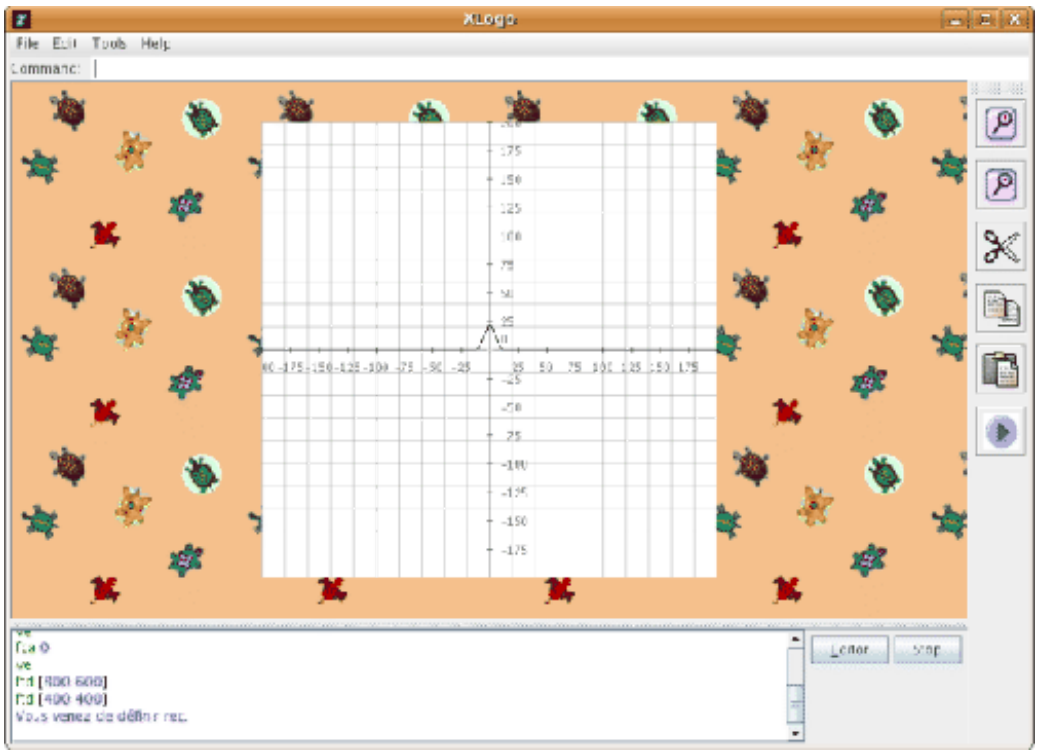
Benutzer-Handbuch

Französisches Original: Loïc Le Coq

Übersetzt von Michael Malien

Web: `http://xlogo.tuxfamily.org`



Inhaltsverzeichnis

Inhaltsverzeichnis

Inhaltsverzeichnis

XLogo Handbuch

1 Einführung
erzeugt am 28.07.2008 11:42

1 Einführung

1.1 XLogo und die Sprache Logo

- Logo ist eine in den 70ern von Seymour Papert entwickelte Computersprache. Es ist eine ausgezeichnete Sprache, um das Programmieren zu lernen, und bietet Dinge wie **Schleifen, Bedingungen, Prozeduren, etc**. Der Benutzer kann eine "Schildkröte" genanntes Objekt mit Kommandos so einfach wie mit `vorwärts`, `zurück`, `rechts` und so weiter bewegen. Bei jedem Schritt hinterläßt die Schildkröte eine Spur hinter sich, wodurch Zeichnungen erzeugt werden. Zudem zählen Operationen auf Wörter und Listen dazu, was Logo eigentlich ausmacht.

    - Geben Sie zum Beispiel `vorwärts 100 rechts 90` ein, wird der Igel `100` Schritte vor gehen, und sich dann um `90` Grad nach rechts drehen.
    - Dieser sehr intuitive graphische Ansatz macht Logo zu einer idealen Sprache für Anfänger, und besonders leicht für Kinder!

- XLogo ist ein in Java geschriebener Logo−Interpreter. Er unterstützt gegenwärtig sieben Sprachen (Französisch, Englisch, Arabisch, Spanisch, Portugiesisch, Esperanto und Deutsch, und wird unter der GPL lizenziert. Dieses Programm ist deswegen frei erhältlich. (Lizenz in Deutsch)

    - XLogo arbeitet zusammen mit:
        - ◊ Windows (95, 98, 2000, XP)
        - ◊ LINUX (Debian und Mandrake 9.*, 10.*)
        - ◊ MacOSX (10.4 Tiger)

- **Anmerkungen des Übersetzers: Schreibweisen in der deutschen Version**

    - Der Einfachheit wegen wird in der deutschen Version das Wort **Igel** anstelle von **Schildkröte** geschrieben.
    - Überhaupt sind die meisten Grundwörter angelehnt an die aus MSW/FMSLogo. So fühlen sich Anfänger automatisch auch in dem Dialekt heimisch.
    - In XLogo heißen die Grundwörter auch **Primitive**, um sie von den Kommandos in der Kommandozeile zu unterscheiden, die sich aus mehreren Grundwörtern zusammensetzen können.

1.2 Java

- XLogo ist in Java geschrieben.

    - Bei Java handelt es sich um eine Programmiersprache mit dem Nutzen, systemübergreifend zu sein. Deswegen wird XLogo auf Linux−, Windows− oder MacOS−Maschinen ohne Probleme laufen.
    - **WICHTIG**: Um eine in Java geschriebene Anwendung laufen zu lassen, müssen Sie die Java−Laufzeit−Umgebung (JRE) auf Ihrem Computer installieren. Diese ist frei und kann geladen werden von: http://java.sun.com/javase/downloads/index.jsp (Empfohlene, neueste Version)
    - Für ältere JRE−(1.4.1−) Versionen und für ältere Apple Macs: http://java.sun.com/products/archive/j2se/1.4.1_07/index.html (älteres JRE)

Erinnern Sie sich daran, die Ihrem Betriebssystem entsprechende JRE zu wählen.

Auf der Download−Seite gibt es eine Version von XLogo für die Verwendung mit JRE 1.4.1. Diese sollte sehr stabil laufen. Allerdings würde ich jedem dazu raten, aktuelle JRE− und XLogo−Versionen zu benutzen.

XLogo Handbuch (siehe Quelle)

Übersetzt von Michael Malien	Erzeugt durch Txt2Tags
Geändert am: 07/28/08 11:42:32	am 2008−07−28 um 12:49:34

XLogo Handbuch

2 Schnittstellen−Merkmale
geändert am 28.07.2008 11:35

2 Schnittstellen−Merkmale

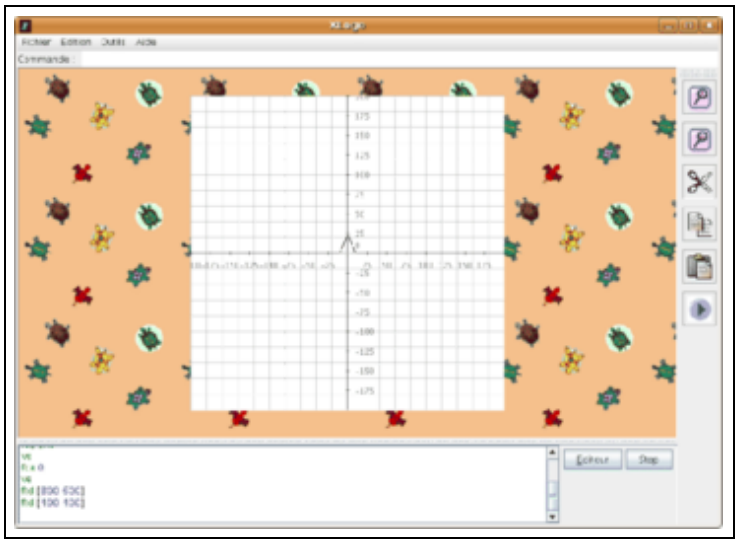
2.1 Erster Programmlauf

Starten Sie XLogo (oder wenn Sie die Datei .xlogo gelöscht haben, siehe Abschnitt 7.1), wird ein Dialogfeld Sie nach Ihrer Sprache fragen.

Dann können Sie die Sprache in dem Dialogfenster unter **Hilfsmittel−Einstellungen** ändern (siehe Abschnitt 3.3).

2.2 Das Hauptfenster

- Entlang der oberen Fensterhälfte gibt es das übliche Menü: **Datei**, **Bearbeiten**, **Hilfsmittel** und **Hilfe**
- Gleich darunter befindet sich die **Kommandozeile**, wo Logoanweisungen eingegeben und ausgeführt werden.

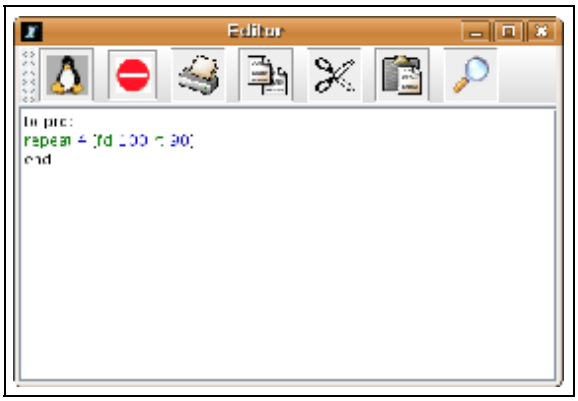


- In der Mitte vom Fenster findet sich die **Zeichenfläche**.
- Unten steht die **Befehlsgeschichte**, welche jedes eingegebene Kommando zeigt und die dazugehörige Antwort. Um schnell ein Kommando wieder aufzurufen, das schon einmal eingegeben worden ist, gibt es zwei Möglichkeiten: Sie können entweder auf das alte Kommando in der Geschichte klicken, oder Sie können einige Male den

oberen Rollpfeil klicken bis das gewünschte Kommando erscheint. Die Rollpfeile erlauben es Ihnen in der Tat nach oben und unten durch alle Kommandos zu navigieren, die Sie schon einmal eingegeben haben (sehr praktisch).

- Rechts von der Befehlsgeschichte sind zwei Schaltflächen: **EDITOR** und **STOP**.
    - ♦ Schaltfläche **STOP** unterbricht die Ausführung des Programms.
    - ♦ Schaltfläche **EDITOR** öffnet den Editor.

2.3 Der Editor



- Es gibt **drei Wege, den Editor zu öffnen**:
    - ♦ Geben Sie ed auf der Kommandozeile am oberen Teil vom Fenster ein. Der Editor wird sich dann öffnen, um alle schon definierten Prozeduren zu zeigen. Wenn Sie nur bestimmte Prozeduren bearbeiten wollen, geben Sie ein:

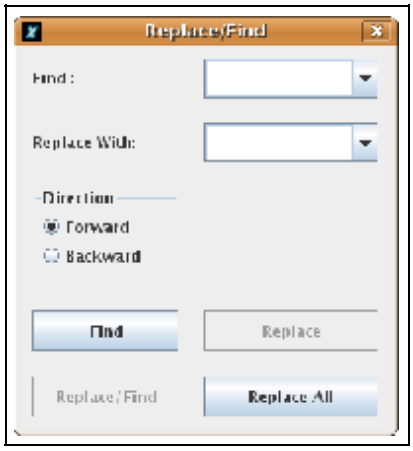
      ```
      ed [Prozedur_1 Prozedur_2 .. ]
      ```

- Drücken Sie auf die **Editor**−Schaltfläche in dem Hauptfenster.
- Benutzen Sie die die Tastaturabkürzung **Alt + E**

- **Dies sind die unterschiedlichen Schaltflächen, die Sie im Editor finden werden**:

Schaltfläche	Erklärung
	Speichere die im Editor gemachten Änderungen und schließe ihn dann. Es ist diese Schaltfläche, die Sie jedes Mal drücken müssen, um neu eingegebene Operationen anzuwenden. Wenn Sie wollen, können Sie das Tastaturabkürzel **ALT + Q** benutzen.
	Verlasse den Editor ohne eine der dort gemachten Änderungen zu übernehmen. Sie können auch die Abkürzung **ALT + C** benutzen.
	Drucke den Inhalt des Editors
	Kopiere den ausgewählten Text in die Zwischenablage

	Schneide den ausgewählten Text aus und kopiere ihn in die Zwischenablage
	Füge den ausgewählten Text aus der Zwischenablage ein
	Öffne ein Finde/Ersetzen−Dialogfeld für den Prozedur−Editor



 Unterhalb vom Editor erlaubt ein Textfeld dem Benutzer ein Hauptkommando zu definieren. Dieses Kommando ist die Haupt−Instruktion zum Starten des Programms. Mit der Play−Taste in der Werkzeugleiste im Hauptfenster kann es gestartet werden. Es wird gespeichert und zurück geladen, wenn der Editorinhalt im Logoformat gespeichert wird.

