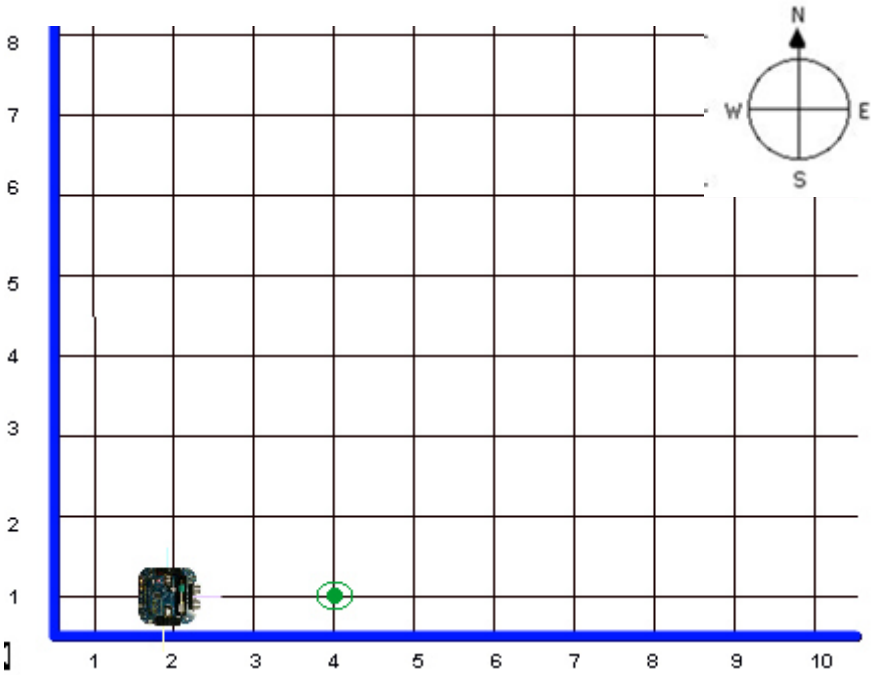
Bertl the Robot

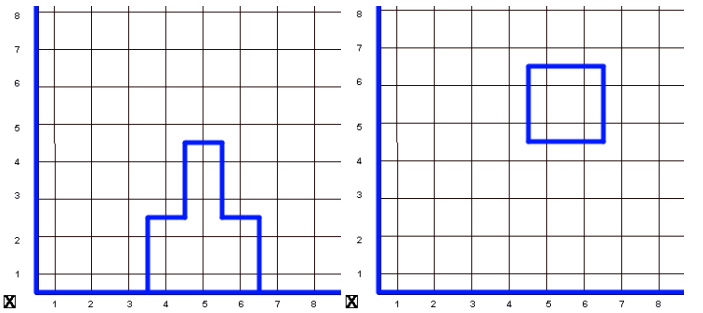
1 Die Roboterwelt (The Robot World)

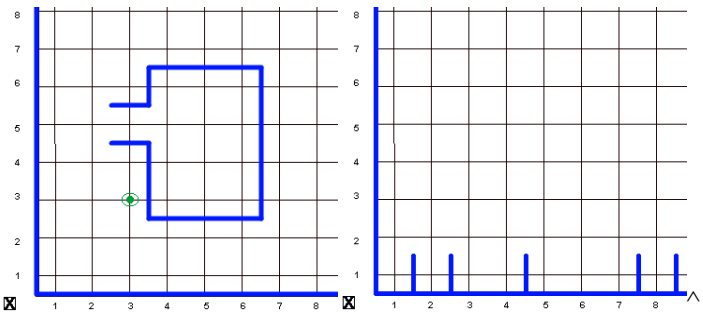


Der Roboter bewegt sich in einer trivialen Welt die aus horizontalen *Streets* (y) und vertikalen *Avenues* (x) besteht. Der Kreuzungspunkt von *Street* und *Avenue* wird Straßenecke (*corner*) genannt und sind durch die x und y Koordinate bestimmt (in der Abbildung y=1 und x=2). Die *Avenues* verlaufen in **South-North** Richtung und die *Streets* in **West-East** Richtung. In dieser Welt könnten noch weitere zwei Elemente vorhanden sein: Mauersegmente und Beepers, die schwarze Linien am Boden entsprechen).

