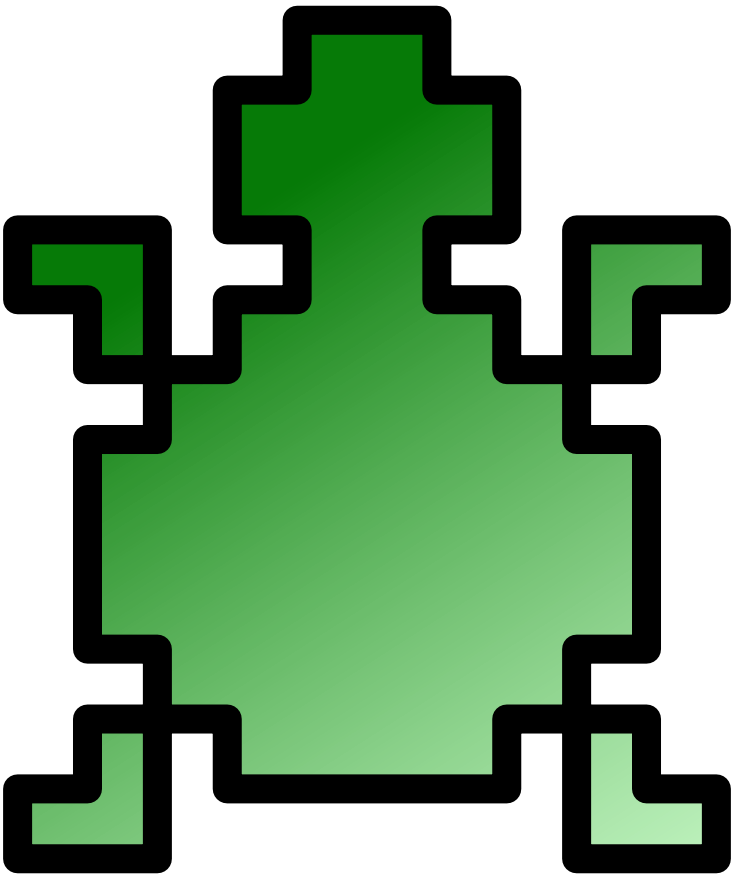
Das Handbuch zu KTurtle

Cies Breijs
Anne-Marie Mahfouf
Mauricio Piacentini
Übersetzung: Burkhard Lück



Das Handbuch zu KTurtle

Inhaltsverzeichnis

Tabellenverzeichnis

Zusammenfassung

KTurtle ist eine Programmierumgebung mit dem Ziel, das Programmieren möglichst einfach zu lernen. Um das zu erreichen, sind in KTurtle alle Programmwerkzeuge in der graphischen Benutzeroberfläche zu erreichen. Als Programmiersprache wird TurtleScript verwendet, in dieser Sprache können die Befehle übersetzt werden.

Kapitel 1

Einleitung

KTurtle ist eine Programmierumgebung für den Unterricht mit TurtleScript, einer Programmiersprache auf der Grundlage von Logo. Das Ziel ist von KTurtle ist es, die Programmierung möglichst einfach und leicht zugänglich zu machen. Daher ist KTurtle besonders geeignet, um Kindern die Grundlagen vom Mathematik, Geometrie und Programmierung zu lehren. Die wichtigste Eigenschaft von TurtleScript ist die Möglichkeit, die Befehle in die Muttersprache des Programmierers zu übersetzen.

KTurtle ist nach der „Schildkröte" benannt, die in der Programmierumgebung die Hauptrolle spielt. Der Benutzer programmiert die Schildkröte mit den TurtleScript-Befehlen, um eine Zeichnung auf der Zeichenfläche zu erstellen.

1.1 Was ist TurtleScript?

TurtleScript, die in KTurtle benutzte Programmiersprache, ist Konzepten der Programmiersprache Logo beeinflusst. Die erste Version von Logo wurde 1967 von Seymour Papert am MIT Artificial Intelligence Laboratory aus der Programmiersprache Lisp entwickelt. Seit dieser Zeit sind viele Versionen von Logo veröffentlicht worden. Im Jahr 1980 wurde Logo bekannt durch Versionen für MSX, Commodore, Atari, Apple II und den IBM PC Computer. Diese Versionen waren für den Unterricht gedacht. Am MIT finden Sie immer noch eine Webseite für Logo unter https://el.media.mit.edu/logo-foundation/ mit einer Liste von mehreren populären Implementationen dieser Sprache.

TurtleScript hat wie viele Implementierungen der Sprache Logo eine besondere Eigenschaft: Die Möglichkeit, die Befehle in die Muttersprache der Benutzer zu übersetzen. Diese Eigenschaft erleichtert Benutzern mit geringen oder keinen Englischkenntnissen den Einstieg. Außerdem hat KTurtle viele weitere Eigenschaften, die das Lernen des Programmierens für die Benutzer erleichtern.

1.2 Eigenschaften von KTurtle

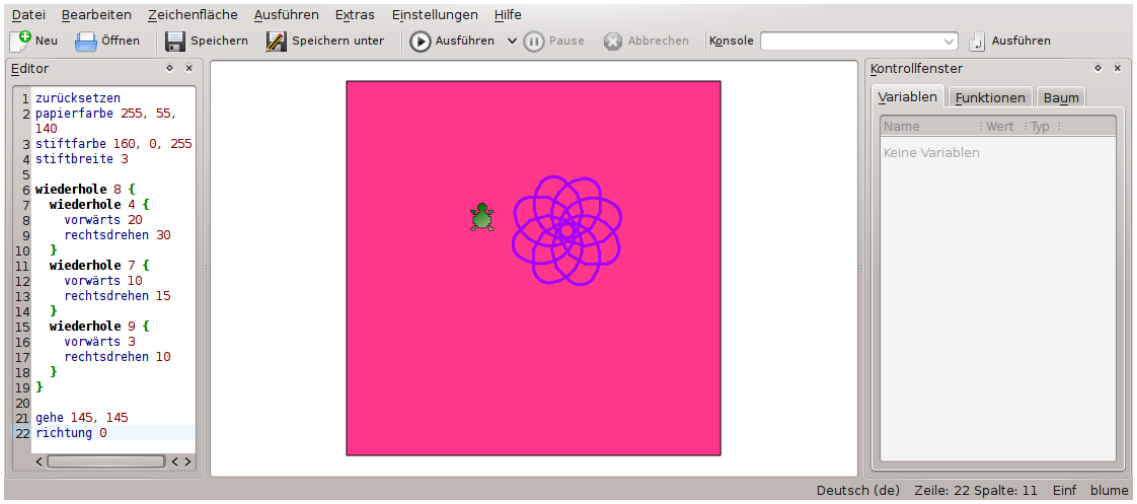
KTurtle hat einige besondere Eigenschaften, die den Einstieg in die Programmierung sehr einfach machen, hervorzuheben sind:

- Eine integrierte Entwicklungsumgebung mit dem TurtleScript-Interpreter, einem Editor, der Zeichenfläche und weiteren Werkzeugen in einem Programm vereint, ohne zusätzliche Abhängigkeiten.

- Die Möglichkeit, die Befehle der Sprache TurtleScript mit den üblichen Verfahren in KDE zu übersetzen.
- TurtleScript unterstützt benutzerdefinierte Funktionen, Rekursion und dynamische Typzuweisung.
- Die Befehle können Sie jederzeit langsamer ausführen lassen, anhalten oder abbrechen.
- Ein leistungsfähiger Editor mit intuitiver Syntaxhervorhebung, Zeilennummerierung, Fehlermarkierung, visueller Kontrolle der Ausführung und mehr.
- Die Zeichenfläche, auf der die Schildkröte zeichnet, kann entweder als Rasterbild (PNG) oder als Vektorzeichnung (SVG) gedruckt und gespeichert werden.
- Kontexthilfe: Drücken Sie **F2** oder benutzen Sie **Hilfe** → **Hilfe zu: ...**, um zu dem aktuellen Quelltext unter dem Cursor eine passende Erläuterung zu erhalten.
- Ein Fehlerdialog, der auf die Fehler im Programm verweist und sie rot markiert.
- Einfache Programmiersprache.
- Integrierte Beispielprogramme, die den Einstieg erleichtern. Diese Beispiele werden mit dem üblichen Verfahren in KDE übersetzt.

Kapitel 2

Arbeiten mit KTurtle



Das Hauptfenster von KTurtle besteht aus drei Teilen: dem Editor (1) auf der linken Seite, wo Sie die TurtleScript-Befehle eingeben, der Zeichenfläche (2) in der Mitte, auf der die Schildkröte zeichnet, und dem Kontrollfenster (3) mit Informationen über die Variablen während der Programmausführung. Außerdem gibt es die Menüleiste (5), in der Sie alle Befehle erreichen können, die Werkzeugleiste (4) für den direkten Zugriff auf häufig gebrauchte Befehle, die **Konsole**, in der Sie einen einzelnen Befehl eingeben und testen können und die Statusleiste, wo Sie den Zustand von KTurtle sehen können.