WICHTIG:

- Beachten Sie, dass ein Klicken auf die Schaltfläche (x) in der Fenster−Titelzeile keine Wirkung haben wird! Nur die zwei Haupt−Schaltflächen werden Ihnen erlauben, den Editor zu verlassen.
- Zum Löschen einer oder mehrerer unerwünschter Prozeduren benutzen Sie die Befehle `vergesse` und `vergessealles`. Oder Sie gehen in die Menüleiste: **Hilfsmittel−Lösche Prozeduren**.

2.4 XLogo beenden

Zum Verlassen von XLogo können Sie **Beende** im **Menü Datei** wählen oder klicken Sie auf die Schließe−Schaltfläche in der Fenster−Titelzeile. Ein Dialogfeld wird Sie fragen, ob Sie wirklich aufhören wollen.

XLogo Handbuch (siehe Quelle)

Übersetzt von Michael Malien	Erzeugt durch Txt2Tags
Geändert am: 07/28/08 11:35:46	am 2008−07−28 um 12:14:42

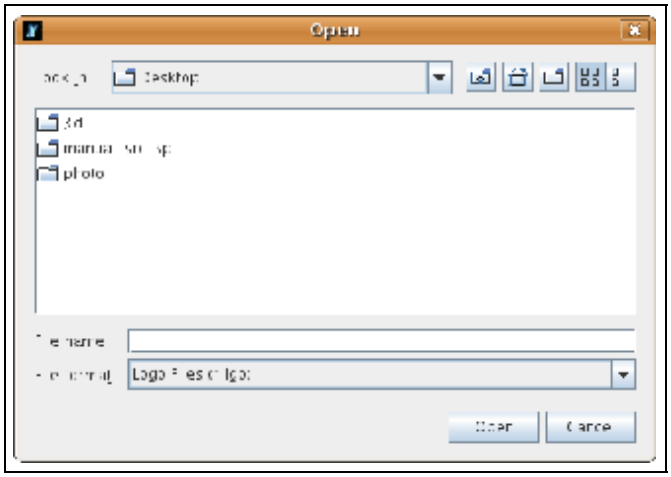
XLogo Handbuch

3 Menüaufbau
geändert am 28.07.2008 11:36

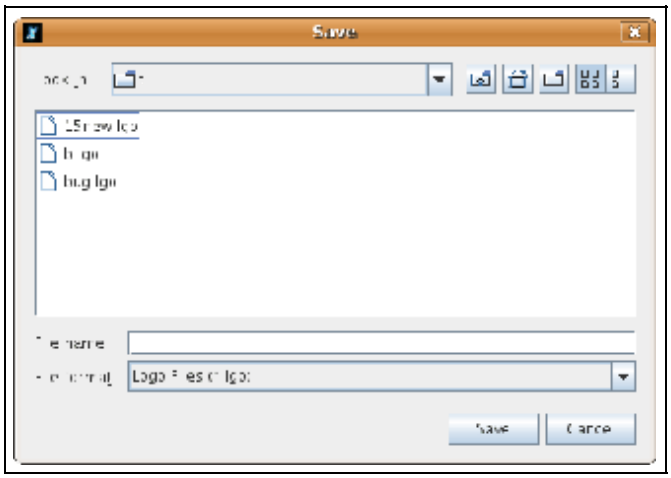
3 Menü−Optionen

3.1 Das Menü Datei

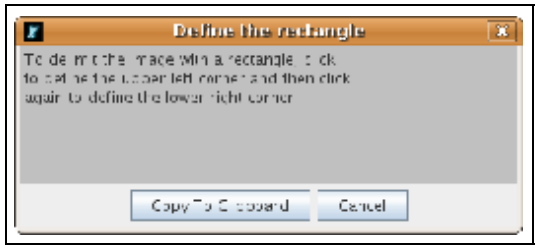
- Datei / Neu: lösche alle Prozeduren und Variable. Sie schaffen eine neue Arbeitsumgebung.
- Datei / Öffne...: öffne eine früher abgespeicherte Logodatei.



- Datei / Speichere unter...: speichere die aktuellen Prozeduren unter einem bestimmten Namen.



- Datei / Speichere: speichere die Prozeduren in der aktuellen Datei.



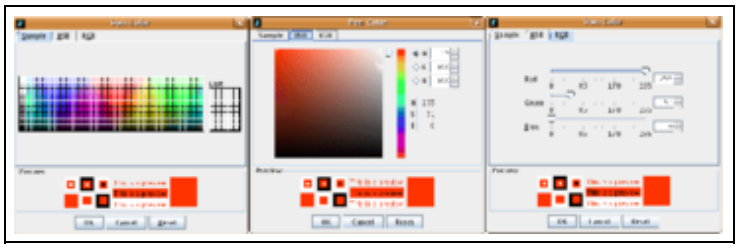
- Datei / Bild / Kopiere in die Zwischenablage: kopiere das Bild in die Zwischenablage. Wie für das Drucken und das Aufzeichnen können Sie einen Bildbereich wählen. Diese

Funktion arbeitet sehr gut unter Windows. Andererseits funktioniert es nicht unter Linux, da die Zwischenablage dort ein anderes Verhalten zeigt. Nicht getestet unter Mac.
- Datei / Bild / Speichere unter...: speichere das Bild im jpg− oder png−Format. Wenn Sie nur einen Teil des Bildes auswählen wollen, können Sie einen begrenzenden Kasten durch zweimaliges Klicken der zwei Punkte einer Diagonalen definieren.
- Datei / Bild / Drucke Bild : drucke das Bild. In der gleichen Weise wie oben können Sie ein zu druckendes Gebiet wählen.
- Datei / Textbereich / Speichere im RTF−Format: speichere die Befehlsgeschichte im RTF−Format (Farbe und Format werden bewahrt).
- Datei / Beenden: beende die Anwendung XLogo.

3.2 Das Menü Bearbeiten

- Bearbeiten / Ausschneide Strg−X: schneide den ausgewählten Text aus
- Bearbeiten / Kopiere Strg−C: kopiere den ausgewählten Text in die Zwischenablage.
- Bearbeiten / Einfüge Strg−V: füge den in der Zwischenablage enthaltenen Text in die Kommandozeile ein.
- Bearbeiten / Markiere Alles: markiere den ganzen Text in der Kommandozeile.

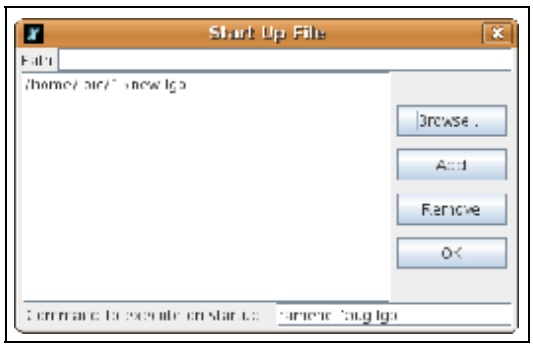
3.3 Das Menü Hilfsmittel



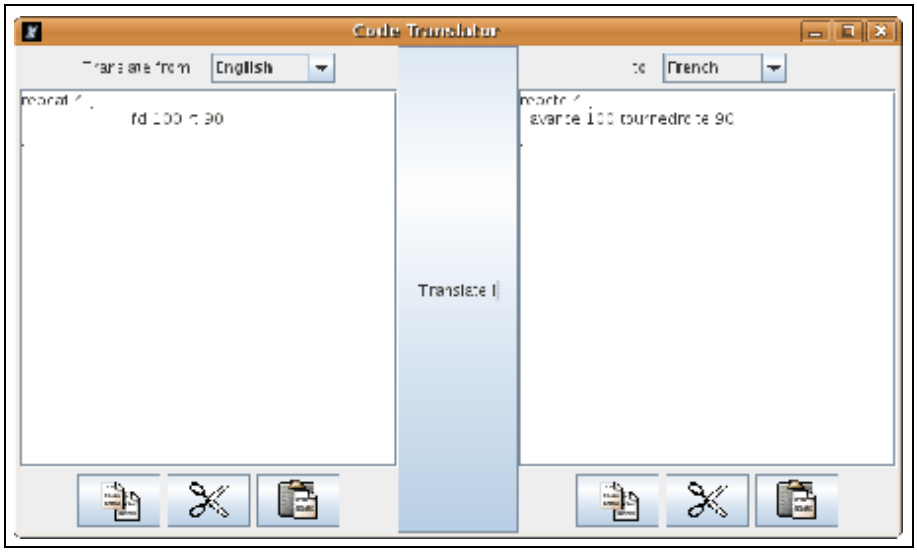
- Hilfsmittel / **Wähle Stiftfarbe**: wähle die Farbe in der der Igel schreiben soll aus einer Palette von Farben. Auch zugänglich über den Befehl `setzestiftfarbe`.

Hintergrundbild

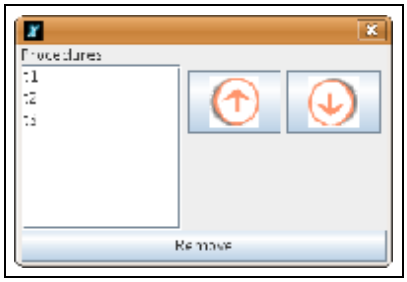
- Hilfsmittel / **Wähle Hintergrundfarbe**: setze die Farbe des Schirmhintergrunds. Zugänglich über den Befehl `setzebildfarbe` .
- Hilfsmittel / **Starte Datei...**: setze den Pfad zu einer Start−Datei. Prozeduren, die in diesen *.lgo−Dateien enthalten sind, werden zu "Pseudo−Befehlen" in XLogo und können weder editiert noch vom Benutzer verändert werden. Sie können damit Befehle definieren, die XLogo an Ihre Erfordernisse anpassen.



- Hilfsmittel / **Übersetze den Quelltext...**: übersetze eine Quelle von einer Sprache in die andere. In der Tat sehr nützlich, wenn Sie zum Beispiel eine Logo−Quelle verwenden wollen, die in einer anderen Sprache geschrieben ist.



- Hilfsmittel / **Prozeduren löschen**: lösche einige Prozeduren. Sie können die Reihenfolge aller Prozeduren im Editor ändern.

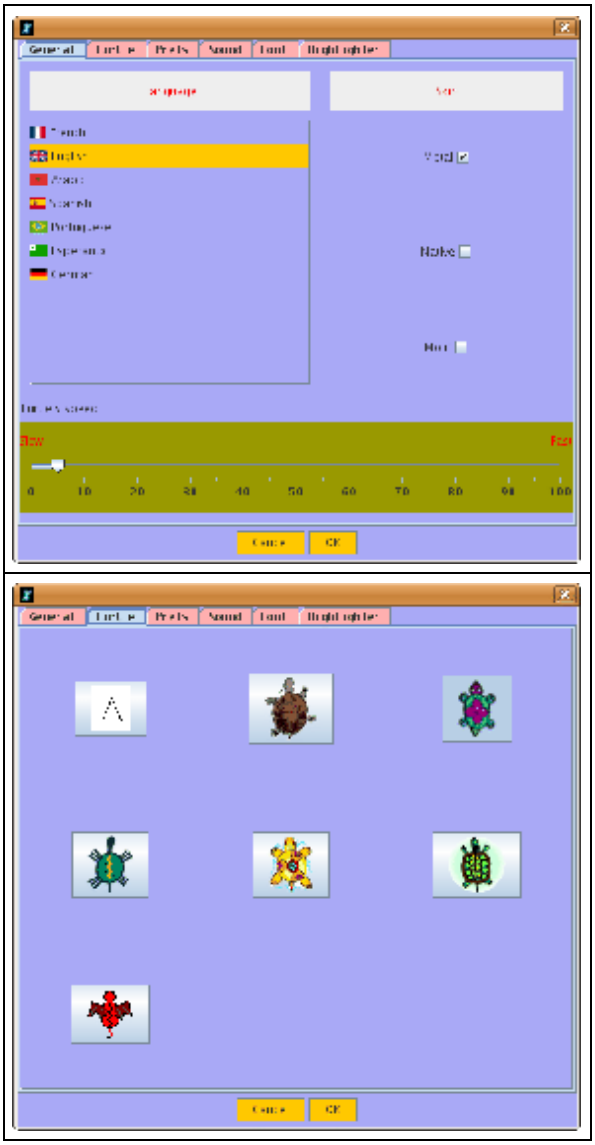


Hintergrundbild

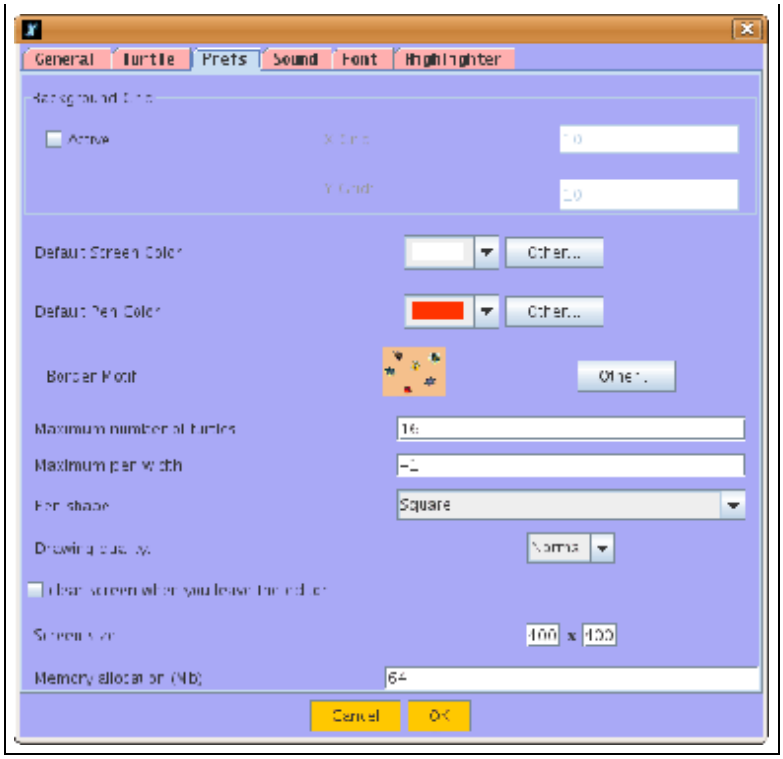
- Hilfsmittel / **Einstellungen**: öffne ein Dialogfeld, in dem Sie einige Dinge konfigurieren können:

Tab Allgemein:

- Sprache: erlaubt es die Sprache zu wählen. Beachten Sie, dass sich die Befehle in jeder Sprache unterscheiden.
- Erscheinungsbild: stellt den Stil des XLogo−Fensters ein. Verfügbar sind die Stile Metall, natives Java und Motif.
- Geschwindigkeit des Igel: Wenn Sie die Bewegungen des Igel sehen wollen, können Sie sie mit dem Einsteller auf dem ersten Tabulator verlangsamen.

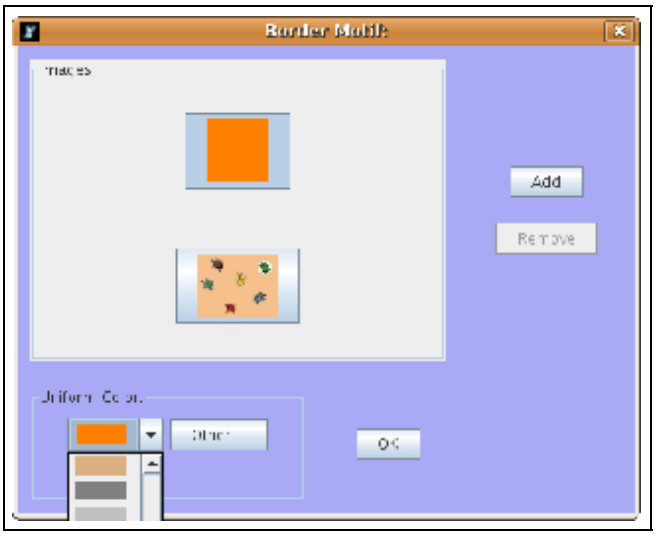


** Tab Igelwahl**: Auf dem zweiten Tab können Sie Ihren bevorzugten Igel wählen.



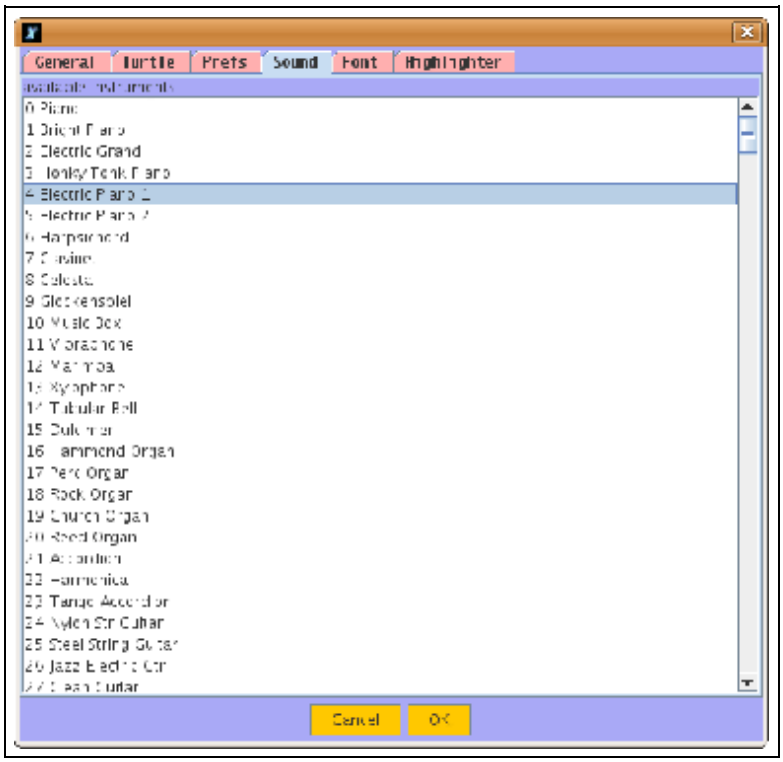
Tab Optionen: Auf dem dritten Tab gibt es viele Optionen:

- Gitter: Sie können ein Gitter auf dem Schirmhintergrund ziehen. Dazu definieren Sie die Breite und die Höhe eines Gitterquadrates.
- Achsen: Sie können eine horizontale oder vertikale Achse auf dem Hintergrund des Zeichenbereichs zeichnen. Sie können den Abstand zweier Teilstriche und die Achsenfarbe einstellen.
- Standard Hintergrundfarbe: Sie können eine Hintergrundfarbe als Standard definieren.
- Standard Schriftfarbe: Sie können eine Schreibstiftfarbe als Standard definieren.
- Randmotiv: Sie können Ihr eigenes Motiv für den Rand des Zeichenbereichs wählen. Es können ein Bild oder eine einheitliche Farbe sein.

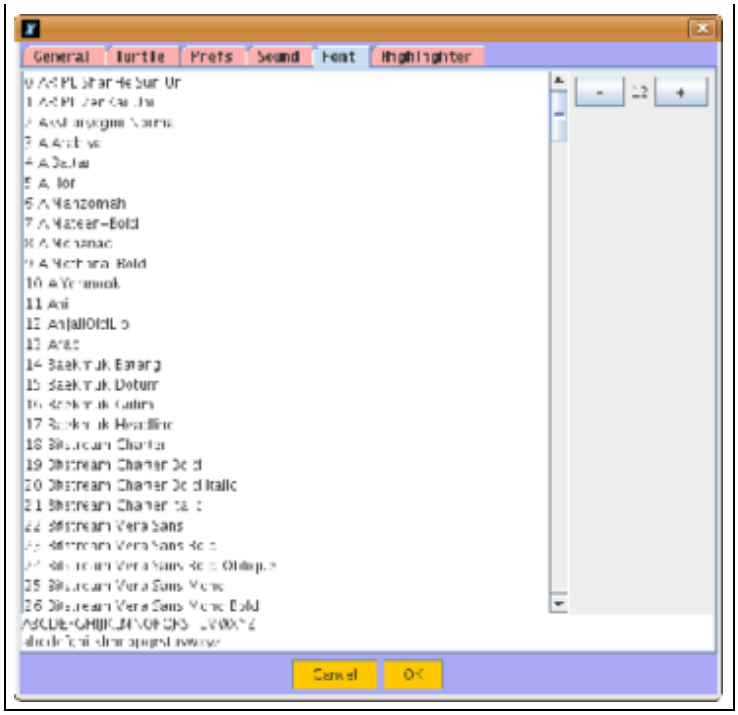


- Anzahl Igel: Sie können die maximale Anzahl von Igeln im Viele−Igel−Modus wählen.
- Maximale Stiftbreite: Sie können die maximal erlaubte Schreibstiftbreite wählen. Stellen Sie auf −1, wenn Sie diese Option nicht benutzen wollen.

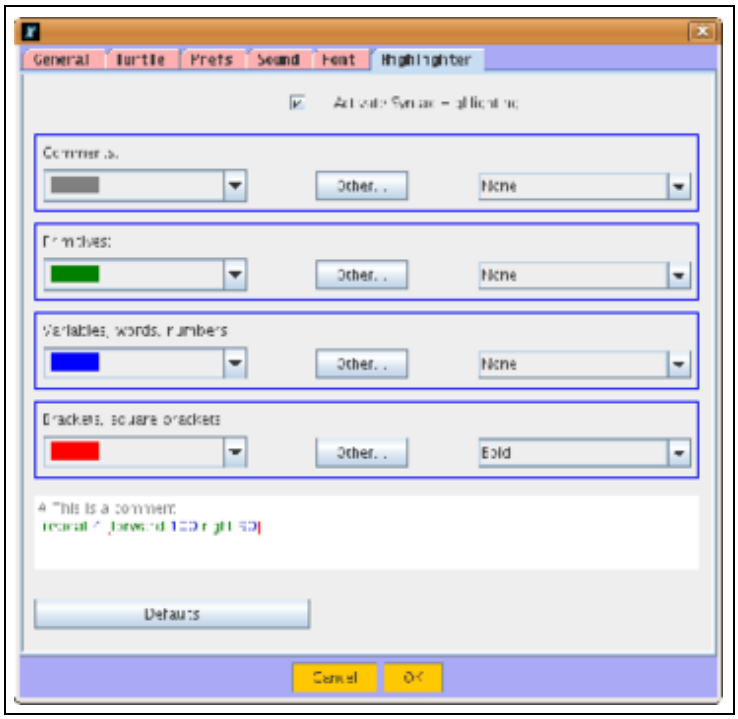
- Form des Stiftes: Sie können die Form des Schreibstifts wählen: rund oder quadratisch.
- Qualität der Zeichnung: Schließlich können Sie die Genauigkeit der Zeichenlinie wählen. In hoher Qualität werden Sie besonders nicht den Quadrateffekt haben. Vergessen Sie nicht, dass Sie durch das Vergrößern der Qualität eine geringere Ausführungsgeschwindigkeit bewirken.
- Lösche den Bildschirm beim Verlassen des Editors: Sie können wählen, ob Sie Schirm löschen wollen, wenn Sie den Editor verlassen.
- Bildschirmgröße zum Zeichnen: Sie können eine persönliche Größe für den Zeichenbereich wählen, sonst öffnet XLogo in einer Größe von 1000 mal 1000 Bildelementen. Seien Sie vorsichtig, wenn Sie die Größe des Bildes vergrößern, werden Sie die Speichergröße von XLogo erhöhen müssen. Eine Fehlermeldung wird auftauchen.
- Zugewiesener Speicher für XLogo: Sie können auch den XLogo zugewiesenen Speicher verändern, sonst wird er 64 Mo groß sein. Sie könnten ihn vergrößern, wenn Sie in einem größeren Zeichenbereich arbeiten wollen. Wenn Sie diesen Parameter ändern, müssen Sie XLogo neu starten, so dass die Änderung wirksam wird. **Seien Sie vorsichtig**, denn ein zu hoher Wert kann Ihr System verlangsamen.



Tab Sound: Auf dem vierten Tab können Sie ein Instrument für Ihre MIDI−Schnittstelle wählen. Einige Erkennungs−Probleme können auftreten, traurig... Auf diese Funktion kann mit dem Befehl `setzeinstrument` zugegriffen werden.



Tab Schriftart: Auf dem fünften Tab können Sie die Schriftart für die Schnittstelle wählen.



Tab Markierung: Sie können hier die Farbhervorhebung von Wörtern aktivieren und definieren Ihre eigenen Farben.

3.4 Das Hilfe−Menü

- Hilfe / Lizenz: zeigt die GPL−Lizenz unter der diese Software verteilt wird.

- Hilfe / Übersetzung der Lizenz: Zeigt eine Übersetzung der obigen Lizenz. Diese Übersetzung hat keinen offiziellen Charakter − diesen gibt es nur in der englischen Version, und die Übersetzung wird hier nur informativ zur Verfügung gestellt.

- Hilfe / Übersetze XLogo: Dieses Dialogfeld erlaubt für Sprache das Nachschlagen, Ändern und Vervollständigen von XLogo−Übersetzungen (Meldungen und Befehle).