1.1.1 Mauersegmente

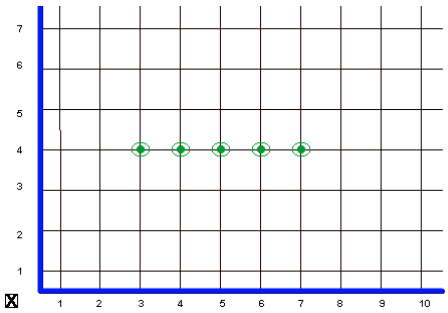
In dieser Welt können die in der Abbildung gezeigten Mauerelmente eingesetzt werden.





1.1.2 Beeper

Beeper entsprechen in etwa den Pagern oder läutenden Handys. Sie geben einen leisen Ton von sich, sodass ein Roboter sie findet, sofern der Roboter sich an dieser Straßenecke befindet, wo ein Beeper liegt.



1.2 Robotereigenschaften

Der Roboter besitzt folgende Dinge:

1. Eine TV Kamera mit der er eine Wand ab einer Entfernung von ½ Block vor sich erkennt.
2. Ein Mikrofon um einen Beeper zu hören.
3. Einen internen Kompass, sodass er weiß in welche Richtung er schaut.
4. Eine mechanische Hand um einen Beeper aufzuheben oder abzulegen.
5. Einen Rucksack (*beeper-bag*) für den Transport der Beeper.

Schließlich kann der Roboter sich auch noch abschalten.

.

1.3 Aufgaben (task) und Stellungen (situation)

Eine Aufgabe (**task)** ist etwas was der Roboter tun soll:

1. Bewege dich (*Move*) zur Straßenecke 5th *Street* & 8th *Avenue*.
2. Auflösen eines Labyrinths
3. Entfliehen aus einem Raum mit Tür
4. Finden eines Beepers und ablegen an einer anderen Stelle
5. Usw.

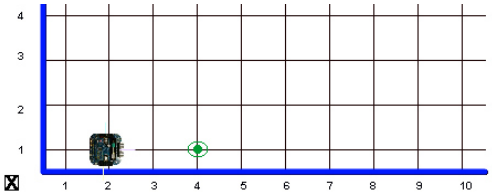
Eine **Stellung** ist eine Beschreibung wie eine Welt aussieht. Neben der grundsätzlichen Struktur der Welt, den Mauersegmenten und Beepers müssen noch folgende Fragen beantwortet werden:

1. Wo ist die momentane Position des Roboters; Sowohl die genaue Lage (an welcher Straßenecke) als auch die Ausrichtung (North, South, ...)?
2. Wo und wie lange ist ein Mauersegment?
3. Wo befinden sich die Beeper in der Welt? Diese Frage beinhaltet auch die Anzahl der Beeper im Rucksack.

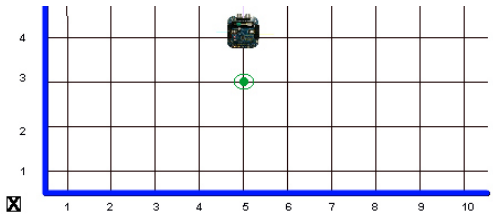
Die Stellungen werden hier mit einer kleinen Landschaft dargestellt. Unterschieden wird noch:

1. die Ausgangsstellung (*initial situation*), die beschreibt wo am Beginn sich jeder Roboter befindet;
2. die Endstellung (*final situation*), ist jener Platz an dem der Roboter sich befindet, wenn er fertig ist.

Anschließend ist noch eine Ausgangsstellung und Endstellung beispielhaft gezeigt.



Ausgangsstellung (*initial situation*)



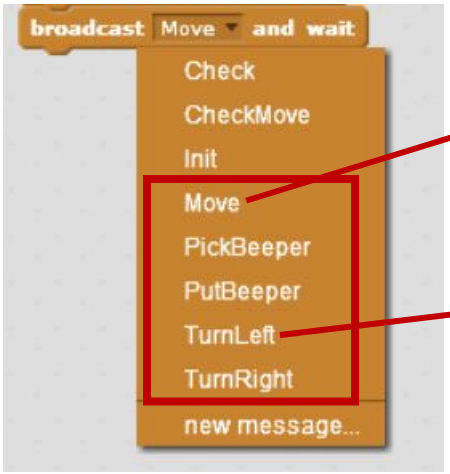
Endstellung (*final situation*)

