

# Programmieren mit Squeak-BotsInc

Rita Freudenberg

2. März 2007

**Die BotsInc-Umgebung** BotsInc ist eine in Squeak geschriebene Programmierumgebung, mit der Roboter programmiert werden, die dann graphische Spuren auf dem Bildschirm hinterlassen. Die zugrundeliegende Idee geht auf Seymour Papert und Logo zurück. In seinem Buch, auf das sich dieser Workshop stützt, beschreibt Prof. Stephane Ducasse die Einführung elementarer Programmierkonzepte (wie Schleifen, Bedingungen, Abstraktion) unter Verwendung der von ihm entworfenen Bots-Inc-Umgebung. In BotsInc wird keine spezielle Skriptsprache benutzt, es wird direkt in Squeak programmiert. Die grundlegenden Elemente einer Squeak (Smalltalk)-Entwicklungsumgebung stehen zur Verfügung, allerdings in deutlich reduzierter Form. So sind im Klassenbrowser nur die zu BotsInc gehörenden Klassen aufgelistet und auch die Kontextmenüs enthalten weniger Einträge (wie im direkten Vergleich der beiden folgenden Abbildungen zu sehen ist). Dadurch wird die Syntax und Arbeitsweise erlernt, die auch für die Programmierung mit Squeak bzw. Smalltalk erforderlich ist.

BotsInc ist also ein Einstieg in die objektorientierte Programmierung und in Smalltalk. Sehr schnell sind erste Programme geschrieben. Es gibt wenige, einfache, einheitliche Syntaxregeln, das Hauptaugenmerk liegt auf der Umsetzung der Konzepte.

**Installation** Um BotsInc zu installieren benötigt man Squeak und ein spezielles Image, welches die BotsInc-Klassen enthält. Am Einfachsten ist es, sich ein fertiges Verzeichnis mit allen benötigten Dateien aus dem Internet herunter zu laden, es ist kostenlos verfügbar. Diese Datei (ReadyMac bzw. ReadyPC) gibt es für Windows und Mac unter

<http://smallwiki.unibe.ch/botsinc/download/>

oder auf der Seite des Apress-Verlages, wenn man im Download-Bereich das Buch „Squeak - Learn Programming with Robots“ auswählt. Da Squeak selbst auf weiteren Plattformen läuft kann man die .image-Datei aus dem heruntergeladenen Verzeichnis zusammen mit einer normalen Squeak-Installation (SqueakV3.sources) z.B. auch unter Linux nutzen.

**Erste Schritte** Nach dem Start von BotsInc öffnet sich ein Fenster mit einer Roboter-Fabrik, einem Roboter, einer geschlossenen Klappe und einer geöffneten Klappe, die einen Workspace enthält (ähnlich Abbildung 1). Der dargestellte Roboter ist in Draufsicht dargestellt und zeigt seine augenblickliche Richtung an. Ihn zu programmieren heißt, ihm Nachrichten zu schicken, die der Roboter dann ausführt.