Sonst können Sie, wenn Sie wollen, eine neue Übersetzung für eine neue Sprache schaffen. In jedem Fall schicken Sie mir die erzeugte Datei an loic@xlogo.tuxfamily.org

- Hilfe / über...: Das standardmäßige Ding.... und xlogo.tuxfamily.org für Ihre Lesezeichen!! o:)

XLogo Handbuch (siehe Quelle)

Übersetzt von Michael Malien	Erzeugt durch Txt2Tags
Geändert am: 07/28/08 11:36:46	am 2008−07−28 um 12:14:51

XLogo Handbuch

4 Konventionen
geändert am 28.07.2008 11:36

XLogo Konventionen

4.1 Befehle und ihre Interpretation

Dieser Abschnitt zeigt einige wichtige Punkte über die Sprache Logo selbst und speziell über XLogo auf.

Die Sprache Logo erlaubt es, gewisse Ereignisse durch interne Kommandos auszugelösen − diese Kommandos werden **Primitive** genannt. Jedes Primitiv kann eine gewisse Anzahl von Parametern haben, die **Argumente** genannt werden. Zum Beispiel nimmt das Primitiv `löschebild`, das den Schirm löscht, keine Argumente, während das Primitiv `summe` zwei Argumente nimmt.

`druckezeile summe 2 3` gibt `5` zurück.

Es gibt **drei Arten von Argumenten**:

- **Zahlen**: einige Primitive erwarten Zahlen als ein Argument: `vw 100` ist ein Beispiel.
- **Wörter**: Wörter beginnen mit " . Ein Beispiel von einem Primitiv, das ein Wort als Argument nehmen kann, ist `druckezeile`. `druckezeile "hallo` ergibt `hallo`. Beachten Sie, dass wenn Sie das " vergessen, der Interpreter eine Fehlermeldung zurückgibt. In Wirklichkeit erwartet `druckezeile` ein Argument, sonst stellt `hallo` für den Interpreter gar nichts dar, da es weder Zahl, Wort noch Liste oder eine schon definierte Prozedur ist.
- **Listen**: Diese werden zwischen Klammern definiert. Zahlen werden in einigen Fällen als ein nummerischer Wert betrachtet (z. B. `vw 100`), und in anderen als ein Wort (z. B. `druckezeile leer? 12`, das `falsch` schreibt).

Einige Primitive haben eine allgemeine Form, das heißt dass sie mit einer unbegrenzten Anzahl von Argumenten benutzt werden können. All jene Primitive stehen in der Tabelle unten:

druckezeile	summe	produkt	oder
und	liste	satz	wort

Um dem Interpreter zu sagen, dass er diese Primitive in ihrer allgemeinen Form benutzen soll, müssen wir unser Kommando in Klammern schreiben. Sehen Sie sich die Beispiele unten an:

```
druckezeile (summe 1 2 3 4 5)
----->
15

(liste [a b] 1 [c d])
----->
Ich weiss nicht, was ich machen soll mit [ [ a b ] 1 [ c d ] ]  ?

wenn (und 1=1 2=2 8=5+3) [vw 100 re 90]
----->
```

4.2 Prozeduren

Zusätzlich zu diesen Primitiven, können Sie Ihre eigenen Befehle definieren. Diese werden **Prozeduren** genannt.

Prozeduren werden vom Wort `lerne` eingeführt und enden mit dem Wort `Ende`. Achten Sie darauf, dass hier `lerne` am Anfang klein geschrieben wird und `Ende` das Ende einer Prozedur hervor hebt. Sie können erzeugt werden, in dem Sie den internen Prozedureditor benutzen. Hier ist ein kleines Beispiel:

```
lerne quadrat
  wiederhole 4 [vorwärts 100 rechts 90]
Ende
```

Diese Prozeduren können ebenso Argumente tragen. Um das zu machen, werden Variable benutzt. Eine Variable ist ein Wort, dem ein Wert zugewiesen werden kann. Hier ist ein sehr einfaches Beispiel:

```
lerne zweimal :wort
  druckezeile :wort
  druckezeile :wort
Ende

zweimal [1 2 3]
----->
1 2 3
1 2 3
```

Sehen Sie sich die verschiedenen Prozedurbeispiele am Ende vom Handbuch an.

4.3 Spezielles Zeichen \

Das spezielle Zeichen \ erlaubt es besonders Wörter zu bilden, die Leerzeichen oder Zeilenvorschubzeichen enthalten. Wenn \n benutzt wird, springt die Schreibposition zur folgenden Zeile, und \ gefolgt von einem Leerzeichen bedeutet eine Lücke im Wort. Beispiel:

```
dz "xlogo\ xlogo
----->
xlogo xlogo

dz "xlogo\nxlogo
----->
xlogo
xlogo
```

Sie können deswegen das \−Symbol nur durch das Tippen von \\ schreiben. Mit gleichem Verhalten sind die Zeichen () [] # bestimmte Begrenzer von Logo. Wenn Sie sie in einem Wort benutzen wollen, müssen Sie das Zeichen \ davor schreiben.

Alle \ Symbole werden ignoriert. Diese Bemerkung ist besonders wichtig für die Verwendung von Dateien. Um Ihren aktuellen Verzeichnispfad zu `c:\Meine Dokumente` zu setzen:

```
setzeordner "c:\\Meine\ Dokumente .
```

Bemerken Sie bitte den Gebrauch von \, um das Leerzeichen zwischen `Mein` und `Dokumente` . Wenn Sie den Doppel−Backslash vergessen, wird der Pfad als `c:My−Dokumente` gelesen und der Interpreter eine Fehlermeldung schicken.

4.4 Groß− und Kleinschreibung

XLogo unterscheidet nicht zwischen Groß− und Kleinschreibung, was Prozedurnamen und Primitive betrifft. Damit wird XLogo die Prozedur `Quadrat` , wie oben definiert, richtig übersetzen und ausführen, ob Sie nun `QUADRAT` oder `qUaDraT` in den Kommandointerpreter tippen.

Andererseits ist XLogo casesensitive bei Listen und Wörtern:

```
druckezeile "Hallo
----->
"Hallo (das Anfangszeichen H bleibt erhalten)
```

4.5 Operatoren und Syntax

Es gibt zwei Wege gewisse Befehle zu schreiben. Zum Beispiel, um 4 und 7 zu addieren, gibt es zwei Möglichkeiten: Sie können entweder das Primitiv `summe` benutzen, das zwei Argumente erwartet: `summe 4 7` , oder Sie können den Operator + benutzen: 4+7. Beide haben die gleiche Wirkung.

Diese Tabelle zeigt die Beziehung zwischen Operatoren und Primitiven:

summe	differenz	produkt	quotient
+	–	*	/
oder	und	gleich?	_
|	&	=	_

Es gibt zwei weitere Operatoren: Operator "Kleiner oder gleich" : <= Operator "Größer oder gleich" : >= Beachte: Die zwei Operatoren `|` ``` und ```& sind speziell für XLogo. Sie existieren nicht in traditionellen Logo−Versionen. Hier sind einige Beispiele zum Gebrauch:

```
dz 3+4 <= 7-1 ----> falsch

dz 3=4 | 7<=49/7 ----> wahr

dz 3=4 & 7<=49/7 ----> falsch
```

XLogo Handbuch (siehe Quelle)

Übersetzt von Michael Malien	Erzeugt durch Txt2Tags
Geändert am: 07/28/08 11:36:24	am 2008−07−28 um 12:15:00

XLogo Handbuch

5 Listen von Grundwörtern
geändert am 28.07.2008 11:35

5 Listen von Grundwörtern

5.1 Bewegung des Igel, Stift− und Farbeinstellungen

- Wie oben schon gesagt wurde, wird der Igel mittels interner Befehle kontrolliert, die **Primitive** heißen. Die folgenden Abschnitte zeigen diese Primitive:

5.1.1 Den Igel bewegen

Diese erste Tabelle zeigt die Primitive auf, die die Bewegung des Igel bestimmen.

Deutsch	Englisch	Argumente	Verwendung
vw, vorwärts	fd, forward	n : Anzahl Schritte	Bewegt den Igel n Schritte vorwärts in die Richtung in die er gerade zeigt.
rw, rückwärts	bk, back	n: Anzahl Schritte	Bewegt den Igel n Schritte rückwärts in die Richtung in die er gerade zeigt.
re, rechts	rt, right	n: Winkel	Dreht den Igel n Grad nach rechts relativ zu der Richtung in die er gerade zeigt.
li, links	lt, left	n: Winkel	Dreht den Igel n Grad nach links relativ zu der Richtung in die er gerade zeigt.
kreis	circle	R: Zahl	Zeichnet einen Kreis vom Radius R um den Igel.
arc, bogen	arc	R cap1 cap2: Zahlen	Zeichnet einen Bogen vom Radius R um den Igel. Dieser Bogen wird eingeschrieben zwischen cap1 und cap2.
heim	home	keine	Setzt den Igel auf die Anfangsposition, d.h. auf die Koordinaten [0 0] mit der Richtung nach 0 Grad.
auf	setpos, setposition	[x y] Liste zweier Zahlen	Bewegt den Igel zu den Koordinaten, die durch die zwei Zahlen in der Liste angegeben sind (x bezeichnet die x−Achse und y die y−Achse)
sx, setzex	setx	x: x−Achse	Bewegt den Igel horizontal zu dem Punkt X auf der x−Achse.
sy, setzey	sety	y: y−Achse	Bewegt den Igel horizontal zu dem Punkt Y auf der y−Achse.
sxy, setzexy	setxy	x y: x−Koordinate gefolgt von der y−Koordinate	Identisch zu `auf [x y]`
ak, aufkurs	setheading	n: Richtung	Orientiert den Igel in die angegebene Richtung. 0 bedeutet, dass der Igel nach oben zeigt.

Deutsch	Englisch	Argumente	Verwendung

igeltext	label	a: Wort oder Liste	Zeichnet das angegebene Wort oder Liste an der Position des Igel, entsprechend der Richtung in die er zeigt. Z.B.: schreibt `igeltext [Hallo Welt!]` den Satz "Hallo Welt!" wo immer der Igel ist, entsprechend seiner Lage oder Richtung.
litext, längeigeltext	labellength	a: Wort oder Liste	Ergibt die mit dem Primitiv `igeltext` in der aktuellen Schriftart zum Darstellen erforderliche Länge eines Wortes oder einer Liste.
punkt	dot	a: Liste	Der Punkt mit den Koordinaten in der Liste wird hervorgehoben (in der Stiftfarbe).|

Die zweite Tabelle zeigt die Primitive auf, die Eigenschaften des Igel einstellen. Sie steuern zum Beispiel, ob der Igel auf dem Schirm sichtbar sein soll oder in welcher Farbe er zeichnen soll, wenn er sich bewegt.

Deutsch	Englisch	Argumente	Verwendung
zi, zeigeigel	st, showturtle	keine	Macht den Igel auf dem Schirm sichtbar.
vi, versteckeigel	ht, hideturtle	keine	Macht den Igel auf dem Schirm unsichtbar.
lb, löschebild	cs, clearscreen	keine	Leert den Zeichenbereich.
wasche	wash	keine	Löscht den Zeichenbereich aber belässt den Igel an dem Ort.
reset	resetall	keine	Initialisiert die XLogo Schnittstelle mit den Standardwerten (Stift−Farbe: schwarz, Schirmfarbe: weiß, Animationsmodus: aus, Text und Graphik Schriftart: Dialog 12 Punkt, Stift−Form: Quadrat, Zeichnungsqualität: normal, Igel erlaubt: 16, Einzelschrittmodus: aus, Schirmgröße [1000 1000]) und leert die Zeichenfläche.
sa, stiftab	pd, pendown	keine	Der Igel wird eine Linie zeichnen, wenn er sich bewegt.
sh, stifthoch	pu, penup	keine	Der Igel wird eine Linie zeichnen, wenn er sich bewegt.
rd, radiere	pe, penerase	keine	Der Igel wird alle Markierungen radieren auf die er trifft.
umkehrstift	px, penreverse	keine	Senke den Stift und setze den Igel in Invertierungsmodus.
ns, normalstift	ppt, penpaint	keine	Senke den Stift und setze den Igel in Zeichnungsmodus.
sstf, setzestiftfarbe	setpc, setpencolor	a: Ganzzahl oder Liste [r g b]	Setzt die Stiftfarbe. Siehe Seite 22 .
sbf setzebildfarbe	setsc, setscreencolor	a: Ganzzahl oder Liste [r g b]	Setzt die Schirmfarbe. Siehe S.22 .

pos	pos, position	keine	Ergibt die aktuelle Position des Igel, z.B: `pos` ergibt [10 −100]
kurs	heading	keine	Ergibt the Lage oder Richtung des Igel (siehe `aufkurs`)
ri, richtung	towards	a: Liste	Die Liste muss zwei Zahlen als Koordinaten enthalten. Ergibt die Richtung, welcher der Igel folgen muß, um den Punkt zu erreichen, der in der Liste angegeben ist.