2.1 Der Editor

Im Editor geben Sie die TurtleScript-Befehle ein. Die meisten Funktionen finden Sie in den Menüeinträgen Bearbeiten. Der Editor kann auf jeder Seite des Hauptfensters oder losgelöst vom Hauptfenster irgendwo auf Ihrer Arbeitsfläche angeordnet werden.

Es gibt mehrere Möglichkeiten, Quelltext im Editor zu erstellen. Am einfachsten ist es, ein Beispiel zu laden: Wählen Sie mit **Datei** → **Beispiele** im ein Beispiel aus. Die ausgewählte Datei wird im Editor geöffnet, dann können Sie mit **Ausführen** → **Ausführen** (**Alt**+**F2**) in der Menüleiste oder **Ausführen** in der Werkzeugleiste das Programm starten.

Sie können TurtleScript-Dateien mit **Datei** → **Öffnen ...** einfügen.

Als dritte Möglichkeit können Sie den eigenen Quelltext direkt im Editor eingeben oder d. h. aus diesem Handbuch kopieren und einfügen.

2.2 Die Zeichenfläche

Die Zeichenfläche ist der Bereich, wo die Schildkröte die Ausgaben der Befehle zeichnet. Nach der Eingabe des Quelltextes im Editor und der Ausführung kann folgendes passieren: Entweder der Quelltext wird fehlerfrei ausgeführt und Sie sehen dann Änderungen auf der Zeichenfläche oder Sie haben einen Fehler im Quelltext und es erscheint ein Fenster, das Ihnen die Art des Fehlers meldet.

Mit dem Mausrad können Sie die den Ausschnitt der Zeichenfläche vergrößern oder verkleinern.

2.3 Das Kontrollfenster

Im Kontrollfenster finden Sie während der Programmausführung Informationen über die Variablen, die gelernten Funktionen und den Quelltextbaum.

Das Kontrollfenster können Sie an jeder Seite des Hauptfenster oder freischwebend auf Ihrem Bildschirm anordnen.

2.4 Die Werkzeugleiste

Hier können Sie schnell die am häufigsten gebrauchten Aktionen erreichen. Die Werkzeugleiste enthält außerdem die das Eingabefeld **Konsole**, mit dem Sie schnell Befehle zum Testen ausführen können, ohne den Text im Editor zu verändern.

Die Werkzeugleiste können Sie mit **Einstellungen** → **Werkzeugleisten einrichten ...** an Ihre Bedürfnisse anpassen.

2.5 Die Menüleiste

In der Menüleiste finden Sie alle Befehle für KTurtle in folgenden Gruppen: **Datei**, **Bearbeiten**, **Zeichenfläche**, **Ausführen**, **Extras**, **Einstellungen** und **Hilfe**. Dieser Abschnitt beschreibt alle Menüeinträge.

2.5.1 Das Menü Datei

Datei → **Neu (Strg-N)**

Erstellt eine neue leere TurtleScript-Datei.

Datei → **Öffnen ... (Strg+O)**

Öffnet eine TurtleScript-Datei.

Datei → **Zuletzt geöffnete Dateien**

Öffnet eine der zuletzt geöffneten TurtleScript-Dateien.

Datei → Beispiele

Öffnet die TurtleScript-Beispielprogramme. Die Beispiele sind in der Sprache übersetzt, die Sie mit **Einstellungen → Skript-Sprache** ausgewählt haben.

Datei → Weitere Beispiele herunterladen ...

Öffnet den Dialog **„Neue Erweiterungen“ herunterladen**, um zusätzliche TurtleScript-Dateien aus dem Internet zu holen.

Datei → Speichern (Strg- S)

Speichert die gerade geöffnete TurtleScript-Datei.

Datei → Speichern unter ... (Strg-Umschalt-S)

Speichert die gerade geöffnete TurtleScript-Datei in einem bestimmten Ordner.

Datei → Als HTML exportieren ...

Exportiert den aktuellen Inhalt des Editors als HTML-Datei mit den farbigen Hervorhebungen.

Datei → Drucken ... (Strg+P)

Druckt den aktuellen Quelltext im Editor.

Datei → Beenden (Strg+Q)

Beendet KTurtle.

2.5.2 Das Menü Bearbeiten

Bearbeiten → Rückgängig (Strg+Z)

Macht die letzte Änderung des Quelltextes wieder rückgängig. Die Anzahl dieser Rückgängig-Schritte in KTurtle ist nicht begrenzt.

Bearbeiten → Rückgängig (Strg+Z)

Nimmt das letzte Rückgängig wieder zurück.

Bearbeiten → Ausschneiden (Strg+X)

Schneidet den gewählten Text aus dem Editor aus und fügt ihn in die Zwischenablage ein.

Bearbeiten → Kopieren (Strg+C)

Kopiert den gewählten Text aus dem Editor in die Zwischenablage.

Bearbeiten → Einfügen (Strg+V)

Fügt den Text aus der Zwischenablage in den Editor ein.

Bearbeiten → Alles auswählen (Strg+A)

Wählt den gesamten Text im Editor aus.

Bearbeiten → Suchen ... (Strg-F)

Mit diesem Befehl können Sie Textpassagen im Quelltext suchen.

Bearbeiten → Weitersuchen (F3)

Geht zum nächsten Auftreten des zuletzt gesuchten Textes.

Bearbeiten → Frühere suchen (Umschalt-F3)

Geht zum vorherigen Auftreten des zuletzt gesuchten Textes.

Bearbeiten → Überschreiben-Modus (Einfg)

Schaltet im Editor zwischen Einfügen und Überschreiben um.

2.5.3 Das Menü Zeichenfläche

Zeichenfläche → Als Bild (PNG) exportieren ...

Exportiert den aktuellen Inhalt der Zeichenfläche als Rasterbild im Format PNG (Portable Network Graphics).

Zeichenfläche → Als Zeichnung (SVG) exportieren ...

Exportiert den aktuellen Inhalt der Zeichenfläche als Vektorzeichnung im Format SVG (Scalable Vector Graphics).

Zeichenfläche → Zeichenfläche drucken ...

Druckt den aktuellen Inhalt der Zeichenfläche.

2.5.4 Das Menü Ausführen

Ausführen → Ausführen (F5)

Startet die Ausführung der Befehle im Editor.

Ausführen → Pause (F6)

Hält die Ausführung an. Diesen Menüeintrag können Sie nur auswählen, wenn gerade Befehle ausgeführt werden.

Ausführen → Abbrechen (F7)

Bricht die Ausführung ab. Diesen Menüeintrag können Sie nur auswählen, wenn gerade Befehle ausgeführt werden.

Ausführen → Bewegungs-Geschwindigkeit

Enthält eine Liste mit unterschiedlichen Ausführungsgeschwindigkeiten: **Volle Geschwindigkeit**, **Volle Geschwindigkeit (keine Hervorhebung und Kontrollfenster)**, **Langsam**, **Langsamer**, **Am Langsamsten**, **Schritt-für-Schritt**. Wenn die Ausführungsgeschwindigkeit auf **Volle Geschwindigkeit** (Standard) eingestellt ist, können Sie kaum verfolgen, was passiert. Manchmal ist dieses Verhalten gewollt, aber manchmal möchten Sie die Ausführung verfolgen können. In diesen Fällen setzen Sie die Ausführungsgeschwindigkeit auf **Langsam**, **Langsamer** oder **Am Langsamsten**, dann wird auch die aktuelle Position der Ausführung im Quelltexteditor angezeigt. Mit **Schritt-für-Schritt** können Sie die Befehle einzeln ausführen.

2.5.5 Das Menü Extras

Extras → Richtungsauswahl ...

Öffnet den Dialog zur Richtungsauswahl.

Extras → Farbauswahl ...

Öffnet den Dialog zur Farbauswahl.

2.5.6 Das Menü Einstellungen

Einstellungen → Skript-Sprache

Wählen Sie hier die Sprache für den Quelltext aus.

Einstellungen → Quelltexteditor anzeigen (Strg-E)

Zeigt den Editor an oder blendet ihn aus.

Einstellungen → Kontrollfenster anzeigen (Strg- I)

Zeigt das Kontrollfenster an oder blendet es aus.

Einstellungen → Fehler anzeigen

Schaltet die Anzeige des **Fehler**fensters mit einer Liste aller Fehler des aktuellen Quelltextes ein oder aus. Ist das Fehlerfenster eingeblendet, klicken Sie auf **Zeichenfläche**, um die Schildkröte wieder zu sehen.

Einstellungen → Zeilennummern anzeigen (F11)

Mir diesem Befehl werden im Editor Zeilennummern angezeigt. Das erleichtert die Fehlersuche.

Einstellungen → Werkzeugleiste anzeigen

Zeigt die Werkzeugleiste an bzw. blendet sie aus.

Einstellungen → Statusleiste anzeigen

Zeigt die Statusleiste an bzw. blendet sie aus.

Einstellungen → Kurzbefehle festlegen ...