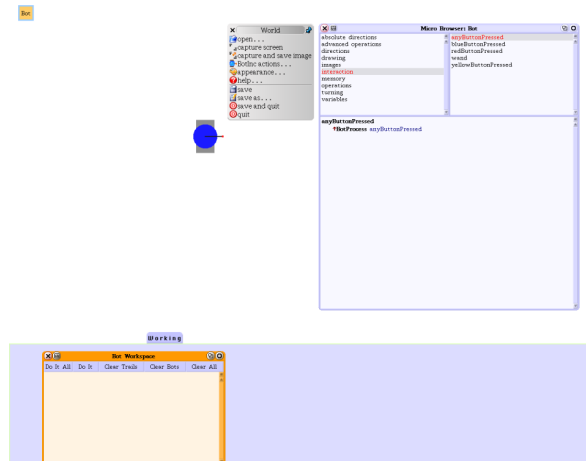


Abbildung 1: BotsInc-Umgebung mit Workspace, Microbrowser und Weltmenü

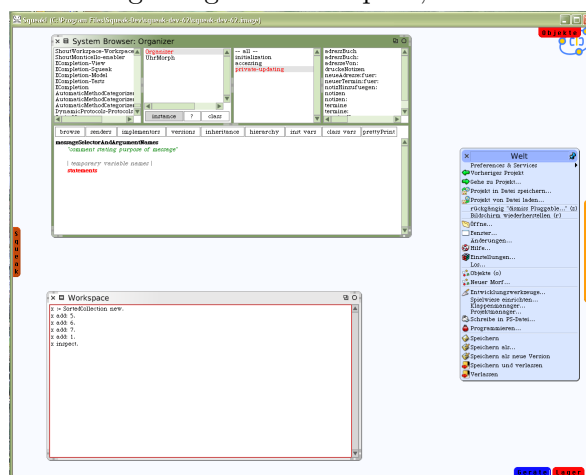


Abbildung 2: Squeak Entwicklungsumgebung mit Workspace, Klassenbrowser und Weltmenü

Bewegt man den Mauszeiger auf den Roboter und wartet einen Moment, erscheint ein Hilfeballon mit einigen Informationen über den Roboter.

Um dem Roboter Nachrichten zu schicken, klickt man ihn mit der linken Maustaste an. In den erscheinenden Nachrichten-Ballon können nun Nachrichten getippt werden, durch Betätigen der Enter-Taste werden diese an den Roboter geschickt, der sie dann ausführt.

Eine einfache Nachricht wäre *go: 200* gefolgt von ENTER, die bewirkt, dass der Roboter sich 200 Pixel in seine augenblickliche Richtung bewegt.

Die Nachricht *turnLeft: 20 + 70* bewirkt eine Drehung (in Uhrzeigerrichtung) um  $20 + 70 = 90$  Grad. Diese zweite Nachricht ist eine sogenannte zusammengesetzte Nachricht, da der Wert, der angibt, um wie viel Grad sich der Roboter drehen soll, wieder eine Nachricht ist ( $20 + 70$ ). Die Farbe eines Roboters kann man mit der Nachricht *color: Color green* zuweisen, was ebenfalls eine zusammengesetzte Nachricht ist. Einen neue Roboter kann man erzeugen,

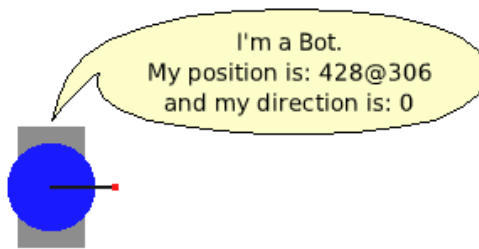


Abbildung 3: Roboter mit Informationen

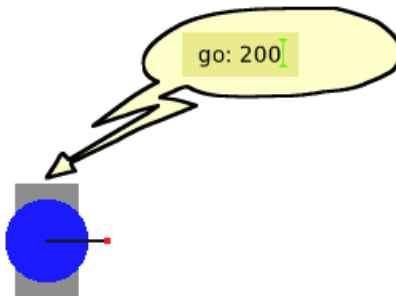


Abbildung 4: Nachricht für den Roboter

indem man an die Roboter-Fabrik die Nachricht *new* schickt. Dieser zeigt standardmäßig nach rechts (east).

In Smalltalk unterscheidet man drei Arten von Nachrichten, die in BotsInc ebenso verwendet werden, nur dass der Empfänger der Nachricht jeweils nicht explizit aufgeschrieben wird, da das der Roboter ist:

- unäre Nachrichten, bestehend aus Empfänger und Methode(*new*)
- binäre Nachrichten, bestehend aus Empfänger, Methode und Parameterobjekt(*20 + 70*) und
- Schlüsselwort-Nachrichten, bestehend aus Empfänger, Methode mit `:` und Parameterobjekt(en)(*go: 200*)

Das Beenden und Speichern erfolgt über die entsprechenden Einträge im Weltmenü, das man durch Klick der linken Maustaste auf den Hintergrund erhält. Wenn man speichert, wird das gesamte System (das Image) neu gespeichert, so dass beim nächsten Start die Oberfläche exakt so aussieht wie zum Zeitpunkt des Speicherns.

