zum Startpunkt zurück und die ‚Computer' tauschen wieder mit den ‚Programmierer*innen'. Die Aufstellung ist also wieder die gleiche, wie auf der Abbildung auf der vorherigen Seite.

Die ‚Programmierer*innen' versuchen nun ihren Code zu ändern und richtigzustellen. Sind sie damit fertig, tauschen sie wieder mit den ‚Computern' den Platz und diese lesen wieder dem ‚Roboter' die

nun vorliegenden Anweisungen vor. Sehen die Anweisungen wie unten abgebildet aus, so endet der Weg des Roboters beim Auto und die Anweisungen waren richtig. Die **Programmierer*innen** können nun kurz jubeln und dann werden die Rollen getauscht bzw. durchgemischt. Das heißt z.B. eine/r der ‚Programmierer*innen' wird zum ‚Roboter', die anderen zu ‚Computern' - und die vorherigen ‚Computer' und der ‚Roboter' werden nun zusammen zum ‚Programmierer*innen'-Team. Nun beginnt das Ganze von vorne: Es wird ein neues Ziel gewählt (z.B. der Baum) und die neuen **‚Programmierer*innen'** versuchen die Anweisungen dafür zu legen usw.



Beispiel einer Programmierung 2,
Grafik angelehnt an: CC-BY-SA Informatik-Werkstatt

Dies spielt man mehrere Runden und tauscht immer wieder die Rollen durch, sodass jeder (der möchte) einmal ‚Programmierer*in'/‚Roboter'/‚Computer' war.

Kleine Ergänzungsmöglichkeiten

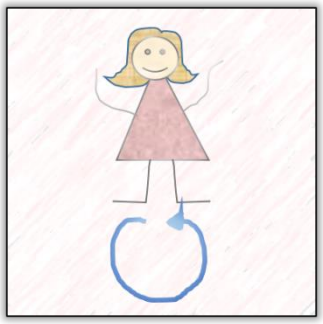
Das Spiel (egal ob in der Basisvariante oder einer erweiterten Variante) kann **auch im Freien gespielt** werden. Dazu kann man auch Isolierband verwenden, sofern der Untergrund dementsprechend ist. Ansonsten kann man das Labyrinth mit Kreide auf Asphalt aufzeichnen – oder mit Heringen (z.B. jenen für Zelte, Campingbedarf) und Wollschnüren in der Wiese aufgespannt werden.

Wird das Spiel mit einer **großen Teilnehmer*innen-zahl** gespielt, so empfiehlt es sich, zwei oder drei Programmierer*innen-Teams zu bilden sowie die Roboterzahl entsprechend zu erhöhen. Jedes Programmierer*innen-Team hat also einen eigenen Roboter (und ggf. auch Computer-gruppe). Ist die Teilnehmer*innen-zahl sehr groß, so wird empfohlen mehrere Labyrinthe zu verwenden.

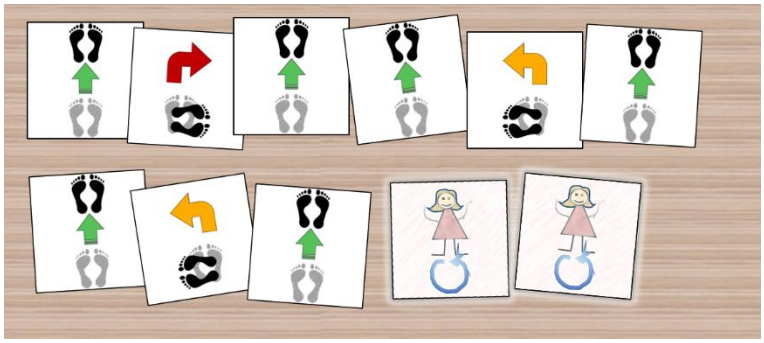
Empfehlenswert wäre es im Zuge des Spiels dessen **Bezug zur Informatik hervorzuheben** und mit den Teilnehmenden zu besprechen. Das beinhält u. a., dass Computer und generell Informatiksysteme von Menschen entwickelte Anweisungen abarbeiten. Dies geschieht auch beim Programmieren, indem man gewisse Anweisungen aneinanderreiht, diese sind aber eben komplizierter. Software und allgemein Programme sind immer Abfolgen von vordefinierten Anweisungen (unterschiedlich komplex natürlich), diese werden auf Informationstechnologie ausgeführt und steuern dort Vorgänge.