Deutsch	Englisch	Argumente	Verwendung
abst, abstand	distance	a: Liste	Die Liste muß zwei Zahlen als Koordinaten enthalten. Ergibt die Anzahl von Schritten zwischen der aktuellem Position und dem Punkt, der durch die Koordinaten in der Liste definiert ist.
sf, stiftfarbe	pc, pencolor	a: Liste	Ergibt die aktuelle Farbe des Stiftes. Die Farbe ist angegeben durch die Liste [r g b] , wo `r` die rote, `b` die blaue und `g` die grüne Komponente ist.
bf, bildfarbe	sc, screencolor	a: Liste	Ergibt die aktuelle Schirmfarbe (Hintergrund). Diese Farbe wird angegeben durch die Liste `[r g b]` , wo `r` die rote, `b` die blaue und `g` die grüne Komponente ist.
fen, fenster	window	keine	Der Igel kann sich außerhalb der Zeichenfläche bewegen, kann aber dort natürlich nicht zeichnen.
rspr, randsprung	wrap	keine	Wenn der Igel die Zeichenfläche verläßt, wird er auf der gegenüber liegenden Seite erscheinen!
perspektivisch	perspective	keine	Der Igel kann sich durch den 3D−Raum bewegen. (Siehe Abschnitt 5.1.1 für diesen Modus). Um diesen Modus zu

			verlassen, benutzen Sie einen der Primitive fenster, randsprung oder zaun
zaun	fence	keine	Der Igel ist auf die Zeichenfläche begrenzt. Wenn er dabei ist sie zu verlassen, wird eine Fehlermeldung angezeigt und die maximale Anzahl von Schritten ausgegeben, die der Igel sich bewegen kann bevor er den Zaun erreicht (bis 1 oder 2 Schritte ...).
ff, findefarbe	fc, findcolor	a: Liste	Ergibt die Farbe eines Pixel. Diese Farbe wird bestimmt durch eine [r g b] Liste, wo r die rote, b die blaue und g die grüne Komponente ist.
ssb, setzestiftbreite	setpw, setpenwidth	n: Zahl	Definiert die Dicke der Stiftspitze in Pixel. Standard ist 1. Der Stift hat eine quadratische Spitze. (Andere Formen werden in zukünftigen Versionen zur Verfügung gestellt.)
sb, stiftbreite	pw, penwidth	keine	Ergibt die Dicke der Stiftspitze in Pixel.
sstform, setzestiftform	setps, setPenShape	0 oder 1	Setzt die Stiftform. 0 Quadrat. 1 Oval
sf, stiftform	ps, penshape	keine	Ergibt die Stiftform. 0 Quadrat. 1 Oval
szq, szeichnungsqualität	setdq, setDrawingQuality	0, 1 oder 2	Setzt die Zeichnungsqualität: 0 normal, 1 hoch und 2 niedrig
zq, zeichnungsqualität	dq, DrawingQuality	keine	Ergibt die Zeichnungsqualität 0 normal, 1 hoch und 2 niedrig
sbg, setzebildgröße	setscreensize	[Breite Höhe]	Setzt die Schirmgröße auf die Dimension, die in der Liste enthalten ist, z.B. `setzebildgröße [1000 1000]`

bg, bildgröße	screensize	Liste	Ergibt die aktuelle Schirmgröße als Liste, z.B. `setzebildgröße [1000 1000] bildgröße`
sform, setzeform	setshape	n: Zahl	Sie können den bevorzugten Igel auf dem zweiten Reiter im Menü Hilfsmittel − Einstellungen setzen. Der bevorzugte Igel kann auch gesetzt werden mit `setshape`. Die Zahl n kann zwischen 0 und 6 liegen. 0 ist die dreieckige Form.
form	shape	keine	Ergibt die Zahl, die für die Form des Igel steht.
setzefontgröße	setfs, setfontsize	n: Zahl	Setzt Größe und Schriftart, wenn Sie auf dem Schirm mit dem Primitiv `igeltext` schreiben. Standardgröße der Schriftart ist 12 Punkt.

5.1.2 Ein Wort zu Farben

Farben werden in XLogo durch eine Liste dreier Zahlen `[r g b]` zwischen 0 und 255 definiert. Die Zahl `r` ist die `rote` Komponente, `b` das `Blau` und `g` das `Grün` . XLogo hat 16 vordefinierte Farben: Sie können sie einstellen, entweder mit ihrer RGB−Liste, mit Hilfe ihrer Nummer oder mit einem Primitiv. Sehen Sie sich diese Tabelle an:

Zahl	Deutsch	Englisch	R G B	Farbe
0	schwarz	black	0 0 0	
1	rot	red	255 0 0	
2	grün	green	0 255 0	
3	gelb	yellow	255 255 0	
4	blau	blue	0 0 255	
5	magenta	magenta	255 0 255	
6	cyan	cyan	0 255 255	
7	weiß	white	255 255 255	

8	grau	gray	128 128 128	
9	hellgrau	lightgray	192 192 192	
10	dunkelrot	darkred	128 0 0	
11	dunkelgrün	darkgreen	0 128 0	
12	dunkelblau	darkblue	0 0 128	
13	orange	orange	255 200 0	
14	rosa	pink	255 175 175	
15	violett	purple	128 0 255	
16	braun	brown	153 102 0	

Diese drei Anweisungen sind die gleichen:

```
setzebildfarbe orange sbildfarbe 13 sbf [255 200 0]
```

5.1.3 Animations−Modus

Es gibt zwei Primitive, die den Igel Befehle ausführen lassen ohne dass sie angezeigt werden: **`animation`** und **`stoppeanimation`**

Deutsch	Englisch	Argumente	Verwendung
anim, animation	anim, animation	keine	Sie gehen in den Animationsmodus. Der Igel zeichnet nicht mehr auf den Schirm, folgt aber der Linie. Um die Zeichnung auf dem Schirm zu aktualisieren, benutzen Sie das Primitiv `auffrischen`. Das ist sehr nützlich zum Erzeugen einer Animation oder um eine Linie schneller zu zeichnen.
stoppeanim, stoppeanimation	stopanim, stopanimation	keine	Animationsmodus ist beendet: Sie schalten zurück in den klassischen Modus. Sie können die Bewegungen des Igel auf dem Schirm sehen.
af, auffrischen	refresh	keine	Aktualisiert den Schirm im Animationsmodus: Das Bild des Zeichengebiets wird aktualisiert.

Damit Sie den Animationsmodus bemerken, erscheint ein Kamera−Icon in der Befehlsgeschichte. Wenn Sie auf das Icon klicken, wird der Animationsmodus stoppen. Das ist gleichwertig zu dem Primitiv `stoppeanimation` .	

5.1.4 Text in der Textbereichs schreiben

Diese Tabelle zeigt die Primitive auf, die die Eigenschaften des Textbereichs einstellen. Jene Primitive, die die Farbe und die Größe der Textbereichs kontrollieren, sind nur verfügbar für die Primitive **`druckezeile`** und **`schreibe`**

Deutsch	Englisch	Argumente	Verwendung
lt, löschetext	ct, cleartext	keine	Leert das Gebiet, das Kommando− und Kommentargeschichte enthält.
dz, druckezeile	pr, print	Wort, Liste oder Zahl	Zeigt das Argument in der Geschichtszone an. `pr "abcd` −−−−−−−−> abcd ; `pr [1 2 3 4]` −−−−> 1 2 3 4 ; `pr 4` −−−−−−−−−−−> 4
schreibe	write	Wort, Liste oder Zahl	Dasselbe für das Primitiv `druckezeile`, aber ohne Zeilenvorschub.
stg, setzetextgröße	setTS, setTextSize	a: Zahl	Definiert die Schriftartgröße im Kommandofenster. Nur gültig mit dem Primtiv `druckezeile`.
stf, stfarbe, setzetextfarbe	setTC, setTextColor	a: Zahl oder Liste	Definiert die Schriftfarbe im Kommandofenster. Nur gültig mit dem Primtiv `druckezeile`. Siehe Seite 22.
stn, setzetextname	setTN, setTextName	n: Zahl	Wählt die Schrift mit Zahl n, wenn Sie im Kommandofenster mit dem Primitiv `druckezeile` schreiben. Sie können die Verbindung zwischen der Nummer und der Schrift im Menü Hilfsmittel−Einstellungen auf dem Reiter Schriftart finden.
sstil, setzestil	setsty, setstyle	Liste oder Wort	Setzt das Format of the police im Textgebiet. Sie können zwischen sieben Stilen wählen: `kein`, `fett`, `kursiv`, `durchgestrichen`, `unterstrichen`, `hochgestellt`, `heruntergestellt`. Wenn Sie mehrere Stile zusammen wollen, schreiben Sie sie in einer Liste. Sehen Sie sich die Beispiele nach dieser Tabelle an.
stil	sty, style	keine	Ergibt eine Liste, welche die unterschiedlichen Stile für das Primitiv `druckezeile` enthält.

Ein paar Beispiele für das Formatieren von Text:

```
setzestil [Fett Unterstrichen] druckezeile "Hallo
```

hallo

```
sstil "Durchgestrichen schreibe [Text durchgestrichen] sstil "Kursiv schreibe "\ x sstil

<strike> gestrichen </strike> <i> x <sup> 2 </sup> </i>
```

Deutsch	Englisch	Argumente	Verwendung

fontgröße	fontsize	keine	Ergibt die Größe der Schriftart, wenn Sie mit dem Primitiv `igeltext` auf den Schirm schreiben.
setzefontname	setfn, setfontname	n: Zahl	Wählt die Nummer n der Schriftart, wenn Sie mit dem Primitiv `igeltext` auf den Schirm schreiben. Sie können die Verbindung zwischen Zahl und Schriftart im Menü Hilfsmittel, Einstellungen auf dem Reiter Schriftart finden.
fontname	fontname	keine	Ergibt eine Liste mit zwei Elementen. Das erste Element ist eine Zahl, die die Schriftart angibt, wenn Sie auf den Schirm mit dem Primitiv `igeltext` schreiben. Das letzte Element ist eine Liste, welche den Namen der Schriftart enthält.
setzetrenner	setsep, setseparation	a: Zahl	Bestimmt das Verhältnis zwischen Graphikschirm und history zone. Die Zahl a muss zwischen 0 und 1 liegen. Wenn a 1 ist, wird das Zeichengebiet den ganzen Raum einnehmen, wenn a gleich 0 ist, wird die history zone das ganze Fenster einnehmen.
trenner	sep, separation	keine	Ergibt das aktuelle Verhältnis zwischen Zeichengebiet und history zone.
gitter	grid	a, b Ganzzahl	Zeichnet ein Gitter. Jedes Quadrat hat die Dimension a und b.
stoppegitter	stopgrid	keine	Löscht das Gitter.
sgf, setzegitterfarbe	setgridcolor	Farbe	Erlaubt den Benutzer eine eigene Farbe für das Gitter zu wählen, z.B: `setzegitterfarbe rot`
gitterfarbe	gridcolor	keine	Ergibt die aktuelle Farbe des Gitter.
gitter?	grid?	keine	Ergibt wahr, wenn das Gitter gezeichnet wird,

			sonst falsch.
achsen	axis	n: Ganzzahl	Zeichnet horizontale und vertikale Achsen. Der Abstand zwischen zwei Teilstrichen beträgt n Schritte.
xachse	xaxis	n: Ganzzahl	Zeichnet nur die horizontale Achse. Der Abstand zwischen zwei Teilstrichen beträgt n Schritte.
yachse	yaxis	n: Ganzzahl	Zeichnet nur die vertikale Achse. Der Abstand zwischen zwei Teilstrichen beträgt n Schritte.
stoppeachsen	stopaxis	keine	Löscht beide Achsen.
saf, setzeachsenfarbe	setaxiscolor	Farbe	Erlaubt den Benutzer eine Farbe für die Achsen zu verwenden, z.B. `setzeachsenfarbe grün`
achsenfarbe	axiscolor	keine	Ergibt die aktuelle Farbe der Achse.
xachse?	xaxis?	keine	Ergibt `wahr`, wenn die horizontale Achse gezeichnet wird, sonst `falsch`.
yachse?	yaxis?	keine	Ergibt `wahr`, wenn die vertikale Achse gezeichnet wird, sonst `falsch`.
zoom	zoom	a	Zoome den Zeichenschirm. Die Zahl a stellt die Skalierung zur originalen Bildgröße dar, wie festgelegt in den Einstellungen.
fenstergröße	zonesize	keine	Ergibt eine Liste aus vier Zahlen. Diese Ganzzahlen sind die Koordinaten der linken oberen Ecke der Zeichenzone und die Koordinaten der rechten unteren Ecke.
nricht, nachricht	msg, message	a: Liste	Zeigt die Meldung der Liste in einer Dialogbox, das Programm stoppt bis der Benutzer den Button "OK" klickt.

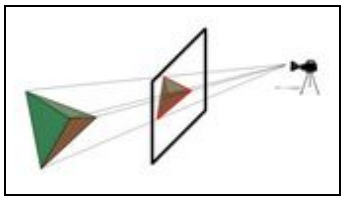
5.1.5 Der Igel in 3D

Beginnend mit Version 0.9.92 kann unser Igel die Ebene verlassen und sich im 3D−Raum bewegen. Wir benutzen das Primitiv `perspektivisch`, um dorthin zu schalten. Willkommen in der 3D−Welt!