**Skripte und Kontrollstrukturen** Wenn man programmiert möchte man mehrere Nachrichten an den Roboter schicken. Dazu kan man entweder jeweils die Nachricht schreiben, die Enter-Taste drücken, die nächste Nachricht schreiben usw. Hilfreich ist dabei, dass man, wenn der Nachrichtenballon erscheint, mit den Pfeiltasten vorherige Nachrichten wiederholen kann und nicht komplett neu schreiben muss. Dennoch ist dies sehr mühselig. Weitere Erleichterung schafft die Möglichkeit, mehrere Nachrichten gleichzeitig als „Kaskade“ zu verschicken. Dazu schreibt man einfach alle Nachrichten hintereinander in den Ballon und trennt diese durch ein Semikolon. Auf diese Art kann man generell in Squeak mehrere Nachrichten an den selben Empfänger schicken.

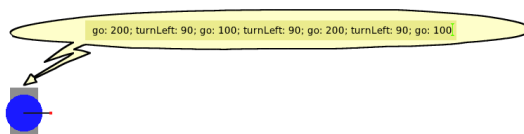


Abbildung 5: Nachrichtenkaskade

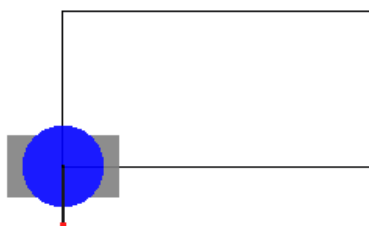


Abbildung 6: Ergebnis der Nachrichtenkaskade

Für komplexere Programme ist das aber immer noch nicht befriedigend. Nicht nur wird der Platz im Nachrichtenballon mit der Zeit knapp, man möchte auch die Programmteile, die man einmal geschrieben hat, abspeichern und später wieder ausführen können.

**Skripte** Dafür gibt es Skripte. Skripte werden im Workspace geschrieben, der nichts anderes als ein Texteditor ist. Der Hauptunterschied zwischen Nachrichten im Nachrichtenballon und einem Skript im Workspace ist die Tatsache, dass im Workspace der Empfänger der Nachricht angegeben werden muss. Im Beispiel ist das *pica*, ein Roboter, der zuerst mit *new* erzeugt wird. Es ist wichtig, dass *pica* mit einem kleinen Buchstaben beginnt, weil nur Klassen mit Großbuchstaben beginnen (wie *Bot*). Befehle werden durch Punkte getrennt, beim letzten Befehl ist das nicht mehr nötig, da kein weiterer Befehl folgt.

Im Workspace befinden sich auch die Buttons für das Ausführen eines Skriptes. **Do It All** führt alle Nachrichten im workspace aus. **Do It** führt nur die markierten Nachrichten aus.

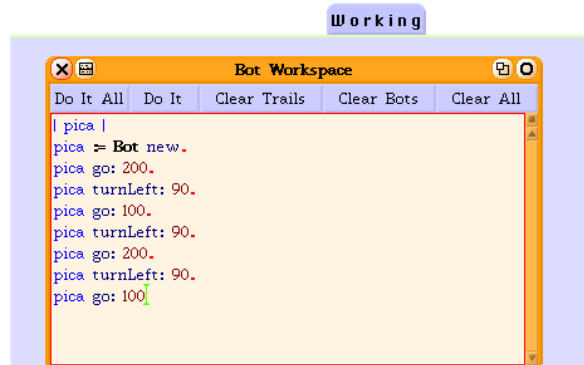


Abbildung 7: Nachrichtenkaskade nun als Skript im Workspace

**Clear Trails** löscht alle Spuren, die die Roboter auf dem Bildschirm hinterlassen haben. **Clear Bots** löscht alle Roboter vom Bildschirm. **Clear All** schließlich löscht sowohl Spuren als auch Roboter.