Öffnet den KDE-Standarddialog zur Einstellung der Kurzbefehle.

Einstellungen → Werkzeugleisten einrichten ...

Öffnet den KDE-Standarddialog zur Einrichtung der Symbole in den Werkzeugleisten.

2.5.7 Das Menü Hilfe

KTurtle hat das bekannten KDE-Menü **Hilfe** aus den KDE-Grundlagen mit einem zusätzlichen Eintrag:

Hilfe → Hilfe zu: ... (F2)

Dies ist eine sehr hilfreiche Funktion, sie zeigt Hilfe zum Quelltext, in dem der Cursor gerade steht. Wenn Sie z. B. den Befehl **`drucke`** in Ihrem Quelltext benutzen und die Hilfe zu diesem Befehl lesen möchten, bewegen Sie den Cursor in diesen Befehl **`drucke`** und drücken **F2**. Das Handbuch zeigt Ihnen dann alle Informationen zum Befehl **`drucke`**.

Diese Funktion ist in der Lernphase des Programmierens mit TurtleScript sehr hilfreich.

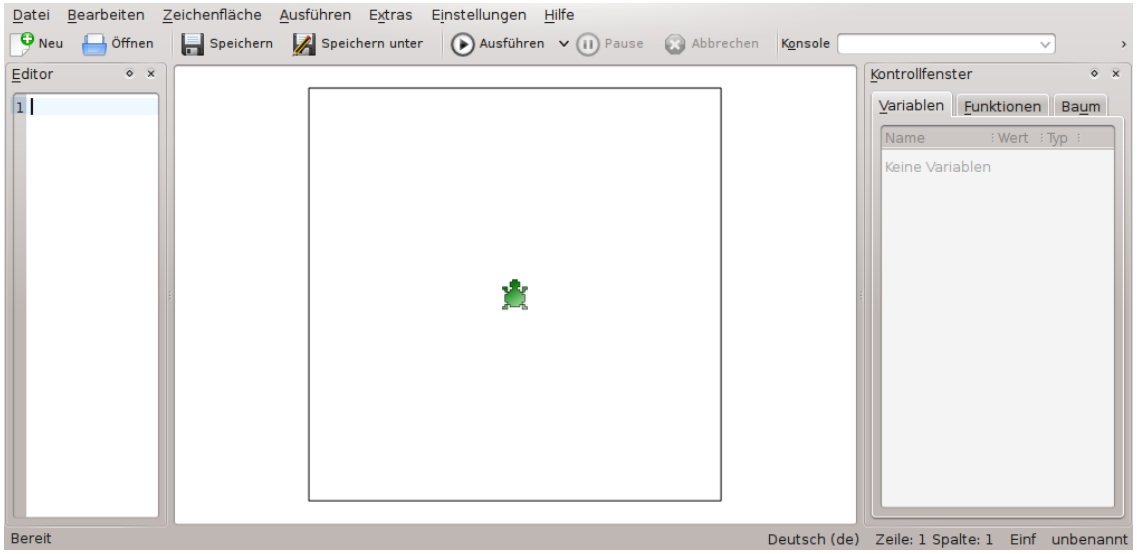
2.6 Die Statusleiste

In der Statusleiste sehen Sie Informationen über den Zustand von KTurtle. Auf der linken Seite wird das Ergebnis des letzten Befehls gezeigt. Auf der rechten Seite sehen Sie die aktuelle Position des Cursors (Zeilen- und Spaltennummer). In der Mitte der Statusleiste wird die aktuell eingestellte Sprache für die Befehle angezeigt.

Kapitel 3

Erste Schritte

Beim ersten Start von KTurtle sehen Sie folgendes Bild:



In diesem Handbuch zur Einführung wird angenommen, dass die Sprache der TurtleScript-Befehle auf deutsch eingestellt ist. Die Spracheinstellung können Sie unter **Einstellungen** → **Skript-Sprache** finden. Die hier für KTurtle eingestellte Sprache wird für die TurtleScript-Befehle verwendet, das ist nicht die Sprache für KDE und für die grafische Oberfläche und die Menüs von KTurtle.

3.1 Erste Schritte mit TurtleScript: Spielen Sie mit der Schildkröte!

Wie Sie sicher bemerkt haben, steht die Schildkröte mitten auf der Zeichenfläche: Sie lernen hier, die Schildkröte mit Befehlen im Editor zu kontrollieren.

3.1.1 Die Schildkröte bewegt sich

Beginnen Sie damit, die Schildkröte in Bewegung zu setzen. Unsere Schildkröte kennt drei Arten von Bewegungen. (1) sie kann vorwärts oder rückwärts gehen, (2) sie kann nach links oder rechts

drehen und (3) sie kann direkt zu einer Position auf der Zeichenfläche springen. Versuchen Sie zum Beispiel diese Befehle:

```
vorwärts 100
linksdrehen 90
```

Im Editor können Sie Text direkt eingeben oder kopieren und einfügen. Mit **Ausführen** → **Ausführen** sehen Sie das Ergebnis.

Nach der Eingabe und Ausführung der oben gezeigten Befehle im Editor ist Ihnen vielleicht folgendes aufgefallen:

1. Die Schildkröte bewegt sich — nach der Ausführung der Befehle — nach oben, zeichnet eine Linie und macht dann eine Vierteldrehung nach links. Dies haben Sie mit den Befehlen **`vorwärts`** und **`linksdrehen`** erreicht.
2. Die Farbe des Codes hat sich bei der Eingabe verändert, diese Eigenschaft nennt sich *intuitive Hervorhebung* —, verschiedene Arten von Befehlen werden unterschiedlich dargestellt. Dies erleichtert das Lesen von Quelltexten.
3. Die Schildkröte zeichnet eine dünne schwarze Linie.
4. Vielleicht haben Sie auch eine Fehlermeldung bekommen. Das kann zwei Gründe haben, entweder haben Sie die Befehle falsch geschrieben oder Sie müssen noch die richtige Sprache für die TurtleScript-Befehle einstellen (mit **Einstellungen** → **Skript-Sprache**).

Sie sehen wahrscheinlich, das **`vorwärts 100`** die Schildkröte vorwärts bewegt und dabei eine Linie zeichnet, und das **`linksdrehen 90`** die Schildkröte um 90 Grad nach links dreht.

In den folgenden Verweisen zum Referenzhandbuch finden Sie ausführliche Erklärungen der neuen Befehle: **`vorwärts`**, **`rückwärts`**, **`linksdrehen`** und **`rechtsdrehen`**.

3.1.2 Weitere Beispiele

Das erste Beispiel war sehr einfach, deshalb machen wir weiter!

```
zurücksetzen

papiergröße 200,200
papierfarbe 0,0,0
stiftfarbe 255,0,0
stiftbreite 5

gehe 20,20
richtung 135

vorwärts 200
linksdrehen 135
vorwärts 100
linksdrehen 135
vorwärts 141
linksdrehen 135
vorwärts 100
linksdrehen 45

gehe 40, 100
```

Auch hier können Sie den Quelltext eingeben oder kopieren und in den Editor einfügen. Oder Sie öffnen die Datei `pfeil` im Menü **Beispiele** und führen sie aus (mit **Ausführen** → **Ausführen**),

um das Ergebnis zu sehen. In den folgenden Beispielen werden diese Möglichkeiten zur Eingabe von Quelltext nicht mehr ausdrücklich genannt.

Wie Sie vielleicht festgestellt haben, benutzt das zweite Beispiel viel mehr Quelltext. Außerdem enthält es viele neue Befehle, die im folgenden kurz erklärt werden.

Mit dem Befehl **`zurücksetzen`** wird alles wieder wie direkt nach dem Start von KTurtle eingestellt.

`papiergröße 200,200` setzt die Breite und Höhe der Zeichenfläche auf 200 Pixel. Breite und Höhe sind gleich, also ist die Zeichenfläche ein Quadrat.

`papierfarbe 0,0,0` färbt die Zeichenfläche schwarz. **`papierfarbe 0,0,0`** ist eine RGB-Kombination, in der alle Werte auf **`0`** gesetzt sind, das Ergebnis ist schwarz.

`stiftfarbe 255,0,0` setzt die Farbe des Zeichenstiftes auf Rot. **`255,0,0`** ist eine RGB-Kombination, in der nur der Wert für Rot auf **`255`** gesetzt ist, während die anderen (Grün und Blau) den Wert **`0`** (aus) haben. Das Ergebnis ist ein leuchtendes Rot.

Wenn Sie die Farbwerte nicht verstehen, finden Sie dazu weitere Erläuterungen unter dem Begriff RGB-Kombinationen

`stiftbreite 5` setzt die Breite des Zeichenstiftes auf **`5`** Pixel. Nach diesem Befehl zeichnet die Schildkröte jede Linie mit einer Breite von **`5`**, bis Sie die **`stiftbreite`** auf einen anderen Wert einstellen.