Die perspektivische Projektion

Um einen 3D−Raum in einer 2D−Ebene darzustellen, benutzt XLogo eine perspektivische Projektion. Eine Kamera sieht auf die 3D−Szene und angezeigt wird das Bild des Projektionsschirms. Hier ist ein kleines Schema, das dies erklären soll:

Einige Primitive erlauben die Kameraposition zu setzen, der Projektionsschirm liegt auf halber Entfernung zur Kamera.



Orientierung in 3D−Welt verstehen

In der Ebene wurde die Orientierung des Igels nur durch seinen Kurs (**`kurs`**) definiert. In der 3D−Welt ist die Orientierung des Igels durch drei Winkel gegeben:

- **`Rollwinkel`**: Die Steigung des Igel um die Achse Oy.
- **`Neigung`**: Die Steigung des Igel um die Achse Ox.
- **`Kurs`**: Die Steigung des Igel um die Achse Oz.

Tatsächlich ist der Igel sehr ähnlich zu einem Jet, wenn er sich selbst in der 3D−Welt bewegt. Hier ist eine kleine Illustration, welche diese drei Werte darstellt:

Es erscheint zunächst sehr schwierig zu sein, aber Sie werden sehen, dass die Dinge sehr ähnlich bleiben zu den Bewegungen in der 2D−Ebene. Hier sind die Grund−Primitive für das Bewegen in der 3D−Welt:

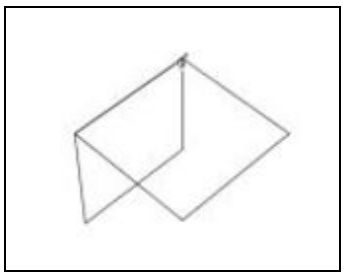
- **`vorwärts`** : Gleiches Verhalten wie in der 2D−Ebene
- **`rechts`** : Gleiches Verhalten wie in der 2D−Ebene
- **`rollerechts`** : Der Igel dreht sich um n Grad rechts um die longitudinale Achse
- **`rollelinks`** : Der Igel dreht sich um n Grad links um die longitudinale Achse
- **`rauf`** : Der Igel geht um n Grad hoch um seine transversale Achse
- **`runter`** : Der Igel geht um n Grad hinab um seine transversale Achse

Wenn wir in der 2D−Ebene ein 200 Schritte breites Quadrat zeichnen wollen, schreiben wir:

```
wiederhole 4 [vorwärts 200 rechts 90]
```

Diese Anweisungen sind auch in der 3D−Welt verfügbar und das Quadrat wird in dem Modus `perspektivisch` gezeichnet. Wenn der Igel `90` Grad herunter geht, können wir ein weiteres Quadrat zeichnen und wir erhalten:

```
löschebild
wiederhole 4 [vorwärts 200 rechts 90]
runter 90
wiederhole 4 [vorwärts 200 rechts 90]
```



Sie müssen gerade einmal ein paar Beispiele probieren, um diese Orientierung zu verstehen und werden ein Experte! Dann werden Sie erkennen, wie die drei Rotations−Primitive verbunden sind. Zum Beispiel versuchen Sie das:

```
löschebild
rollelinks 90 rauf 90 rollerechts 90
```

Das Bewegen des Igel ist äquivalent zu `links 90`.

Verfügbare Primitive im 2D− und 3D−Modus

Die folgenden Primitive sind verfügbar in der Ebene und in der 3D−Welt. Der einzige Unterschied sind die Argumente, die diese Primitive empfangen. Zum Beispiel warten die Primitive `setpos` und `setposition` immer noch auf ein Listenelement als Argument. In 3D muss diese Liste aus drei Zahlen bestehen, welche die drei Punktkoordinaten darstellen. Hier sind diese Primitive:

`kreis`	`bogen`	`heim`	`richtung`
`abstand`	`setzeposition`	`setzex`	`setzey`
`setzerichtung`	`igeltext`	`längeigeltext`	`punkt`
`pos`	`richtung`	.	.

Nur im 3D−Modus verfügbare Primitive

- **`setzexyz`** : Dieses Primitiv bewegt den Igel zum gewählten Punkt. Dieses Primitiv wartet auf drei Argumente, die die Punktkoordinaten darstellen. `setzexyz` ist sehr ähnlich zu `setzeposition` aber die Koordinaten stehen dort nicht als Liste.

  Zum Beispiel bewegt `setzexyz minus 100 200 50` den Igel zu dem Punkt x=−100; y=200; z=50

- **`setzez`** : Dieses Primitiv bewegt den Igel zu dem Punkt mit dem gültigen Wert für z. `setzez` wartet auf eine Zahl als Argument. Dieses Primitiv ist vergleichbar zu `setzex` und `setzey`.

- **`setzeorientierung`** : Setzt die Orientierung des Igel. Dieses Primitiv wartet auf eine Liste, welche drei Zahlen enthält: den Rollwinkel, die Neigung und die Kurs.

Zum Beispiel setzt `setzeorientierung [100 0 58]` den Rollwinkel `100`, die Neigung auf `0` und den Kurs auf `58` Grad.

- **`Orientierung`** : Gibt die Orientierung des Igel in einer Liste zurück: `Rollwinkel , Neigung , Kurs`. Beachten Sie die Zahlenreihenfolge. Zum Beispiel ergibt `100 , 20 , 90` dieselbe Orientierung, wenn nach der Anweisung `löschebild` von der Ursprungsposition ausgegangen wird:

```
rollerechts 100 rauf 20 rechts 90
```

Wenn Sie die Reihenfolge der Anweisungen ändern, erhalten Sie nicht die gültige Orientierung!

- **`setzerollwinkel`** : Der Igel dreht sich um die longitudinale Achse und erhält den gewählten Winkel.
- **`rollwinkel`** : Ergibt den aktuellen Wert des Rollwinkel.
- **`setzeneigung`** : Der Igel dreht sich um die transversale Achse und erhält die gewählte Neigung.
- **`neigung`** : Ergibt den aktuellen Wert der Neigung.

3D Betrachter

Ein 3D−Betrachter ist in XLogo enthalten und erlaubt es Ihre Zeichnungen in 3D darzustellen. Dieses Modul verwendet die Java3D Bibliotheken, so dass es notwendig ist Java3D voll installiert zu haben.

Hier sind die Regeln zur Verwendung des 3D−Betrachters:

Wenn wir geometrische Figuren im Zeichenbereich erschaffen wollen, müssen wir den 3D−Betrachter anweisen, welche Formen wir für zukünftige Darstellungen aufzeichnen wollen. Es ist möglich Vielecke/Vieleckflächen, Linien, Punkte und Text aufzuzeichnen.

Hier sind die Primitive:

- **`vieleckanfang`** : Die als nächstes folgenden Igelbewegungen werden gespeichert, um ein Vieleck zu erzeugen.

- **`vieleckende`** : Seit dem letzten Aufruf von `vieleckanfang` hat der Igel mehrere Ecken angesteuert. Dieses neue Vieleck wird aufgezeichnet, seine Farbe wird bestimmt durch alle Farben der Eckpunkte. Dieses Primitiv vollendet das Vieleck.

- **`linienanfang`** : Die als nächstes folgenden Igelbewegungen werden gespeichert, um eine Linie zu erzeugen.

- **`linienende`** : Seit dem letzten Aufruf von `linienanfang` hat der Igel mehrere Ecken angesteuert. Diese neue Linie wird aufgezeichnet, seine Farbe wird bestimmt durch alle Farben der Eckpunkte. Dieses Primitiv vollendet die Linie.

- **`punkteanfang`** : Die als nächstes folgenden Igelbewegungen werden gespeichert, um eine Punktemenge zu erzeugen.

- **`punkteende`** : Dieses Primitiv beendet die Punktemenge.

- **`textanfang`** : Jedesmal wenn der Benutzer mit dem Primitiv `igeltext` einen Text im Zeichenbereich anzeigt, wird er aufgezeichnet und dann im 3D−Betrachter angezeigt.

- **`textende`** : Beendet die Texteaufzeichnung.

- **`view3d vieleckansicht`** : Startet den 3D−Betrachter. Alle aufgezeichneten Objekte werden in dem neuen Fenster gezeichnet. Sie können die Kamera steuern:
  - ♦ Sie können die Szene drehen, in dem Sie auf die linke Maustaste klicken.
- Sie können die Szene übersetzen, in dem Sie auf die rechte Maustaste klicken.
- Sie können die Szene zoomen, in dem Sie das Mausrad verwenden.

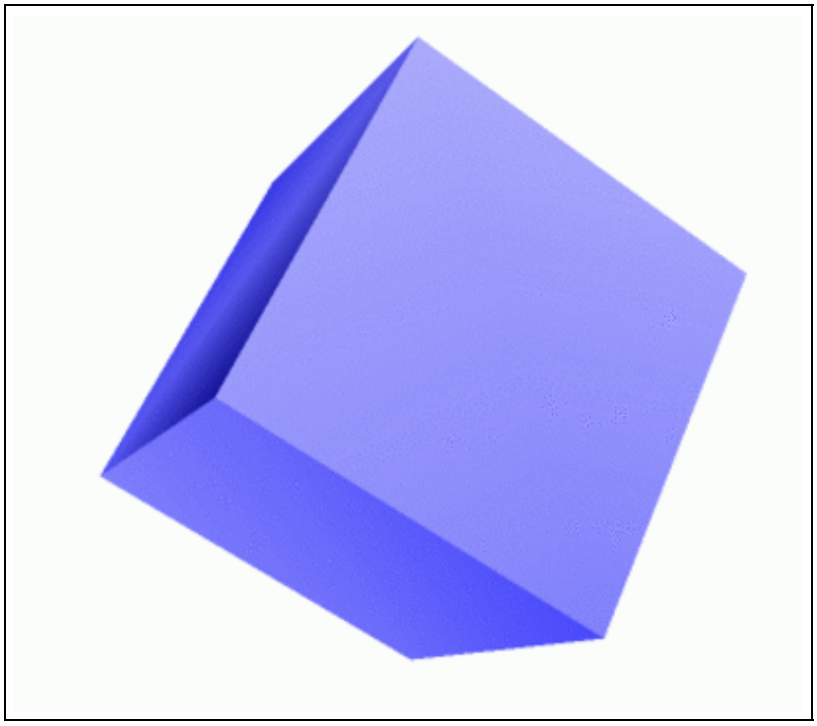
Zeichnen eines Würfels

Alle Seitenflächen bestehen aus 400 Schritten breiten Quadraten. Hier ist das Programm:

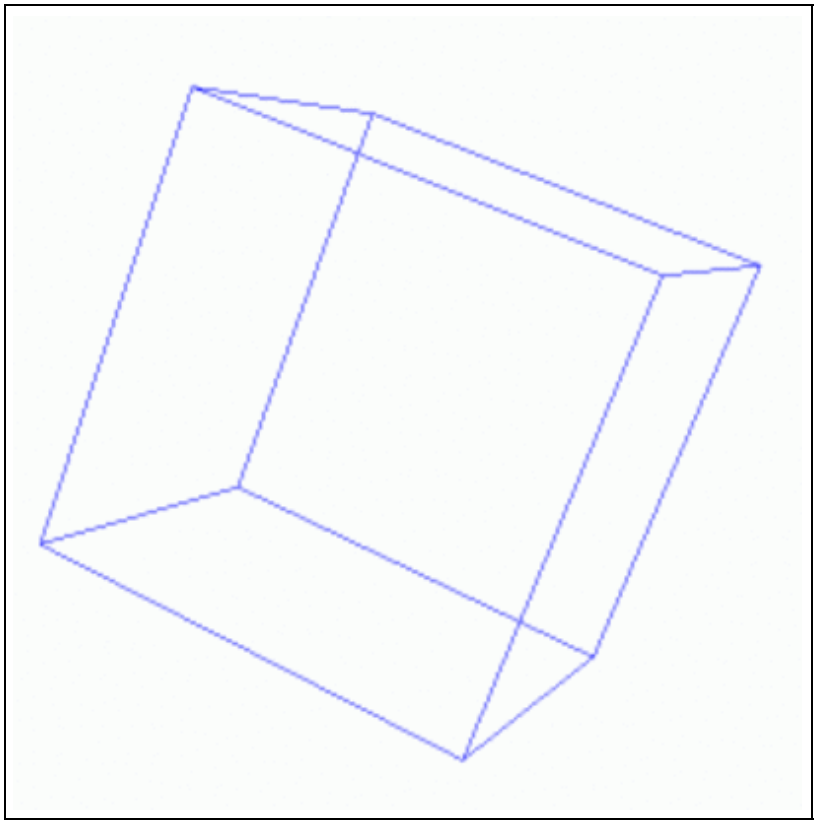
```
lerne quadrat
  # we record the vertice square
  vieleckanfang wiederhole 4[vorwärts 400 rechts 90] vieleckende
Ende

lerne einfacherWürfel
  # gelb cube
  löschebild perspektivisch setzestiftfarbe gelb
  # lateral faces
  wiederhole 4[quadrat stifthoch rechts 90 vorwärts 400 links 90 rollerechts 90 stiftab]
  # bottom face
  runter 90 quadrat rauf 90
  # upper face
  vorwärts 400 runter 90 quadrat
  # visualization
  vieleckansicht
Ende
```

Wir starten das Kommando `einfacherWürfel`:



Wenn wir in der Prozedur `Quadrat` `vieleckanfang` durch `linienanfang` und `vieleckende` durch `linienende` ersetzen:



Hätten wir `punkteanfang` und `punkteende` statt `linienanfang` und `linienende` verwendet, sollten wir auf dem Schirm nur die acht Würfelecken sehen. Diese Primitive sind sehr wichtig zum Darstellen von Punktemengen in 3D.