Wenn man Text in den Workspace tippt, wird dieser gleich auf syntaktische Korrektheit geprüft. Ein rotes Wort weist auf einen Fehler hin, beispielsweise eine falsch geschriebene Nachricht. Versucht man trotzdem, das Skript auszuführen, öffnet Squeak ein Fenster mit einem Hinweis auf den Fehler und der Aufforderung, aus einer angebotenen Liste die Nachricht auszuwählen, die man eigentlich gemeint hat.

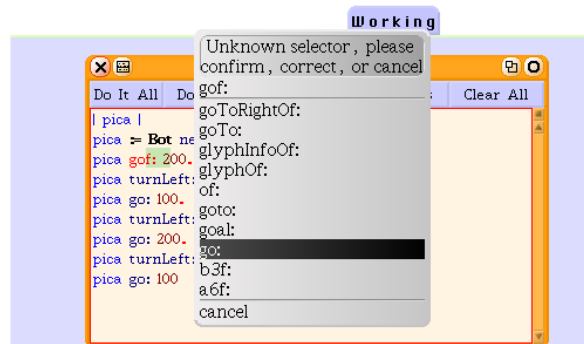


Abbildung 8: Fehler im Skript Zeile 3, Auswahlliste für die Ersetzung

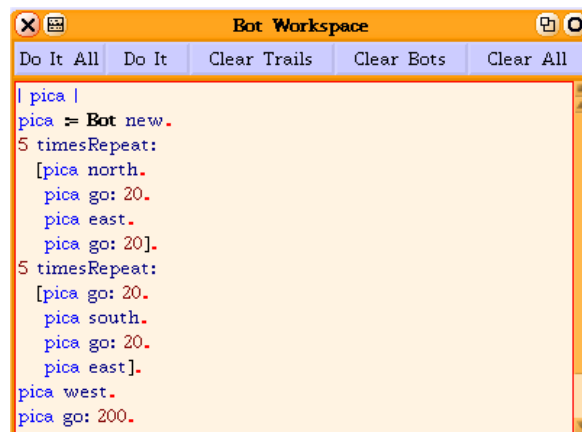
**Schleifen** Im Buch von S. Ducasse werden erst einige weitere nützliche Grundlagen und Nachrichten beschrieben, was für den Einsatz in Schulen unbedingt wichtig ist. Da hier ein möglichst umfassender Überblick gegeben werden soll, überspringe ich einige Kapitel. Auch mit den bisher bekannten Dingen lassen sich Schleifen in BotsInc motivieren und verwenden.

Das Schreiben eines Skriptes zum Zeichnen regelmäßiger geometrischer Formen wie z.B.

eines Sternes erfordert schon etliche Zeilen, und man muss aufpassen, dass man nicht den Überblick verliert, da sie sich fast nur durch die entsprechenden Zahlenwerte unterscheiden. Durch solche Aufgaben (bzw. Wünsche der Schüler, die ja meistens selber auf die Idee kommen, immer komplexere Formen zu zeichnen) lassen sich Schleifen gut einführen. Eine Schleife ist einfach eine spezielle Nachricht, die ein Roboter versteht. Die Syntax einer Zählschleife lautet folgendermaßen:

```
n timesRepeat: [ Sequenz von Ausdrücken ]
```