`gehe 20,20` bewegt die Schildkröte auf eine bestimmte Stelle auf der Zeichenfläche, die 20 Pixel vom linken Rand und 20 Pixel vom oberen Rand der Zeichenfläche entfernt ist. Mit dem Befehl **`gehe`** zeichnet die Schildkröte keine Linie.

`richtung 135` setzt die Bewegungsrichtung der Schildkröte. Die Befehle **`linksdrehen`** und **`rechtsdrehen`** ändern die Bewegungsrichtung der Schildkröte relativ zur vorhandenen Richtung. Der Befehl **`richtung`** setzt die Bewegungsrichtung unabhängig von der vorhandenen Richtung neu.

Nach dem Befehl **`richtung`** folgen viele **`vorwärts`** und **`linksdrehen`** Befehle. Diese Befehle erzeugen die aktuelle Zeichnung.

Zum Schluss bewegt der Befehl **`gehe`** die Schildkröte zu Seite.

Lesen Sie die Verknüpfungen zur Befehlsreferenz, in der die einzelnen Befehle ausführlicher erklärt werden.

Kapitel 4

Programmreferenz für TurtleScript

Dies ist die Referenz der in KTurtle verwendeten Sprache TurtleScript. Im ersten Abschnitt dieses Kapitels werden einige Grundzüge der grammatischen Regeln des Sprache TurtleScript vorgestellt. Der zweite Abschnitt behandelt ausschließlich mathematische Operatoren, boolesche Operatoren (wahr/falsch) und Vergleichs-Operators.Im dritten Abschnitt finden Sie eine große Liste aller Befehle, die einzeln erklärt werden. Abschnitt vier behandelt die Zuweisung von Werten and Variablen. Zum Schluss wird im Abschnitt fünf die Steuerung des Programms mit Kontrollanweisungen und dann im Abschnitt sechs erläutert, wie eigene Befehle oder Funktionen mit **lerne** erzeugt werden.

4.1 Die grammatischen Regeln der Sprache TurtleScript

Wie in jeder Sprache gibt es in TurtleScript verschiedene Arten von Wörtern und Symbolen. Englisch zum Beispiel unterscheidet zwischen Verben wie „gehen" oder „singen" und Substantiven wie „Schwester" oder „Haus". Diese verschiedene Arten von Wörtern und Symbolen werden für verschiedene Zwecke benutzt. TurtleScript ist eine Programmiersprache, die dazu benutzt wird, KTurtle Anweisungen zu geben.

In diesem Abschnitt werden einige der verschiedenen Arten von Wörtern und Symbolen in TurtleScript kurz erklärt. In TurtleScript gibt es Kommentare, Befehle und die drei verschiedenen Arten von echten Werten: Zahlen, Zeichenfolgen und boolesche Werte (wahr/falsch).

4.1.1 Kommentare

Ein Programm besteht aus Befehlen, die beim Ablauf ausgeführt werden, und sogenannten Kommentaren. Kommentare werden nicht ausgeführt, KTurtle ignoriert sie beim Starten des Programms. Kommentare werden benutzt, damit andere Programmierer den Quelltext besser verstehen. Jeder Text nach einem **#**-Zeichen wird in TurtleScript als Kommentar betrachtet. Dieses kleine Programm zum Beispiel tut nichts:

```
# dieses kleine Programm tut nichts, das ist nur ein Kommentar
```

Das Programm ist natürlich nutzlos, erklärt aber gut die Wirkung von Kommentaren.

Kommentare sind sehr hilfreich, wenn das Programm komplizierter und umfangreicher wird. Außerdem erleichtern sie anderen Programmierern, den Quelltext zu verstehen. Im folgenden Programm werden Kommentare zusammen mit dem Befehl drucke verwendet:

```
# Cies Breijs hat dieses Programm geschrieben.
drucke "dieser Text wird auf der Zeichenfläche ausgedruckt"
# die vorige Zeile ist kein Kommentar, aber die Folgende:
# drucke "dieser Text wird nicht ausgedruckt"
```

Die erste Zeile beschreibt das Programm. Die zweite Zeile wird von KTurtle ausgeführt, es wird **`dieser Text wird auf der Zeichenfläche ausgedruckt`** geschrieben. Die dritte Zeile ist ein Kommentar. In der vierten Zeile steht ein Kommentar, der einen Quelltext in TurtleScript enthält. Wird das Symbol **`#`** in der vierten Zeile entfernt, wird der Druckbefehl von KTurtle ausgeführt. Programmierer bezeichnen das als „auskommentieren".

Kommentarzeilen werden im Quelltexteditor hellgrau hervorgehoben.

4.1.2 Befehle

Mit Befehlen geben Sie der Schildkröte oder KTurtle Anweisungen, was sie tun soll. Einige Befehle brauchen eine Eingabe, andere geben einen Wert zurück.

```
# der Befehl vorwärts braucht eine Eingabe, in diesem Fall die Zahl 100:
vorwärts 100
```

Die erste Zeile ist ein Kommentar. Die zweite Zeile enthält den Befehl **`vorwärts`** und die Zahl **`100`**. Die Zahl ist nicht Bestandteil des Befehls, sie ist die „Eingabe" für den Befehl. Einige Befehle wie z. B. **`gehe`** brauchen mehr als einen Eingabewert. Mehrere Werte müssen durch Komma (`,`) getrennt werden.

Einige Befehle wie z. B. **`gehe`** brauchen mehr als einen Eingabewert. Mehrere Werte müssen durch Komma (`,`) getrennt werden.

Eine ausführliche Übersicht aller von KTurtle unterstützten Befehle finden Sie hier. Die eingebauten Befehle werden dunkelblau hervorgehoben.

4.1.3 Zahlen

Wahrscheinlich wissen Sie schon ziemlich viel über Zahlen. In KTurtle werden Zahlen etwa so wie in der Sprache oder in der Mathematik verwendet.

Es gibt die so genannten natürlichen Zahlen: **`0`**, **`1`**, **`2`**, **`3`**, **`4`**, **`5`** usw. Die negativen Zahlen: **`-1`**, **`-2`**, **`-3`** usw. Und die Zahlen mit Dezimalstellen oder reelle Zahlen, zum Beispiel: **`0.1`**, **`3.14`**, **`33.3333`**, **`-5.05`**, **`-1.0`**. Der Punkt (`.`) wird als Dezimaltrennzeichen benutzt.

Zahlen können in mathematischen Berechnungen und Vergleichen benutzt werden. Sie können auch in Variablen gespeichert werden. Zahlen werden in dunklem Rot hervorgehoben.

4.1.4 Zeichenfolgen

Zuerst ein Beispiel:

```
drucke "Hallo, ich bin eine Zeichenfolge."
```

In diesem Beispiel ist **`drucke`** ein Befehl und **`"Hallo, ich bin eine Zeichenfolge."`** eine Zeichenfolge. Zeichenfolgen haben am Anfang und am Ende das Zeichen ", daran erkennt sie KTurtle.

Zeichenfolgen können in Variablen abgelegt werden, genauso wie Zahlen. Im Gegensatz zu Zahlen können Zeichenfolgen nicht in mathematischen Berechnungen und Vergleichen benutzt werden. Zeichenfolgen werden in Rot hervorgehoben.

4.1.5 Boolesche Werte (wahr/falsch)

Es gibt nur zwei Boolesche Werte: **wahr** und **falsch**. Diese Werte werden auch „**Ein**" und „**Aus**", „**Ja**" und „**Nein**", „**Eins**" und „**Null**"genannt. Aber in TurtleScript heißen Sie immer **wahr** und **falsch**. Betrachten Sie diesen Quelltext in TurtleScript:

```
$a = wahr
```

Im Kontrollfenster können Sie sehen, dass die Variable **$a** den Wert **wahr** hat und ein Boolescher Wert ist.

Boolesche Werte sind häufig das Ergebnis eines Vergleichs, wie der folgende Quelltext in TurtleScript zeigt:

```
$antwort = 10 > 3
```

Die Variable **$antwort** erhält den Wert **wahr**, weil **10** größer als **3** ist.

Boolesche Werte, **wahr** und **falsch**, werden dunkelrot hervorgehoben.

4.2 Mathematische, Boolesche und Vergleichs-Operatoren

Der Titel dieses Abschnitts mag sehr kompliziert klingen, aber das täuscht.

4.2.1 Mathematische Operatoren

Dies sind alle grundlegenden mathematischen Symbole: addieren (**+**), subtrahieren (**-**), multiplizieren (*****), dividieren (**/**) und die Potenz (**^**).

Ein kleines Beispiel der mathematischen Operatoren, die Sie in TurtleScript verwenden können:

```
$plus      = 1 + 1
$minus     = 20 - 5
$multipiziert = 15 * 2
$geteilt   = 30 / 30
$potenz    = 2 ^ 2
```

Die Werte, die sich aus diesen mathematischen Operationen ergeben, werden den verschiedenen Variablen zugewiesen. Im Kontrollfenster werden die Variablen und ihre Werte angezeigt.