5.2 Arithmetische und logische Operationen

Deutsch	Englisch	Argumente	Verwendung
summe	sum	a b: Zahlen	Addiert die zwei Zahlen a und b und gibt das Ergebnis zurück, z.B.: `summe 40 60` ergibt 100
differenz	difference	a b: Zahlen	Ergibt a `-` b, z.B.: `differenz 100 20` ergibt 80
minus	minus	a : Zahl	Ergibt den negativen Wert von a, z.B.: `minus 5` ergibt −5. Siehe die Anmerkung am Ende dieser Tabelle.
produkt	product	a b: Zahlen	Ergibt das Ergebnis der Multiplikation von a und b.
teile	div, divise	a b: Zahlen	Ergibt das Ergebnis der Division von a und b, z.B.: `teile 3 6` ergibt 0.5
quotient	quotient	a b: Zahlen	Ergibt den Quotient von a und b, z.B.: `quotient 15 6` ergibt 2
rest	mod, modulo	a b: ganze Zahlen	Ergibt den Rest der Division von a und b.

runde	rnd, round	a: Zahl	Ergibt die nächste ganze Zahl zur Zahl a, z.B.: `round 6.4` ergibt 6
ganzzahl	integer	a: Zahl	Ergibt den ganzzahligen Teil der Zahl, z.B.: `ganzzahl 8.9` ergibt 8 ; `integer 6.8` ergibt 6
potenz	power	a b: Zahlen	Ergibt die Potenz 'a hoch b', z.B.: `power 3 2` ergibt 9
qw	sqrt, squareroot	n: Zahl	Ergibt die Quadratwurzel von n.
log10	log10	n : Zahl	Ergibt den Zehnerlogarithmus von n
sin, sinus	sin, sine	a: Zahl	Ergibt den Sinus von a. (a wird in Grad angegeben)
cos, cosin	cos, cosine	a: Zahl	Ergibt den Cosinus von a. (a wird in Grad angegeben)
tan	tan, tangent	a: Zahl	Ergibt den Tangens von a. (a wird in Grad angegeben)
acos, arccos	acos, arccosine	a: Zahl	Ergibt den Winkel im Bereich 0 bis 180 dessen Cosinus a ist.
asin, arcsin	asin, arcsine	a: Zahl	Ergibt den Winkel dessen Sinus a ist.
atan, arctan	atan, arctangent	a: Zahl	Ergibt den Winkel dessen Tangens a ist.
pi	pi	aucun	Ergibt die Zahl 3.141592653589793
zz, zufallszahl	random, ran	n: ganze Zahl	Ergibt eine Zufallszahl zwischen `0` und `n - 1`.
ovz, ohnevorzeichen, betrag	absolute, abs	n: Zahl	Ergibt den Absolutbetrag (den Zahlenwert ohne Vorzeichen) einer Zahl.

Wichtig: Seien Sie sorgfältig mit den Primitiven, die zwei Parameter benötigen!

```
Z.B.:
  ``setzexy a b`` , wenn b negativ ist,
  zum Beispiel bei ``setxy 200-10``
```

Der Logointerpreter wird die Operation `200-10` durchführen (d. h. es wird `10` von `200` subtrahiert). Er wird deswegen schlussfolgern, dass es nur einen Parameter (`190`) gibt, aber zwei benötigt, und wird daher eine Fehlermeldung erzeugen. Um diese Art von Problem zu vermeiden, benutzen Sie das Primitiv `minus` , um die negative Zahl zu spezifizieren:

- `setxy 200 minus 10` .

  Dies ist eine Liste von logischen Operatoren:

Deutsch	Englisch	Argumente	Verwendung
oder, eines?	or	b: boolesch	Ergibt `wahr`, wenn a oder b `wahr` sind, sonst `falsch`
und, alle?	and	b: boolesch	Ergibt `wahr`, wenn a und b `wahr` sind, sonst `falsch`

nicht	not	a :boolesch	Ergibt die Negation von a. Wenn a `wahr` ist, `falsch`. Wenn a `falsch` ist, `wahr`.

5.3 Operationen auf Listen und Wörter

Deutsch	Englisch	Argumente	Verwendung
wort	word	a b	Verbindet zwei Wörter a und b, z.B.: `dz word "a 1` ergibt a1
liste	list	a b	Ergibt eine Liste bestehend aus a und b, zum Beispiel, `liste 3 6` ergibt [3 6]. `liste "a "Liste` ergibt [a Liste]
satz	se, sentence	a b	Ergibt eine Liste bestehend aus a und b. Wenn a oder b Listen sind , dann wird jedes Element von a und b ein Element der Ergebnisliste (Quadratische Klammern werden entfernt), z.B.: `satz [4 3] "Hallo` ergibt [4 3 Hallo] ; satz [Wo sind] "Dinge ergibt [Wo sind Dinge]
me, miterstem	fput	a b: a irgendwas, b Liste	Fügt a vor dem ersten Element von Liste b ein, z.B: `me "cocoa [2]` ergibt [cocoa 2]
ml, mitletztem	lput	a b: a irgendwas, b Liste	Fügt a hinter dem letzten Element von Liste b ein, z.B.: `ml 5 [7 9 5]` ergibt [7 9 5 5]
umdrehliste	reverse	a : Liste	Kehrt die Reihenfolge der Elements in List a um, z.B.: `umdrehliste [1 2 3]` ergibt [3 2 1]
nehme, nehmewas	pick	a : ein Wort oder Liste	Wenn a ein Wort ist, gibt sie einen zufälligen Buchstaben von a zurück. Wenn a eine Liste ist, gibt sie ein zufälliges Element von a zurück.
entferne	remove	a b: a irgendwas, b Liste	Entfernt Element a von Liste b, wenn a vorkommt, z.B.: `entferne 2 [1 2 3 4 2 6 ]` ergibt [1 3 4 6]

Deutsch	Englisch	Argumente	Verwendung
element	item	a b: a ganze Zahl, b List oder Wort	Wenn b ein Wort ist, ergibt sie den Buchstaben an Position a im Wort (1 stellt den ersten Buchstaben dar.). Wenn b eine Liste ist, ergibt sie das Element an Position n in der Liste.
ol, ohneletztes	bl, butlast	a: Liste oder Wort	Wenn a eine Liste ist, ergibt sie die ganze Liste außer ihr letztes Element. Wenn a ein Wort ist, ergibt sie das Wort

			minus seinen letzten Buchstaben.
oe, ohneerstes	bf, butfirst	a: Liste oder Wort	Wenn a eine Liste ist, ergibt sie die ganze Liste außer ihr erstes Element. Wenn a ein Wort ist, ergibt sie das Wort minus seinen ersten Buchstaben.
lz, letztes	last	a: Liste oder Wort	Wenn a eine Liste ist, ergibt sie das letzte Element der Liste. Wenn a ein Wort ist, ergibt sie den letzten Buchstaben des Wortes.
erstes	first	a: List oder Wort	Wenn a eine Liste ist, ergibt sie das erste Element der Liste. Wenn a ein Wort ist, ergibt sie den ersten Buchstaben des Worts.
selem, setzeelement	setitem, replace	li1 n li2: li1 Liste, n Ganzzahl, li2 Wort oder Liste	Ersetzt Element n in der Liste li1 durch das Wort oder die Liste li2. `setzelement [a b c] 2 8` −−> [a 8 c]
fez, fügeelementzu	additem	li1 n li2: li1 Liste, n Ganzzahl, li2 Wort oder Liste	Fügt an der Position n in the List li das Wort oder die Liste li2 ein, z.B. `fügeelementzu [a b c] 2 8` −−> [a 8 b c]
zähle	count	a: Liste oder Wort	Wenn a ein Wort ist, ergibt sie die Anzahl von Buchstaben in a. Wenn a eine Liste ist, ergibt sie die Anzahl von Elementen in a.
unicode	unicode	a: Wort	Ergibt den Unicode−Wert von Zeichen "a", z.B.: `dz unicode "A` ergibt 65
zeichen	character, char	a: Zahl	Ergibt das Zeichen dessen Unicode−Wert "a" ist, z.B.: `dz zeichen 65` ergibt "A

5.4 Boolesche Funktionen

Eine boolesche Funktion ist ein Primitiv, das **`Wahr`** oder **`Falsch`** zurückgibt. Diese Primitive enden hier mit einem Fragezeichen. Sie heißen auch Prädikate (engl. predicates), vergleichbar mit den Informationsfunktionen in Excel (Anmerkung des Übersetzers).

Deutsch	Englisch	Argumente	Verwendung
wahr	true	keine	Ergibt `wahr`.
falsch	false	keine	Ergibt `falsch`.

wort?	word?	a	Ergibt `wahr`, wenn a ein Wort ist, sonst `falsch`.
zahl?	number?	a	Ergibt `wahr`, wenn a eine Zahl ist, sonst `falsch`.
ganzzahl?	integer?	a	Ergibt `wahr`, wenn a eine ganze Zahl ist, sonst `falsch`.
liste?	list?	a	Ergibt `wahr`, wenn a eine Liste ist, sonst `falsch`.
leer?	empty?	a	Ergibt `wahr`, wenn a eine leeres Wort oder eine leere Liste ist, sonst `falsch`.
gleich?	equal?	a b	Ergibt `wahr`, wenn a und b gleich sind, sonst `falsch`.

Deutsch	Englisch	Argumente	Verwendung
vorher?	before?	a b : Wörter	Ergibt `wahr`, wenn a alphabetisch vor b liegt, sonst `falsch`.
element?	member?	a b	Ergibt `wahr`, wenn b eine Liste ist und a ein Element von b ist. Ergibt `wahr`, wenn b ein Wort ist und a ein Buchstabe von b ist.
elementab	member	a b	Sucht das Element a in dieser Liste, wenn b eine Liste ist.

Hier gibt es zwei mögliche Fälle:

- Wenn a in b ist, ergibt sie eine Teilliste aller Listenelemente beginnnend mit der ersten Position von a in b.
- Wenn a nicht in b ist, ergibt sie das Wort `falsch`. Wenn b ein Wort ist, sucht sie nach einem Buchstaben a in diesem Wort. Da gibt es zwei Möglichkeiten:

- Wenn a in b ist, ergibt sie den letzten Teil von dem Wort, beginnend mit a.
- Sonst ergibt sie das Wort `falsch`. `elementab "o "cocoa` ergibt `ocoa` ; `elementab 3 [1 2 3 4]` liefert `[3 4]` |

sa? stiftab?	pd?, pendown?	irgendwas	Ergibt das Wort `wahr`, wenn der Stift abgesetzt ist, sonst `falsch`.
sichtbar?	visible?	irgendwas	Ergibt das Wort `wahr`, wenn der Igel sichtbar ist, sonst `falsch`.
grundwort?	prim?, primitive?	a: Wort	Ergibt `wahr`, wenn das Wort ein XLogo−Primitiv ist, sonst `falsch`.
proz?, prozedur?	proc?, procedure?	a: Wort	Ergibt `wahr`, wenn das Wort eine benutzerdefinierte Prozedur ist, sonst `falsch`.

5.5 Einen Ausdruck mit dem Primitiv Wenn testen

Wie in jeder Programmiersprache auch erlaubt es Ihnen auch Logo zu überprüfen, ob einer Bedingung entsprochen wird, und dann den gewünschten Code auszuführen, wenn sie `wahr` oder `falsch` ist.

Mit dem Primitiv **Wenn** ermöglichen Sie solche Tests. Hier ist die Syntax:

```
Wenn ausdruck_test [Liste1] [Liste2]
```

Wenn `ausdruck_test` wahr ist, werden die in `Liste1` eingeschlossenen Anweisungen ausgeführt.

Wenn `ausdruck_test` falsch ist, werden die Anweisungen in `Liste2` ausgeführt. Die zweite Liste ist optional.