Für n wird die Anzahl der gewünschten Wiederholungen angegeben, dann folgen alle zu wiederholenden Nachrichten in eckigen Klammern. Nachrichten in eckigen Klammern werden Block genannt. Für bessere Lesbarkeit wird empfohlen, alle Nachrichten im Block etwas einzurücken, damit man sofort erfassen kann, welche Nachrichten zum Block gehören. Ein Beispiel für ein Skript mit einer Zählschleife:



```
Bot Workspace
Do It All Do It Clear Trails Clear Bots Clear All
| pica |
pica := Bot new.
5 timesRepeat:
  [pica north.
  pica go: 20.
  pica east.
  pica go: 20].
5 timesRepeat:
  [pica go: 20.
  pica south.
  pica go: 20.
  pica east].
pica west.
pica go: 200.
```

Abbildung 9: Skript mit Zählschleife

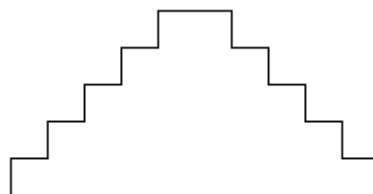


Abbildung 10: Ausgabe Skript von Bild 9

Neben Zählschleifen gibt es natürlich auch bedingungsabhängige Schleifen. Um sinnvolle Bedingungen formulieren zu können fehlen an dieser Stelle noch die Kenntnisse über Variablen und ein Überblick über verfügbare Eigenschaften der Roboter, die abgefragt werden könnten.

**Variablen** Die mit den Skripten gezeichneten Bilder haben eine feste Größe. Wenn man z.B. die Pyramide von Bild 10 verkleinert zeichnen möchte, muss man alle Zahlenwerte im Skript per Hand ändern und dabei noch beachten, dass die Änderungen maßstabsgetreu sind. Es ist nicht schwer zu zeigen, dass solche Änderungen fehleranfällig und mühselig sind. Variablen sind an dieser Stelle die Rettung. Mit Variablen können wir

- die Höhe der Stufen und die Gesamtbreite einmal für das Skript definieren
- bei Bedarf auf diese Werte verweisen
- diese Werte bei Bedarf ändern

Variablen sind *Namen*, die mit einem *Wert* assoziiert werden. Sie müssen *deklariert* und es muss ihnen ein *Wert zugewiesen* werden. Dann können wir auf diese Variable *verweisen* und auf ihren *Wert* zugreifen. Es ist ebenfalls möglich, den Wert zu *verändern*. Der Wert einer Variable kann eine Zahl, eine Menge von Objekten oder auch ein Roboter sein.

Eine Variable in einem Skript wird folgendermaßen deklariert:

```
| hoehe breite |
```

Genaugenommen werden durch die vertikalen Striche temporäre Variablen erzeugt, also Variablen, die nur während der Abarbeitung des Skriptes existieren. Die Wertzuweisung erfolgt mit

```
hoehe := 20.  
breite := 200.
```

Erfolgt die Wertzuweisung, bevor die Variable zum ersten Mal verwendet wird, nennt man das *Initialisieren*. Es können nicht nur Werte zugewiesen werden, sondern auch Ausdrücke, deren Ergebnis dann zugewiesen wird, z.B.

```
hoehe := 10 + 10.
```

Die Namen dieser Variablen werden dann in den Nachrichten **anstelle** der Zahlenwerte verwendet. Im Pyramidenbeispiel könnte man neben der Größe der Stufen auch deren Anzahl durch eine Variable festlegen. Durch die Verknüpfung von Schleifen und Variablen können Labyrinth und Spiralen gezeichnet werden.

**Skripte und Methoden** Die nächste Abstraktionsstufe besteht darin, die Nachrichtenfolgen eines Skriptes als Methode zu verwenden. Das heißt, man kann ein Skript dann jedem beliebigen Roboter schicken und in einem Skript ein anderes aufrufen. Dazu muss man eine Folge von Nachrichten als Methode definieren.

Zum Erstellen eigener Methoden in Squeak braucht man einen Klassenbrowser. Dieser enthält **alle** im Image vorhandenen Klassen und deren Methoden. In BotsInc gibt es den Microbrowser, der nur eine Klasse enthält, nämlich den Roboter und nur für Roboter können Methoden geschrieben werden.