Für eine einfache Berechnung geben Sie zum Beispiel folgendes ein:

```
drucke 2010-12
```

Nun ein Beispiel mit Klammern:

```
drucke ( ( 20 - 5 ) * 2 / 30 ) + 1
```

Der Ausdruck in Klammern wird zuerst berechnet. In diesen Beispiel wird also 20-5 berechnet, dann mit 2 multipliziert, durch 30 dividiert und dann 1 addiert (das ergibt 2). Klammern können auch in anderen Fällen benutzt werden.

KTurtle hat noch erweiterte mathematische Funtionen in der Form von Befehlen. Schauen Sie sich die folgenden Befehle an, dabei handelt es sich um fortgeschrittene mathematische Operationen: rund, zufall, wurzel , pi, sin, cos, tan, arcsin, arccos, arctan.

4.2.2 Boolesche Operatoren (wahr/falsch)

Während mathematische Operatoren Zahlen verarbeiten, behandeln boolesche Operatoren die boolesche Werte **wahr** und **falsch**. Es gibt nur drei boolesche Operatoren, nämlich: **und**, **oder** und **nicht**. Im folgenden Quelltext von TurtleScript wird gezeigt, wie sie verwendet werden:

```
$und_1_1 = wahr und falsch    # -> wahr
$und_1_0 = wahr und falsch   # -> false
$und_0_1 = falsch und wahr   # -> false
$und_0_0 = falsch und falsch  # -> false

$oder_1_1 = wahr oder wahr    # -> wahr
$oder_1_0 = wahr oder falsch   # -> wahr
$oder_0_1 = falsch oder wahr   # -> wahr
$oder_0_0 = falsch oder falsch  # -> falsch

$nicht_1 = nicht wahr   # -> falsch
$nicht_0 = nicht falsch  # -> wahr
```

Im Kontrollfenster werden die Ergebnisse dargestellt, sie sind aber auch als Kommentar am Ende der Zeilen angefügt. **und** ergibt nur **wahr**, wenn beide Seiten **wahr** sind. **oder** ergibt **wahr**, wenn mindestens eine der beiden Seiten **true** ist. Und **nicht** ändert ein **wahr** in **falsch** und ein **falsch** in **wahr**.

Boolesche Operatoren werden pinkfarben hervorgehoben.

4.2.2.1 Einige etwas schwierigere Beispiele

Betrachten sie das folgende Beispiel mit **und**:

```
$a = 1
$b = 5
wenn (($a < 10) und ($b == 5)) und ($a < $b) {
  drucke "Hallo"
}
```

In diesem Quelltext von TurtleScript wird das Ergebnis von drei Vergleichsoperatoren mit dem Operator **und** miteinander verbunden. Daher müssen alle drei Vergleiche „wahr" ergeben, damit das Wort „Hallo" gedruckt wird..

Ein Beispiel mit **oder**:

```
$n = 1
wenn ($n < 10) oder ($n == 2) {
  drucke "Hallo"
}
```

In diesem Quelltext von TurtleScript ergibt die linke Seite des **oder** „wahr", die rechte Seite „falsch". Da eine der beiden Seiten des Operators **oder** „wahr" ergibt, ist das Ergebnis des Operartors **oder** auch „wahr". Damit wird das Wort „Hallo" gedruckt.

Und schließlich ein Beispiel mit **nicht**, das „wahr" in „falsch" und „falsch" in „wahr" ändert:

```
$n = 1
wenn nicht ($n == 3) {
  drucke "Hallo"
} sonst {
  drucke "nicht Hallo ;-)"
}
```

4.2.3 Vergleichs-Operatoren

Betrachten Sie diesen einfachen Vergleich:

```
$ergebnis = 10 > 3
```

Hier wird **10** mit **3** mit Hilfe des Operators „größer als" verglichen. Das Ergebnis dieses Vergleichs, der Boolesche Wert **wahr**, wird in der Variablen **ergebnis** gespeichert.

Alle Zahlen und Variablen, die Zahlen enthalten, können durch die Vergleichs-Operatoren miteinander verglichen werden.

Hier sind alle möglichen Vergleichs-Operatoren aufgeführt:

$A == B	gleich	Ergebnis ist „wahr" wenn **$A** gleich **$B** ist
$A != $B	nicht gleich	Ergebnis ist „wahr" wenn **$A** ungleich **$B** ist
$A > **$B**	größer als	Ergebnis ist „wahr" wenn **$A** größer ist als **$B**
$A < **$B**	kleiner als	Ergebnis ist „wahr" wenn **$A** kleiner ist als **$B**
$A >= $B	größer oder gleich	Ergebnis ist „wahr" wenn **$A** größer oder gleich **$B** ist
$A <= $B	kleiner oder gleich	Ergebnis ist „wahr" wenn **$A** kleiner oder gleich **$B** ist

Tabelle 4.1: Art der Frage

$A und $B müssen Zahlen oder Variablen sein, die Zahlen enthalten.

4.3 Befehle

Mit Befehlen sagen Sie der Schildkröte oder KTurtle, was sie tun soll. Einige Befehle brauchen Eingabewerte, andere geben Werte zurück. In diesem Abschnitt werden alle eingebauten Befehle für KTurtle erklärt. Zusätzlich können mit **lerne** benutzerdefinierte Befehle erstellt werden. Eingebaute Befehle, die hier vorgestellt werden, sind dunkelblau hervorgehoben.

4.3.1 Die Schildkröte in Bewegung setzen

Es gibt verschiedene Befehle, um die Schildkröte auf dem Bildschirm zu bewegen.

vorwärts (vw)

```
vorwärts X
```

vorwärts bewegt die Schildkröte um einen Betrag von X Pixeln vorwärts. Wenn der Stift unten ist, zeichnet die Schildkröte eine Linie. **vorwärts** kann als **vw** abgekürzt werden.

rückwärts (rw)

```
rückwärts X
```

rückwärts bewegt die Schildkröte um einen Betrag von X Pixeln zurück. Wenn der Stift unten ist, zeichnet die Schildkröte eine Linie. **rückwärts** kann als **rw** abgekürzt werden.

linksdrehen (ld)

```
linksdrehen X
```

linksdrehen dreht die Schildkröte um einen Betrag von X Grad nach links. **linksdrehen** kann als **ld** abgekürzt werden.

rechtsdrehen (rd)

```
rechtsdrehen X
```

rechtsdrehen dreht die Schildkröte um einen Betrag von X Grad nach rechts. **rechtsdrehen** kann als **rd** abgekürzt werden.

richtung (rtg)

```
richtung X
```

richtung setzt die Bewegungsrichtung der Schildkröte auf X Grad gerechnet von Null, unabhängig von der vorherigen Bewegungsrichtung der Schildkröte. **richtung** kann als **rtg** abgekürzt werden.

holerichtung

```
holerichtung
```

holerichtung gibt die Bewegungsrichtung der Schildkröte in Grad bezogen auf die Richtung senkrecht nach oben, die als Null definiert ist.

mitte

```
mitte
```

mitte bewegt die Schildkröte in die Mitte der Zeichenfläche.

gehe

```
gehe X,Y
```

gehe bewegt die Schildkröte an eine bestimmten Stelle auf der Zeichenfläche. Dieser Ort ist X Pixel vom linken Rand und Y Pixel vom oberen Rand der Zeichenfläche entfernt.

gehex

```
gehex X
```

Mit dem Befehl **gehex** bewegt sich die Schildkröte auf X Pixel vom linken Rand der Zeichenfläche, bleibt dabei aber auf gleicher Höhe. **gehex** kann als **gx** abgekürzt werden.

gehey

```
gehey Y
```

Mit dem Befehl **gehey** bewegt sich die Schildkröte auf Y Pixel vom oberen Rand der Zeichenfläche, bleibt dabei aber auf gleicher Entfernung vom linken Rand der Zeichenfläche. **gehey** kann als **gy** abgekürzt werden.

ANMERKUNG
Mit den Befehlen **gehe**, **gehex**, **gehey** und **mitte** zeichne die Schildkröte keine Linie, egal ob der Stift oben oder unten ist.

4.3.2 Wo ist die Schildkröte?

Es gibt zwei Befehle, die die Position der Schildkröte auf dem Bildschirm zurückgeben.

holex

holex gibt die Anzahl der Pixel vom linken Rand der Zeichenfläche bis zur aktuellen Position der Schildkröte zurück.

holey

holey gibt die Anzahl der Pixel vom oberen Rand der Zeichenfläche bis zur aktuellen Position der Schildkröte zurück.