Erweiterung mit eigenen Anweisungen (Codierungen)

Das Spiel kann von den Kindern selbst erweitert werden, indem man leere Anweisungskärtchen ausdruckt (nicht laminieren, da sonst schwer beschreibbar). Diese findet man auf Seite 9 (und ggf. 10) in der Druckdatei. Die **Kinder können nun selbst eine eigene Anweisung kreieren**, wie z.B. ‚drehe dich einmal im Kreis' und **diese dann zeichnen**. So könnte dies beispielsweise aussehen:



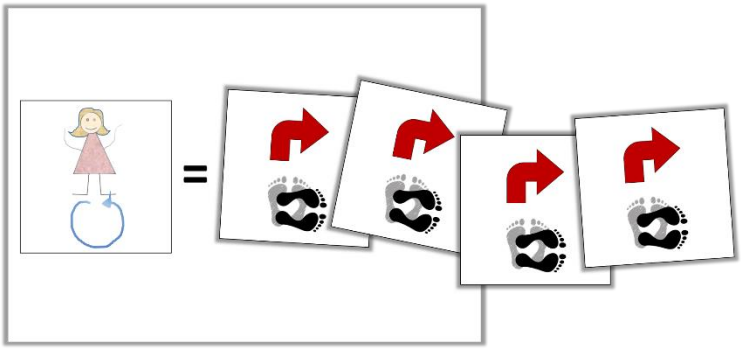
Beispiel einer selbst gezeichneten Anweisung



Beispiel einer Programmierung mit eigener Anweisung,
Grafik angelehnt an: CC-BY-SA Informatik-Werkstatt

Diese Anweisung könnte man dann verwenden, um das Ankommen des Roboters beim Ziel zu feiern. Der Roboter würde sich also, beim Ziel (bei vorherigem Beispiel: Baum) angekommen, zweimal im Kreis drehen (analog einem Siegestanz).

Man kann auch **Anweisungen erschaffen, die aus vielen kleinen Anweisungen bestehen** (nennt man in der Informatik Methoden). Zum Beispiel würde man dann das ‚Sich-einmal-im-Kreis-drehen' als vier Rechtsdrehungen beschreiben und dieses als eigene Anweisung definieren. Entweder **zeichnet man die Erklärung der Anweisung auf ein Blatt** (gibt es zum Ausdrucken bei den Kärtchen) oder legt dort die ‚Unteranweisungen' hin. Man legt die Beschreibung (Definition der Anweisung) in Sichtweite, z.B. auf einen nebenstehenden Tisch. Dort kann man dann nachsehen, was das Zeichen bedeutet, falls man es vergisst oder jemand anderes noch nicht weiß, was es bedeutet.



Beispiel: Beschreibung einer Anweisung durch andere Anweisungen

Solche Anweisungen und auch die schon vorhandenen (1 Schritt vorwärts usw.) kann man auch **unterschiedlich bezeichnen (codieren) lassen**. Die Kinder können sich ein Wort überlegen, welches die jeweilige Anweisung bezeichnet. Zum Beispiel, wie schon am Beginn erklärt das Wort „rot" für eine Rechtsdrehung. Das im-Kreis-Drehen könnte dann als ‚pink' bezeichnet werden. Es ist aber auch möglich ganz andere Wörter zu verwenden, wie z.B. ‚Prinzessin' oder ‚Banane'. Wichtig ist dabei nur, dass die Bezeichnung für alle Mitspielenden klar ist.

Anmerkung (Bezug zur Informatik): *In der Informatik gibt es oft Codierungen, zum Beispiel versteht der Computer nur Strom oder kein Strom, das heißt, dort wird alles durch 0 und 1 codiert. Zum Beispiel bedeutet ‚01001110' für den Computer den Buchstaben ‚K'. Es gibt aber noch viel mehr Codierungen in der Informatik, aber auch im echten Leben. Ein Beispiel dafür ist die Ampel: Jeder weiß, dass ‚Rot' Stopp bedeutet und ‚Grün' Gehen. Auch die Blindenschrift, die fühlbaren Punkte bei Liftknöpfen z.B., ist eine Codierung. Bei all diesen Beispielen ist eines immer gleich: Es gibt Symbole/Zeichenfolgen, die eine bestimmte Bedeutung haben und für etwas Anderes stehen. Man versteht die Codierung nur, wenn alle Beteiligten wissen oder nachschauen können, wofür die Zeichen stehen (also wie sie definiert sind).*

Anweisungen in einer Programmiersprache sind auch eine Art Codierung, jede Anweisung steht für etwas. Definiert man eine neue Anweisung, die aus mehreren schon bekannten Anweisungen besteht, so nennt man dies ‚Methode' oder ‚Funktion'. Man kann diese neue (größere) Anweisung dann an vielen unterschiedlichen Stellen im Code verwenden. Das ist besonders hilfreich, wenn man diese an mehreren Stellen braucht. Würde man keine neue Anweisung definieren, müsste man all die vielen ‚Unteranweisungen' an all diesen Stellen eigens hinschreiben. Man spart sich also durch solch zusammengebaute, neue Anweisungen Zeit. Zudem macht man weniger Fehler, wenn man etwas einmal definiert und sich gut durchdenkt, dann weiß man, dass der Code korrekt ist und kann die neue Anweisung einfach verwenden.