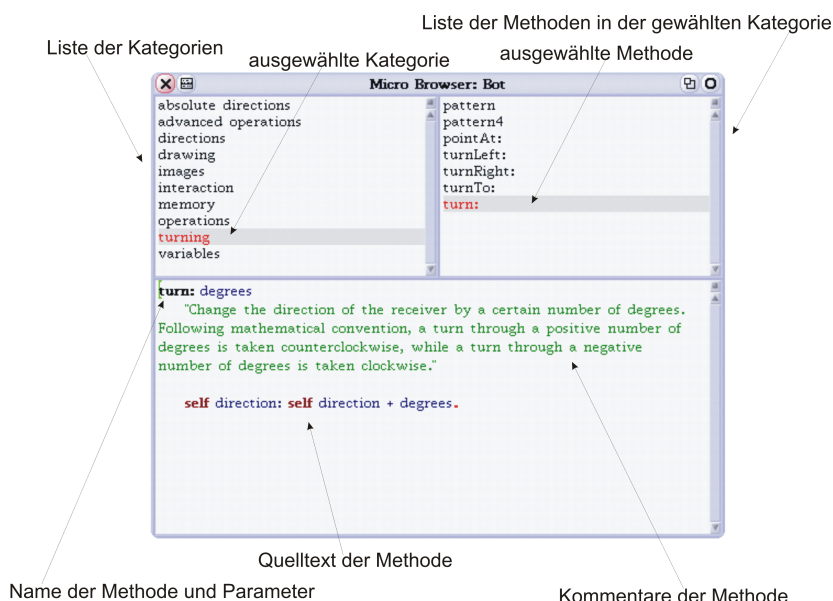


Abbildung 11: Der Microbrowser

Jede Methode wird in eine Kategorie eingeordnet. Das dient der Übersichtlichkeit, man könnte auch alle Methoden einer Klasse in eine einzige Kategorie schreiben, aber dann muss man nur unnötig lange nach einer bestimmten Methode suchen. Eine neue Methode kann entweder in eine existierende Kategorie einsortiert werden oder man erstellt eine neue Kategorie. Dazu gibt es verschiedene Kontextmenüs (unter Windows rechte Maustaste), die sich unterscheiden, je nachdem, in welchen Bereich des Browsers man klickt. Den Quelltext der Methode schreibt man dann in den unteren Bereich des Microbrowsers, dieser ist nämlich einfach ein Workspace (vgl. Abb. 13). Man kann einfach den dort stehenden Text überschreiben, sobald man einen neuen Methodennamen geschrieben hat und die Methode kompilieren will, wird sie unter dem neuen Namen abgelegt (Abb. 14). Sie wird kompiliert, indem man im Kontextmenü „accept“ auswählt.

In der Methode „quadrat“ steht zuerst der Methodenname, das ist als der name der an-richt, die der Roboter jetzt neuerdings versteht. Da hinter dem namen kein Doppelpunkt und weitere Parameter folgen, benötigt diese Nachricht auch beim Aufruf keine Parameter. Darunter steht in Anführungszeichen ein Kommentar, der beschreibt, was die Methode tut.