2 Elementarbefehle (*primitive instruction*)

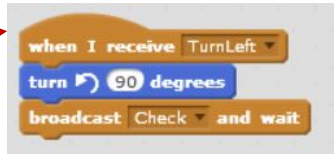
Wir beginnen nun mit dem Studium der Roboter Programmiersprache und fangen mit den Elementarbefehlen (*primitive instruction*) an, die in der Grundausstattung des Roboters von der Fabrik eingebaut sind:

1. **Move**: Ändern der Position; Der Roboter bewegt sich einen Block vorwärts.
2. **PickBeeper**: Aufheben eines Beepers und ablegen im Rucksack.
3. **PutBeeper**: Einen Beeper aus dem Rucksack nehmen und an die Straßenecke legen (virtuell :-).
4. **TurnLeft**: Drehen im Stehen; Der Roboter dreht sich ¼ Drehung (-90°) nach links.
5. **TurnRight**: Drehen im Stehen; Der Roboter dreht sich ¼ Drehung (90°) nach rechts.

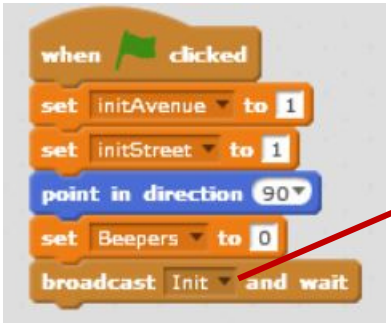
Aus diesen Elementarbefehlen lassen sich weitere kompliziertere Befehle bilden. Andererseits lassen sich primitive Befehle nicht in weitere Befehle teilen. Muss TurnRight auch ein Elementarbefehl sein?
In Scratch werden diese Befehle via Broadcasts (sowie Fernsehen oder Radio) übertragen:



z.B. Move und TurnLeft (siehe Scratch Programm Scripts Bertl):



Für die Ausgangssituation muss der Roboter noch an eine bestimmte Stelle und Ausrichtung in die virtuelle Welt gesetzt werden. Das geschieht nach dem Start der Ausführung mit der grünen Flagge  :

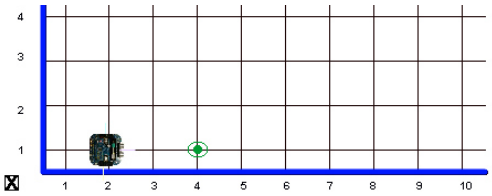


Mit den Sprites Beeper und Checker haben wir vorerst nichts zu tun.

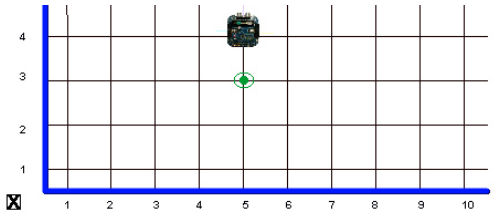
2.1 Das erste Programm: bertl-first-2-3 i.sb2

Das folgende Projekt FIRST löst folgenden Task:
Bertl soll sich von der Ausgangssituation y=1 und x=2 um zwei Avenues nach rechts bewegen; einen Beeper aufheben; eine weitere Avenue nach rechts; sich nach links drehen; um zwei Streets nach oben bewegen; den Beeper ablegen und auf die Position (5,4) (Endstellung) sich bewegen. Die *initial situation* ist in Programm der Scratch-Datei bertl-first-2-3 i.sb2.

2.1.1.1



Ausgangsstellung (*initial situation*)



Endstellung (*final situation*)

2.1.1.2

2.2 Programmausführung

Bevor das Programm ausgeführt werden kann, muß es zuerst von der Fabrik gelesen und übersetzt (vom Compiler) werden, um sicherzustellen, dass es keine Fehler aufweist. Nachdem das Programm erfolgreich übersetzt wurde kann es auf den Bertl geladen werden. Die Programmausführung kann nachdem der Pilot (Schüler, Lehrer) den Roboter an die angegebene Position (Ausgangssituation) gestellt hat beginnen. Mit der grünen Flagge  wird der Bertl-Roboter gestartet.