4.3.3 Die Schildkröte kann zeichnen

Die Schildkröte kann mit einem Stift Linien zeichnen, wenn sie sich bewegt. Um diesen Stift zu kontrollieren, gibt es einige Befehle, die diesem Abschnitt beschrieben werden.

stifthoch (sh)

```
stifthoch
```

stifthoch hebt den Stift von der Zeichenfläche. Wenn der Zeichenstift „hoch" ist, zeichnet die Schildkröte bei ihrer Bewegung keine Linie. Siehe auch **stiftrunter**. **stifthoch** kann als **sh** abgekürzt werden.

stiftrunter (sr)

```
stiftrunter
```

stiftrunter drückt den Stift auf die Zeichenfläche. Wenn der Zeichenstift „runter" ist, zeichnet die Schildkröte eine Linie bei ihrer Bewegung. Siehe auch **stifthoch**. **stiftrunter** kann als **sr** abgekürzt werden.

stiftbreite (sb)

```
stiftbreite X
```

stiftbreite setzt die Stiftbreite (die Linienbreite) auf X Pixel. **stiftbreite** kann als **sb** abgekürzt werden.

stiftfarbe (sf)

```
stiftfarbe R,G,B
```

stiftfarbe setzt die Farbe für den Zeichenstift. **stiftfarbe** braucht eine RGB-Kombination als Eingabewert. **stiftfarbe** kann als **sf** abgekürzt werden.

4.3.4 Befehle für die Zeichenfläche

Es gibt mehrere Befehle für die Zeichenfläche.

papiergröße (pg)

```
papiergröße X,Y
```

Mit dem Befehl **papiergröße** stellen Sie die Größe der Zeichenfläche ein. Als Eingabewert sind X und Y erforderlich, dabei ist X die neue Breite in Pixeln und Y die neue Höhe der Zeichenfläche in Pixeln. **papiergröße** kann als **pg** abgekürzt werden.

papierfarbe (pf)

```
papierfarbe R,G,B
```

papierfarbe setzt die Farbe der Zeichenfläche. **papierfarbe** braucht eine RGB-Kombination als Eingabewert. **papierfarbe** kann als **pf** abgekürzt werden.

4.3.5 Befehle um aufzuräumen

Es gibt zwei Befehle, um die Zeichenfläche nach einem Fehler wieder aufzuräumen.

lösche (lös)

```
lösche
```

Mit **lösche** entfernen Sie alle Zeichnungen von der Zeichenfläche. Alle anderen Einstellungen bleiben erhalten: Die Position und die Bewegungsrichtung der Schildkröte, ob die Schildkröte sichtbar ist und Farbe und Größe der Zeichenfläche.

zurücksetzen

```
zurücksetzen
```

zurücksetzen räumt gründlicher auf als der Befehl **lösche**. Nach der Ausführung des Befehls **zurücksetzen** sind alle Einstellungen wie beim Start von KTurtle. Die Schildkröte steht mitten auf einer weißen Zeichenfläche, zeichnet die Linien mit schwarzer Farbe und die Papiergröße beträgt 400 x 400 Pixel.

4.3.6 Die Schildkröte ist ein Grafiksymbol

Viele Leute wissen nicht was Grafiksymbole sind, hier eine kurze Erklärung: Grafiksymbole sind kleine Bilder, die auf dem Bildschirm bewegt werden können. (Weitere Informationen finden Sie im Glossar unter Grafiksymbol). Die Schildkröte ist so ein Grafiksymbol.

Als nächstes finden Sie eine vollständige Übersicht aller Befehle für das Grafiksymbol.

> ANMERKUNG
> Diese Version von KTurtle erlaubt noch nicht die Verwendung anderer Grafiksymbole als die Schildkröte. In späteren Versionen können Sie die Schildkröte durch selbst entworfene Symbole ersetzen.

zeigebild (zb)

```
zeigebild
```

zeigebild zeigt das Symbol der Schildkröte wieder an, wenn es ausgeblendet war. **zeigebild** kann als **zb** abgekürzt werden.

versteckebild (vb)

```
versteckebild
```

versteckebild schaltet das Symbol der Schildkröte aus, wenn sie nicht in Ihre Zeichnung passt. **versteckebild** kann als **vb** abgekürzt werden.

4.3.7 Kann die Schildkröte schreiben?

Die Antwort lautet „ja". Die Schildkröte kann jeden gewünschten Text schreiben.

drucke

```
drucke X
```

Mit dem Befehl **drucke** schreibt die Schildkröte Text auf die Zeichenfläche. **drucke** braucht Zahlen und Zeichenfolgen als Eingabewert. Sie können auch mehrere Zahlen und Zeichenfolgen mit dem Symbol **+** verbinden, wie das folgende kleine Beispiel zeigt:

```
$jahr = 2003
$autor = "Cies"
drucke $autor + " startete das KTurtle Projekt " + $jahr + " und ↩
    arbeitet immer noch gerne daran!"
```

schriftgröße

```
schriftgröße X
```

schriftgröße setzt die Größe der Schrift, die mit dem Befehl **drucke** benutzt wird. **schriftgröße** braucht nur eine Zahl als Eingabewert. Die Größe der Schrift wird in Pixeln angegeben.

4.3.8 Mathematische Befehle

Im Folgenden werden die erweiterten mathematischen Befehle in KTurtle beschrieben.

rund

```
rund(x)
```

rund die angegebene Zahl zur nächsten ganzen Zahl.

```
drucke rund(10.8)
vorwärts 20
drucke rund(10.3)
```

mit diesen Anweisungen druckt die Schildkröte die Zahlen 11 and 10.

zufall (zuf)

```
zufall X,Y
```

zufall ist ein Befehl, der Eingabewerte braucht und einen Wert zurückgibt. Als Eingabe sind zwei Zahlen nötig, die erste (X) bestimmt die Untergrenze, die zweite (Y) die Obergrenze der Ausgabe. Der Rückgabewert ist eine zufällige Zahl größer oder gleich der unteren Grenze und kleiner oder gleich der oberen Grenze. Hier ein kurzes Beispiel:

```
wiederhole 500 {
  $x = zufall 1,20
  vorwärts $x
  linksdrehen 10 - $x
}
```

Mit dem Befehl **zufall** können Sie ein wenig Chaos in Ihr Programm einfügen.

mod

```
mod X,Y
```

Der Befehl **mod** gibt den Rest einer Division der ersten durch die zweite Zahl zurück.

wurzel

```
wurzel X
```

Der Befehl **wurzel** wird benutzt, um die Quadratwurzel einer Zahl X zu berechnen.

pi

```
pi
```

Dieser Befehl gibt die Konstante Pi, **3.14159** zurück.

sin, cos, tan

```
sin X
cos X
tan X
```

Diese drei Befehle sind die bekannten trigonometrischen Funktionen **sin**, **cos** und **tan**.Das Eingabeargument X dieser Befehle ist eine Zahl.

arcsin, arccos, arctan

```
arcsin X
arccos X
arctan X
```

Diese Befehle sind die inversen Funktionen von sin, cos und tan. Das Eingabeargument X dieser Befehle ist eine Zahl.

4.3.9 Eingaben und Nachrichten mit Dialogen

Ein Dialog ist ein kleines Fenster, das Meldungen anzeigt oder Eingaben ermöglicht. KTurtle hat zwei Befehle für Dialoge: **nachricht** und **frage**

nachricht

```
nachricht X
```

Der Befehl **nachricht** braucht eine Zeichenfolge als Eingabewert. Es öffnet sich ein Dialog und der Text der Zeichenfolge wird angezeigt.

```
nachricht "Cies startete KTurtle 2003 und arbeitet immer noch gerne ↩
    daran!"
```

frage

```
frage X
```

frage braucht eine Zeichenfolge als Eingabewert. Der Text der Zeichenfolge wird in einem Dialog angezeigt, genau wie bei nachricht, aber zusätzlich erscheint noch ein Eingabefeld. Nachdem eine Zahl oder eine Zeichenfolge in diese Textfeld eingegeben wurde, kann das Ergebnis in einer Variablen gespeichert oder als Argument an einen Befehl übergeben werden. Zum Beispiel

```
$ein = frage "In welchem Jahr sind Sie geboren?"
$aus = 2003-$ein
drucke "Im Jahr 2003 waren Sie irgendwann " + $aus + " Jahre alt."
```

Wenn Sie den Eingabedialog abbrechen oder keine Zeichen eingeben, hat die Variable keinen Inhalt.

4.4 Zuweisung von Variablen

Zuerst werden die Variablen vorgestellt, dann wird die Zuweisung von Werten erklärt.

Variablen sind Wörter oder Zeichenfolgen, die mit einem „$" beginnen, im Editor werden sie purpurfarben hervorgehoben.

Variablen können eine beliebige Zahl, Zeichenfolge oder Booleschen Wert (wahr/falsch) enthalten. Mit der Zuweisung **=** erhält eine Variable ihren Inhalt. Sie behält diesen Wert, bis das Programm beendet oder der Variablen ein anderer Wert zugewiesen wird.

Ist Variablen erst einmal ein Wert zugewiesen worden, können Sie wie ihr Inhalt benutzt werden. Als Beispiel dazu der folgende Quelltext in TurtleScript:

```
$x = 10
$x = $x / 3
drucke $x
```

Zuerst wird der Variablen **$x** der Wert **10** zugewiesen. Dann erhält **$x** den vorhandenen Wert geteilt durch **3** — in diesem Fall erhält **$x** den Wert **10 / 3**. Dann wird **$x** ausgegeben. In Zeile zwei und drei wird die Variable **$x** wie iht Inhalt benutzt.