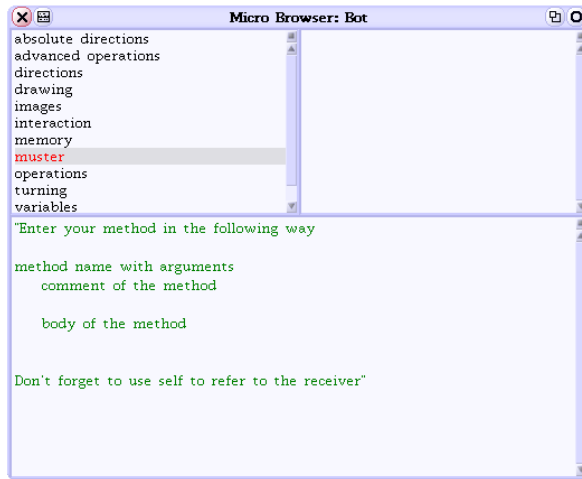


Abbildung 12: Nach dem Einfügen einer neuen Kategorie

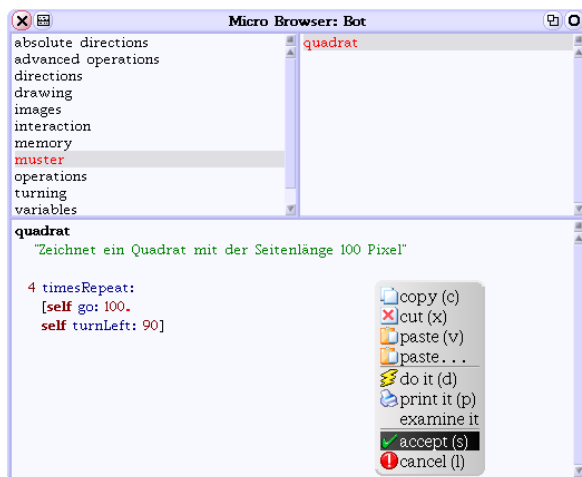


Abbildung 13: Die Methode „quadrat“ wurde geschrieben und kompiliert

Schließlich folgt der Rumpf der Methode, in diesem Beispiel eine Zählschleife, die den in eckigen Klammern stehenden Block viermal ausführt. Im Workspace (oder auch in einer anderen Methode) kann nun die neue Nachricht aufgerufen werden.

```
| pica |  
pica := Bot new.  
pica color: Color red.  
pica quadrat
```

Will man in einer Methode Parameter übergeben, muss man diese bei der Methodendefinition angeben (vgl. Abb. 12). Die Methode *turn*: erwartet einen Parameter, der in der Methode wie eine lokale Variable namens *degrees* verwendet wird. Eine Typangabe ist nicht notwendig, da alles in Squeak ein Objekt ist. Aber natürlich muss ich wissen, was die Methode *turn* macht, damit ich einen sinnvollen Wert als Parameter übergebe. Glücklicherweise kann ich mir ja den Quelltext der Methode ansehen, falls ich mir nicht sicher bin. Da dieser auch einen Kommentar enthält, weiß ich nun, dass die Methode die Richtung des Empfängers dieser Nachricht um den Wert der Variable *degrees* ändert, und zwar entgegen der Uhrzeigerichtung für positive Zahlenwerte und in Uhrzeigerichtung für negative Werte. Auch ohne den Kommentar kann ich das herausbekommen, wenn ich mir den Methodenrumpf anschau.

```
self direction: self direction + degrees.
```

Es wird die Nachricht *direction*: an *self* geschickt, also an den Empfänger der Nachricht. Diese Nachricht enthält einen Parameter, nämlich *self direction + degrees*. Der Parameter enthält ebenfalls eine Nachricht, nämlich *direction*. **Das ist nicht die gleiche Nachricht wie *direction*!** Im zweiten Fall wird nämlich kein Wert übergeben. Es wird der augenblickliche Wert von *direction* ermittelt und diesem wird die Nachricht *+* geschickt mit dem Parameter *degrees*. Es wird dann also der übergebene Wert zum aktuellen Wert von Richtung addiert.

Um herauszufinden, welche Nachrichten die Roboter verstehen können, kann man sich also auch die Liste der Kategorien und dann alle Methoden in den Kategorien ansehen. Die Bezeichner in *BotsInc* sind englisch, das müsste aber nicht so sein. Man kann ja beliebige Namen für Methoden vergeben und könnte so auch das gesamte System ins Deutsche übersetzen, sodass die Programmierung noch verständlicher wäre. Es gibt insgesamt nur sehr wenige Schlüsselworte in Squeak, die feste Bezeichner haben. Die zugrundeliegende Sprache heißt nicht zufällig *Smalltalk*.