2.3 Abschaltung im Fehlerfall

Ist ein Roboter nicht in der Lage einen Befehl vollständig auszuführen, so schaltet er sich selbst ab (Fehlerabschaltung, *error shutoff*). Die nachfolgend behandelten Befehle könnten einen Fehler verursachen, sodass die folgenden Bedingungen erfüllt sein müssen:

1. Ein Roboter darf einen **Move**-Befehl nur ausführen, wenn der Weg vor ihm bis zur nächsten Straßenecke frei ist.
2. Ein Roboter darf einen **PickBeeper**-Befehl nur ausführen, wenn an derselben Straßenecke zumindest ein Beeper liegt.
3. Ein Roboter darf einen **PutBeeper**-Befehl nur ausführen, wenn der Rucksack auch einen Beeper beinhaltet.

Diese Bedingungen können aber nur garantiert werden, wenn vor dem Schreiben des Programms die **Ausgangslage**, also den Anfangsort und –zustand des Roboters bekannt ist.

3 Programmierfehler

In diesem Abschnitt werden mögliche Programmierfehler behandelt. Warum müssen wir uns mit Fehlern beschäftigen? Die Anwort besteht darin, daß das Programmieren eine ungewöhnlich hohe Anforderung an die präzise (genaue) Arbeitsweise des Programmierers stellt, mit der Aufgaben durch Software gelöst werden sollen. Prinzipiell sollten Fehler natürlich nicht auftreten; Praktisch aber treten Fehler „exzessiv“ häufig auf. Es sollten die Fehler sehr schnell gefunden und behoben werden. Dafür soll die folgende Klassifikation aller Programmierfehler in vier Kategorien dienen.

3.1 Lexikalische Fehler (lexical error)

Ist im Programm ein Wort enthalten, welches nicht zum Wortschatz des Roboters gehört, so spricht man von lexikalischen Fehlern. (später)

3.2 Syntaxfehler (syntax error)

Es sind zwar alle im Programm vorkommenden Worte im Wortschatz des Computers vorhanden, allerdings sind sie falsch angeordnet. Im Programm wird also eine falsche Grammatik verwendet, oder sie entspricht nicht der Grammatik des Roboter. (später)

Beide obigen Fehler werden vom Compiler erkannt und angezeigt, sind also Compiler Fehler. Die Top 100 Compiler Fehler sind auf https://developer.mbed.org/cookbook/Top-Compiler-Errors. Da Scratch Blockprogrammierung ist können diese Fehler nicht auftreten.

3.3 Ausführungsfehler (execution error)

Im Gegensatz zu den beiden vorigen Fehlern, die in der Fabrik (vom Compiler) erkannt werden, können Ausführungsfehler vom Piloten nur während das Programm läuft gefunden werden. Ausführungsfehler treten immer dann auf, wenn ein Programm einen Befehl nicht erfolgreich ausführen kann. Ein Ausstieg aus dem Programm mit Fehler ist die Folge (*error shutoff*).
z.B.:
Move()-Befehl, wenn eine Mauer davor steht.
PickBeeper()-Befehl, wenn kein Beeper an der Straßenecke liegt.
PutBeeper()-Befehl, wenn sich kein Beeper im Rucksack befindet.

3.4 Logische Fehler (logical or intent error)

Ein logischer Fehler tritt auf, wenn das Programm erfolgreich beendet wird, allerdings die Aufgabenstellung nicht vollständig erfüllt wurde. Diese Fehler sind am schwierigsten zu finden und verlangen daher ein sorgfältiges Design des Algorithmus.

In anderen Programmiersprachen wie C, C++, o.ä. müssen die einzelnen erzeugten Module jedoch mit dem Linker zu einer ausführbaren BIN- oder EXE-Datei gebunden werden.

3.5 Linker-Fehler

Wenn nach einer Übersetzung eine Fehlermeldung des Linkers erscheint, die meist >>unresolved external<< enthält, dann fordert im Allgemeinen das Programm eine Bibliothek an, die der Linker nicht hinzugebunden hat. Für solche Fehler kann es verschiedene Ursachen geben:

- hat es beim Compilieren bereits eine Warnung gegeben, dann könnte der Funktions-/Methodenname falsch geschrieben sein
- der `#include`-Befehl fehlt für diese Methode/Klasse.
- die Bibliothek (Library) wurde nicht eingebunden:
    - bei einer IDE muss die Bibliothek, sofern sie keine Standardbibliothek ist, in den Projekt-Einstellungen oder in der Projektmappe bzw. im Program Workspace eingetragen sein.
    - die Bibliothek liegt in einem Pfad, der vom Linker nicht nach Bibliotheken durchsucht wird.

linker errors, e.g.: "Undefined symbol pstorage_init"