Variablen muss ein Wert zugewiesen werden, ehe sie benutzt werden können. Ein Beispiel:

```
drucke $n
```

Mit diesem Befehl erhalten Sie eine Fehlermeldung.

Betrachten Sie den folgen Quelltext in TurtleScript:

```
$a = 2004
$b = 25

# der nächste Befehl druckt "2029"
drucke $a + $b
rückwärts 30
# der nächste Befehl druckt "2004 plus 25 gleich 2029"
drucke $a + " plus " + $b + " gleich " + ($a + $b)
rückwärts 30
```

In den ersten zwei Zeilen wird der Wert der Variablen **$a** auf 2004 und **$b** auf 25 gesetzt. Der Rest des Beispiels besteht aus zwei **drucke** Befehlen mit dem Befehl **rückwärts 30** dazwischen. Der Kommentar vor jedem **drucke** erklärt was ausgegeben wird. Der Befehl **rückwärts 30** ist erforderlich, um jede Ausgabe auf einer neuen Zeile zu schreiben. Dieses Beispiele zeigt, dass Variablen mit allen Operatoren in der gleichen Weise wie ihr Wert benutzt werden können. Außerdem können sie als Eingabe für Befehle dienen.

Ein weiteres Beispiel:

```
$name = frage "Wie heißt Du?"
drucke "Hallo " + $name + ", Viel Spaß beim Programmieren..."
```

Sie sehen, die Variable **$name** wird genauso wie eine Zeichenfolge behandelt.

Wenn Sie mit Variablen arbeiten, ist das Kontrollfenster sehr hilfreich. In diesem Fenster werden alle zur Zeit benutzten Variablen und ihr Wert angezeigt.

4.5 Kontrolle der Programmausführung

Mit den Befehlen zur Kontrollanweisung können Sie — wie der Name schon sagt — den Ablauf der Ausführung kontrollieren.

Befehle zur Kontrollanweisung werden durch eine dunkelgrüne, fettgedruckte Schrift hervorgehoben. Die meistens zusammen mit der Kontrollanweisung verwendeten Klammern werden durch schwarze Schrift hervorgehoben.

4.5.1 Lass die Schildkröte warten

Wenn Sie schon in KTurtle programmiert haben, ist Ihnen bestimmt aufgefallen, dass sich die Schildkröte beim Zeichnen sehr schnell bewegt. Dieser Befehl lässt die Schildkröte eine bestimmte Zeit warten.

warte

```
warte X
```

Mit **warte** wartet die Schildkröte X Sekunden.

```
wiederhole 36 {
  vorwärts 5
  rechtsdrehen 10
  warte 0.5
}
```

Dieser Quelltext zeichnet einen Kreis, aber die Schildkröte wartet eine halbe Sekunde nach jedem Schritt. Dies vermittelt den Eindruck einer Schildkröte, die sich langsam bewegt.

4.5.2 Kontrollanweisung „wenn"

wenn

```
wenn boolescher wert { ... }
```

Der Quelltext zwischen den Klammern wird nur dann ausgeführt, **wenn** der boolesche Wert „wahr" ist.

```
$x = 6
wenn $x > 5 {
  drucke "$x ist größer als fünf!"
}
```

In der ersten Zeilen weisen Sie **$x** den Wert 6 zu. In der zweiten Zeile wird ein Vergleichsoperator benutzt, um den Wert von **$x** > **5** zu ermitteln. Da das Ergebnis „wahr" ist 6 ist größer als 5, lässt Kontrollanweisung **wenn** den Quelltext zwischen den Klammern ausführen.

4.5.3 Kontrollanweisung „sonst”

sonst

```
wenn boolescher wert { ... } sonst { ... }
```

sonst kann zusammen mit der Kontrollanweisung **wenn** benutzt werden. Der Quelltext zwischen den Klammern nach **sonst** wird nur dann ausgeführt, wenn der boolesche Wert „falsch” ergibt:

```
zurücksetzen
$x = 4
wenn $x > 5 {
  drucke "$x ist größer als fünf!"
}
sonst
{
  drucke "$x ist kleiner als sechs!"
}
```

Der Vergleichsoperator wertet Ausdruck **$x** > **5** aus. Da 4 kleiner als 5 ist, ist das Ergebnis „falsch”. Das führt dazu, dass der Quelltext zwischen den Klammern nach **sonst** ausgeführt wird.

4.5.4 Die „solange”-Schleife

solange

```
solange boolescher wert { ... }
```

Die Kontrollanweisung **solange** ist der Anweisung **wenn** sehr ähnlich. Der Unterschied liegt darin, dass **solange** die Ausführung des Quelltext zwischen den Klammern (in einer Schleife) solange wiederholt, bis der booleschen Wert „falsch” ergibt:

```
$x = 1
solange $x < 5 {
  vorwärts 10
  warte 1
  $x = $x + 1
}
```

In der ersten Zeile wird **$x** der Wert 1 zugewiesen. In der zweiten Zeile wird **$x** < **5** ausgewertet. Da das Ergebnis „wahr” ist, lässt die Kontrollanweisung **solange** den Quelltext zwischen den Klammern ausführen, bis **$x** < **5** „falsch” ist. In diesem Beispiel wird der Quelltext zwischen den Klammern viermal ausgeführt, da bei jeder Ausführung der fünften Zeile **$x** um 1 vergrößert wird.

4.5.5 Die „wiederhole”-Schleife

wiederhole

```
wiederhole zahl { ... }
```

Die Kontrollanweisung **wiederhole** funktioniert ähnlich wie **solange**, nur dass mit **wiederhole** der Quelltext innerhalb der Klammern mit der angegebenen Anzahl von Wiederholungen ausgeführt wird.

4.5.6 Die „von"-Schleife, eine zählende Schleife

von

```
von Variable = Zahl bis Zahl { ... }
```

Die **von**-Schleife ist eine „zählende Schleife", d. h. zählt sie für Sie. Die erste Zahl bestimmt die Variable für den Wert in der ersten Schleife. Nach jedem Durchlauf der Schleife wird die Zahl vergrößert, bis die zweite Zahl erreicht ist.

```
von $x = 1 bis 10 {
  drucke $x * 7
  vorwärts 15
}
```

Jedes Mal, wenn der Quelltext zwischen den Klammern ausgeführt wird, vergrößert sich der Wert von **$x** um eins, bis **$x** den Wert 10 erreicht. Der Quelltext zwischen den Klammern druckt den Wert von **$x** multipliziert mit 7. Nach dem Ende dieses Programms sehen Sie das 1x7 auf der Zeichenfläche.
Als Standard ist die Schrittweite einer Schleife auf 1 eingestellt, für einen anderen Wert benutzen Sie:

```
von variable = zahl bis zahl schritt zahl { ... }
```

4.5.7 Verlassen einer Schleife

abbrechen

```
abbrechen
```

Beendet die laufende Schleife sofort. Die Ausführung des Programms wird mit der ersten Anweisung nach der Schleife fortgesetzt.

4.5.8 Programmausführung abbrechen

ende

```
ende
```

Beendet die Ausführung des Ihres Programms.

4.5.9 Zusicherungen zur Laufzeit überprüfen

zusichern

```
zusichern boolescher wert { ... }
```

Zusicherungen können als Begründung der Fehlerfreiheit des Programms oder einer Eingabe verwendet werden.

```
$in = frage "In welchem Jahr sind Sie geboren?"
# das Jahr muss positiv sein
zusichern $in
> 0
```

4.6 Schreiben Sie Ihre eigenen Befehle mit „lerne”

lerne ist ein besonderer Befehl, weil Sie damit Ihre eigenen Befehle erzeugen können. Ihre eigenen Befehle können einen Eingabewert erfordern und einen Ausgabewert zurückgeben. Dieses Beispiel zeigt, wie Sie einen neuen Befehl schreiben:

```
lerne kreis $x {
  wiederhole 36 {
    vorwärts $x
    linksdrehen 10
  }
}
```

Der neue Befehl hat den Namen **kreis**. **kreis** braucht einen Eingabewert, eine Zahl als Argument, um die Größe des Kreises festzulegen. **kreis** gibt keinen Ausgabewert zurück. **kreis** kann jetzt wie ein normaler Befehl benutzt werden, wie das folgende Beispiel zeigt:

```
lerne kreis $X {
  wiederhole 36 {
    vorwärts $X
    linksdrehen 10
  }
}

gehe 200,200
kreis 20

gehe 300,200
kreis 40
```

Im nächsten Beispiel wird ein Befehl mit Rückgabewert gezeigt.

```
lerne fakultät $x {
  $r = 1
  von $i = 1 bis $x {
    $r = $r * $i
  }
  zurück $r
}

drucke fakultät 5
```

In diesem Beispiel wird ein neuer Befehl mit dem Namen **fakultät** erstellt. Wenn der Eingabewert des Befehls **5** ist, wird **5*4*3*2*1** zurückgegeben. Mit dem Befehl **zurück** wird der Rückgabewert bestimmt und die Ausführung kehrt wieder zum Aufruf dieses Befehls zurück.

Befehle können auch mehr als zwei Eingaben haben. Im nächsten Beispiel sehen Sie einen Befehl zum Zeichnen eines Rechtecks.

```
lerne rechteck $x, $y {
  vorwärts $y
  rechtsdrehen 90
  vorwärts $x
  rechtsdrehen 90
  vorwärts $y
  rechtsdrehen 90
  vorwärts $x
  rechtsdrehen 90
}
```

Dann können Sie **`rechteck 50, 100`** benutzen und die Schildkröte zeichnet ein Rechteck auf die Zeichenfläche.

Kapitel 5

Erläuterung der Begriffe und Abkürzungen

In diesem Kapitel finden Sie die Erläuterung der oft „unbekannten" Begriffe, die in diesem Handbuch verwendet werden.

Grad

Grad ist die Einheit, um Winkel oder Drehungen zu messen. Eine ganze Drehung sind 360 Grad, eine halbe Drehung 180 Grad und eine viertel Drehung 90 Grad. Die Befehle **`links drehen`**, **`rechtsdrehen`** und **`richtung`** brauchen eine Eingabe in Grad.

Eingabe- und Rückgabewerte der Befehle

Einige Befehle brauchen eine Eingabe, einige geben einen Wert zurück und andere brauchen eine Eingabe *und* geben einen Wert zurück und einige Befehle brauchen weder Eingabewerte noch geben sie einen Wert zurück.

Einige Beispiele für Befehle, die nur Eingabewerte brauchen, sind:

```
vorwärts 50
stiftfarbe 255,0,0
drucke "Hallo!"
```

Der Befehl **`vorwärts`** braucht **`50`** als Eingabe. **`vorwärts`** braucht diesen Wert, um zu wissen, um wie viele Pixel es vorwärts gehen soll. **`stiftfarbe`** braucht einen Farbcode als Eingabe und **`drucke`** braucht eine Zeichenfolge (einen Text) als Eingabe. Beachten Sie, das die Eingabe auch eine Variable sein kann, wie das folgende Beispiel zeigt:

```
$x = 50
drucke $x
$str = "Hallo!"
drucke $str
```

Nun einige Beispiele von Befehlen mit Rückgabewerten:

```
$x = frage "Bitte geben Sie irgendwas ein und drücken dann OK ... Danke ↩
    !"
$r = zufall 1,100
```

Der Befehl **`frage`** braucht eine Zeichenfolge als Eingabe und gibt die eingegebene Zahl oder Zeichenfolge wieder zurück. Wie Sie sehen, wird die Ausgabe von **`frage`** in der Variablen **`x`** gespeichert. Der Befehl **`zufall`** gibt auch einen Wert zurück, in diesem Fall eine

Zahl zwischen 1 und 100. Der Zufallswert wird wieder in einer Variablen namens **r** gespeichert. Beachten Sie, das die Variablen **x** und **r** in diesem Beispiel nicht weiter verwendet werden.
Es gibt außerdem Befehle, die weder Eingabewerte benötigen noch einen Wert zurückgeben. Dazu einige Beispiele:

```
lösche
stifthoch
```

Intuitive Hervorhebung

Dies ist eine Eigenschaft von KTurtle, die das Schreiben und Lesen von Quelltexten erleichtert. Mit intuitiver Hervorhebung wird der Text, den Sie schreiben, farbig dargestellt, sodass Sie den Typ des Textes erkennen. In der folgenden Liste finden Sie die verschiedenen Typen von Quelltext und die zugehörige Farbe im Editor.

Normale Befehle	dunkelblau	Die normalen Befehle werden hier beschrieben.
Befehle zur Programmsteuerung	schwarz (fett)	Diese besonderen Befehle kontrollieren die Ausführung des Quelltextes, mehr dazu finden Sie hier.
Kommentare	grau	Kommentarzeilen beginnen mit dem Kommentarzeichen (#). Diese Zeilen werden bei der Ausführung des Quelltextes nicht beachtet. Mit Kommentaren kann der Programmierer seinen Quelltext erläutern oder damit zeitweise den Quelltext von der Ausführung ausschließen.
Klammern {, }	dunkelgrün (fett)	Klammern brauchen Sie, um Teile des Quelltextes in Gruppen zusammenzufassen. Klammern werden häufig zusammen mit Kontrollanweisungen benötigt.
Der Befehl lerne	hellgrün (fett)	Mit dem Befehl lerne schreiben Sie neue Befehle.
Zeichenfolgen	rot	Zeichenfolgen haben am Anfang und Ende doppelte Anführungszeichen (″).
Zahlen	dunkelrot	Einfach nur Zahlen.
Boolesche Werte	dunkelrot	Es gibt nur zwei boolesche Werte, wahr und falsch.

Variablen	lila	Beginnt mit den Zeichen „$“ und kann Zahlen, Zeichenfolgen und boolesche Werte enthalten.
Mathematische Operatoren	grau	Es gibt folgende mathematischen Operatoren: **+**, **−**, *****, **/** und **^**.
Vergleichsoperatoren	hellblau (fett)	Es gibt folgende Vergleichsoperatoren: **==**, **!=**, **<**, **>**, **<=** und **>=**.
Boolesche Operatoren	pink (fett)	Es gibt folgende Boolesche Operatoren: **und**, **oder** und **nicht**.
normaler Text	schwarz	

Tabelle 5.1: Verschiedene Typen von Quelltexten und deren Farbe

Pixel

Ein Pixel ist ein Punkt auf dem Bildschirm. Wenn Sie Ihren Monitor aus der Nähe anschauen, sehen Sie die Pixel. Alle Bilder auf dem Bildschirm sind aus diesen Pixeln zusammengesetzt. Ein Pixel ist die kleinste Einheit, die auf dem Bildschirm dargestellt werden kann.

Viele Befehle brauchen als Eingabe eine Anzahl von Pixeln. Diese Befehle sind: **vorwärts**, **rückwärts**, **gehe**, **gehex**, **gehey**, **papiergröße** und **stiftbreite**.

In früheren Versionen von KTurtle war die Zeichenfläche ein Rasterbild, in den aktuellen Versionen ist es eine Vektorzeichnung. Daher kann die Ansicht vergrößert und verkleinert werden, deshalb muss ein Pixel nicht notwendigerweise in einen Punkt auf dem Bildschirm umgerechnet werden.

RGB Kombinationen (Farbcodes)

RGB Kombinationen braucht man zur Bestimmung von Farben. Das „R” bedeutet „rot”, das „G” „grün” und das „B” bedeutet „blau”. Ein Beispiel für eine RGB Kombination lautet **255,0,0**, der erste Wert („rot”) ist 255 und die anderen sind 0, also steht dies für ein kräftiges Rot. Jeder Wert einer RGB Kombination muss zwischen 0 und 255 liegen. Hier eine Liste von oft gebrauchten Farben:

0,0,0	schwarz
255,255,255	weiß
255,0,0	rot
150,0,0	dunkelrot
0,255,0	grün
0,0,255	blau
0,255,255	hellblau
255,0,255	pink
255,255,0	gelb

Tabelle 5.2: Oft gebrauchte RGB-Kombinationen

Zwei Befehle brauchen eine RGB-Kombination als Eingabe: Das sind **papierfarbe** und **stiftfarbe**.

Grafiksymbol

Ein Grafiksymbol ist ein kleines Bild, dass auf dem Bildschirm bewegt werden kann. Die Schildkröte ist zum Beispiel so ein Symbol.

ANMERKUNG

In dieser Version von KTurtle kann die Schildkröte nicht durch ein anderes Symbol ersetzt werden. In späteren Versionen von KTurtle wird dies möglich sein.

Kapitel 6

Übersetzerhandbuch für KTurtle

Wie Sie vielleicht schon wissen, kann die in KTurtle verwendete Programmiersprache TurtleScript übersetzt werden. Damit ist es für Anfänger einfacher, die Grundlage des Programmierens zu verstehen.

Wenn Sie KTurtle in eine neue Sprache übersetzen, werden Sie feststellen, dass zusätzlich zu den Nachrichten der Benutzeroberfläche die Befehle der Programmiersprache, die Beispiele und die Fehlermeldungen in den üblichen `.pot`-Dateien enthalten sind, die in KDE für die Übersetzung benutzt werden. Alles wird mit dem bekannten Verfahren in KDE übersetzt, trotzdem sollten Sie lernen, wie diese Nachrichten übersetzt werden und wie die Kommentare zu den Nachrichten für Übersetzung zu verstehen sind..

Weitere Informationen zur Übersetzung finden sie unter https://edu.kde.org/kturtle/translator.php. Vielen Dank für Ihre Mitarbeit, die Übersetzung ist für KTurtle sehr wichtig.

Kapitel 7

Danksagungen und Lizenz

Kapitel 8

Index