Beispiele:

- `Wenn 1+2=3 [druckezeile "wahr][druckezeile "falsch]`
- `Wenn (erstes "XLOGO)="Y [vw 100 re 90] [dz [XLOGO fängt mit einem X an!]]`
- `Wenn (3*4)=6+6 [dz 12]`

5.6 Sich mit Prozeduren und Variablen befassen

5.6.1 Prozeduren

Prozeduren sind eine Art von "Programm". Wenn eine Prozedur aufgerufen wird, werden die Anweisungen im Körper von der Prozedur ausgeführt. Eine Prozedur wird mit dem Schlüsselwort **lerne** definiert.

```
lerne Name_von_Prozedur :v1 :v2 :v3.... [:v4 ....] [:v5 ....]
  Körper der Prozedur
Ende
```

- `Name_von_Prozedur` ist der Name, der der Prozedur gegeben wird.
- `:v1 :v2 :v3` stehen für die Variablen, die innerhalb der Prozedur benutzt werden. (lokale Variablen).
- `[:v4 ... ], [:v5...]` sind optionale Variablen, die wir der Prozedur hinzufügen können (gehen Sie nach unten für mehr Erklärungen).
- `Körper der Prozedur` stellt die Befehle dar, die ausgeführt werden, wenn diese Prozedur gerufen wird.

  Z.B.:

  ```
  lerne quadrat :s
    wiederhole 4 [vw :s re 90]
  Ende
  ```

  Die Prozedur wird `quadrat` genannt und nimmt einen s genannten Parameter. `quadrat 100` wird deswegen ein Quadrat produzieren, das eine Seitenlänge von `100` hat. (Betrachten Sie die Beispiele von Prozeduren am Ende vom Handbuch.)

Seit Version 0. 7 c, ist es möglich, Bemerkungen nach dem durch # eingeleiteten Code einzufügen.

```
lerne quadrat :s
  # diese Prozedur erlaubt es, ein Quadrat zu zeichnen dessen Seite :s ist.
  wiederhole 4 [vw :s re 90] # praktisch, nicht wahr?
Ende
```

Optionale Variablen

Es ist nun möglich, einer Prozedur von XLogo optionale Argumente hinzu zu fügen. Schauen Sie sich das Beispiel unten an:

```
lerne vieleck :n [:l 10]
  wiederhole :n [vw :l re 360/:n]
Ende

# mit diesem Befehl wird ein reguläres Vieleck gezeichnet
# 20 Seiten der Länge 10
vieleck 20
```

Während der Interpretation ist die Variable `:l` durch ihren Standardwert ersetzt worden, ich meine die `10` . Wenn wir diesen Wert ändern wollen, müssen wir die Prozedur Vieleck zwischen Klammern aufrufen, um dem Interpreter zu sagen, dass wir optionale Argumente benutzen werden.

```
# Dieser Befehl wird ein reguläres Polygon mit Länge 20 zeichnen
#−Seiten. Jede Seite ist lang 5.
(vieleck 20 5)

# Dies ist ein Quadrat mit jeder Seite der Länge 100
(Vieleck 4 100)
```

5.6.2 Das Konzept der Variablen

Es gibt zwei Arten von Variablen:

- Globale Variablen: Diese sind immer zugänglich von irgendeinem Ort im Programm.
- Lokale Variablen: Diese sind nur zugänglich in den Prozeduren, wo sie definiert werden. In dieser Logo−Version sind lokale Variable nicht zugänglich in Unterprozeduren. Am Ende der Prozedur werden die lokalen Variablen gelöscht.

5.6.3 Das Primitiv verfolge

Es ist möglich, die Arbeit eines Programms zu verfolgen, um sich die Prozeduren zeigen zu lassen, wenn sie arbeiten. Dieser Modus zeigt, dank des Primitivs `rückgabe`, was die Prozeduren ausgeben. Um diesen Modus zu verwenden, tippen Sie **`verfolge`**.

`stoppeverfolge` wird den `Verfolge`−Modus deaktivieren. Ein kleines Beispiel mit der Fakultät (siehe Seite 52).

```
Verfolge dz fak 4
```

ergibt:

```
fak 4
  fak 3
    fak 2
      fak 1
      fak ergibt 1
    fak ergibt 2
  fak ergibt 6
fak ergibt 24
24
```

Deutsch	Englisch	Argumente	Verwendung
setze	make		

		a b: a Wort, b irgendwas	Wenn die lokale Variable a existiert, weist sie den Wert b zu. Wenn nicht, erzeugt sie eine globale Variable a und weist ihr den Wert b zu. Z.B.: `setze "a 100` weist den Wert 100 der Variablen a zu.
lokal	local	a: Wort	Erzeugt eine Variable namens a. Beachte, diese wird nicht initialisiert. Um einen Wert zuzuweisen, siehe `setze`.
lokalsetze	localmake	a b: a Wort, b irgendwas	Erzeugt eine neue lokale Variable und weist ihr den Wert b zu.
def, definiere	def, define	Wort1 Liste2 Liste3	Definiert eine neue Prozedur namens Wort1, welche die Variablen in Liste2 erfordert. Liste3 enthält die Anweisungen der Prozedur. Z.B. `def "Vieleck [nb länge][wiederhole :nb [vw :länge re 360/:nb]]`

−−−> Dieses Kommando definiert eine Prozedur namens Vieleck mit zwei Variablen :nb und :länge. Diese Prozedur zeichnet ein reguläres Vieleck, wir können die Anzahl der Seiten und ihre Längen wählen. ||

Deutsch	Englisch	Argumente	Verwendung
wert	thing	a: Wort	Ergibt den Wert der Variablen :a. `wert "a` ist ähnlich wie `:a`
vg, vergesse	er, erase	a: Wort	Entfernt die Prozedur namens a.
vgv, vergessevar	kill	a: Wort	Löscht die Variable a.
vga, vergessealles	erall, eraseall	keine	Entfernt alle aktuellen Variablen und aktuellen Prozeduren.
zga, zeigealles	poall, printoutall	keine	Listet alle aktuell definierten Prozeduren.
starte	run	a :Liste	Führt die Liste von Instruktionen aus, die in Liste a stehen.
variablen	lvars, listvariables	keine	Ergibt eine Liste die alle definierten Variablen enthält.

5.6.4 Eigenschaftslisten

Nun können Sie in XLogo Eigenschaftslisten definieren. Jede Liste hat einen bestimmten Namen und enthält einige Schlüssel−Wert−Paare.

Zum Beispiel können wir eine Eigenschaftsliste namens "Auto" betrachten. Sie soll einen Schlüssel "Farbe" mit dem Wert "rot" assoziieren, und den Schlüssel "Typ" mit dem Wert "4x4".

Um diese Listen zu handhaben, können wir folgende Primitive benutzen:

- **`setzeeg`**

  Syntax: setzeeg Listenname Schlüssel Wert

  Fügt zur Eigenschaftsliste namens Listenname eine Eigenschaft hinzu. Auf den Wert kann mit dem Schlüssel zugegriffen werden. Existiert keine Eigenschaftsliste namens

Listenname, wird sie erzeugt.

- **`gebeeg`**

  Syntax: gebeeg Listenname Schlüssel

  Gibt den Wert zurück, der mit dem Schlüssel Schlüssel in der Eigenschaftsliste namens Listenname assoziiert ist. Wenn diese Eigenschaft nicht existiert oder kein gültiger Schlüssel angegeben ist, wird eine leere Liste zurückgegeben.

- **`entferneeg`**

  Syntax: entferneeg Listenname Schlüssel

  Entfernt die zusammengehörigen Schlüsselwert−Paare aus der Eigenschaftsliste Listenname

- **`egliste`**

  Syntax: egliste Listenname

  Zeigt alle Schlüssel−Wert−Paare an, die in der Eigenschaftsliste namens Listenname stehen.

Nun zurück zur Eigenschaftsliste "Auto":

```
# Füllen einer Eigenschaftsliste
setzeeg "Auto "Farbe "rot
setzeeg "Auto "Typ "Polo
setzeeg "Auto "Hersteller "VW

# Zeige einen Wert an
druckezeile gebeeg "Auto "Farbe
```

rot

```
# Zeige alle Elemente an
druckezeile egliste "Auto
Hersteller VW Farbe rot Typ Polo
```

5.7 Dateien handhaben

Deutsch	Englisch	Argumente	Verwendung
listedateien	ls, listfiles	keine	Listet den Inhalt eines Verzeichnisses. (Äquivalent zu dem ls Kommando für Linux−Benutzer und dem dir Kommando für DOS−Benutzer)
ladebild	li, loadimage	a: Liste	Ladet eine Bilddatei. Ihre obere linke Ecke wird an der Position des Igel gesetzt. Die einzigen unterstützten Formate sind .png und .jpg. Der Pfad muss relativ zum

			aktuellen Ordner angegeben werden. Z.B.: `setzeordner "C:\\meine_bilder ladebild "igel.jpg`
sordner, setzeordner	setdir, setdirectory	l: Liste	Setzt das aktuelle Verzeichnis. Der Pfad muss absolut angegeben werden. Das Verzeichnis muss mit einem Wort bezeichnet werden.
wo, wechsleordner	cd, changedirectory	m: Wort	Erlaubt es das aktuelle Verzeichnis zu wählen. Der Pfad ist relativ zum aktuellen Verzeichnis. Sie können die '..' Schreibweise verwenden, um sich auf das Eltern−Verzeichnis zu beziehen.
ord, ordner	dir, directory	keine	Ergibt das das aktuelle Verzeichnis. Das Standard Homeverzeichnis des Benutzers lautet `/home/your_login` für Linux−Benutzer, `C:\WINDOWS` für Benutzer von Windows.
speichere	save	w: Wort l: Liste	Ein gutes Beispiel das zu erklären ist: `speichere "test.lgo [proc1 proc2 proc3]` speichert in der Datei test.lgo im Verzeichnis die Prozeduren proc1, proc2 und proc3. Wenn die Erweiterung `.lgo` fortgelassen wird, wird sie standardmäßig hinzugefügt. Das Wort gibt einen relativen Pfad beginnend vom aktuellen Verzeichnis. Dieses Kommando wird nicht mit einem absoluten Pfad arbeiten.
gespeichert	saved	w: Wort	`gespeichert "test.lgo` speichert in der Datei test.lgo im aktuellen Verzeichnis die aktuell definierten Prozeduren. Wenn die Erweiterung `.lgo` fortgelassen wird, wird sie standardmäßig hinzugefügt. Das angegebene Wort gibt den relativen Pfad beginnend mit dem aktuellen

			Verzeichnis. Dieses Kommando wird nicht mit einem absoluten Pfad arbeiten.
lade	load	w: Wort	Öffnet und liest die Datei `w`. Um zum Beispiel alle definierten Prozeduren zu löschen und die Datei test.lgo" zu laden, würden Sie schreiben: `efns lade "test.lgo`. Das angegebene Wort gibt den relativen Pfad beginnend mit dem aktuellen Verzeichnis. Dieses Kommando wird nicht mit einem absoluten Pfad arbeiten.
öffnefluss	openflow	id Datei	Wenn Sie von einer Datei lesen oder in sie schreiben wollen, müssen Sie sie zuerst öffnen. Das Argument Datei" muss der Name der Datei sein. Sie müssen ein Wort geben, um die Datei im aktuellen Verzeichnis zu bezeichnen. Das id Argument ist die Zahl, die dem Fluss gegegen wird, um die Datei zu identifizieren.
listefluss	listflow	keine	Zeigt die Liste der verschiedenen offenen Flüsse mit ihren Bezeichnern.

Deutsch	Englisch	Argumente	Verwendung
lesezeilenfluss	readlineflow	id	Öffnet einen Fluß dessen Bezeichner der Zahl der Datei entspricht and dann eine Zeile einliest.
lesebuchstabenfluss	readcharflow	id	Öffnet einen Fluß dessen Bezeichner der Zahl entspricht, die als Argument benutzt wird und dann ein Zeichen in diese Datei einliest. Dieses Primitiv sendet eine Zahl zurück, die den Wert eines Zeichens darstellt (ähnlich zu `lesezeichen`).
schreibezeilenfluss	writelineflow	id Liste	Schreibt die Textzeile in der Liste an den Anfang der Datei bezeichnet durch id. Seien Sie vorsichtig: Das Schreiben funktioniert nur, wenn der Fluss durch das Primitiv `closeflow` geschlossen wurde.
hängezeileanfluss	appendlineflow	id Liste	

			Schriebt die Textzeile in der Liste an das Ende der Datei bezeichnet durch id. Seien Sie vorsichtig: Das Schreiben funktioniert nur, wenn der Fluss durch das Primitiv `closeflow` geschlossen wurde.
schließefluss	closeflow	id	Schliesst den Fluss mit dem Bezeichner als Argument.
endefluss?	endflow?	id	Sendet `wahr`, wenn es das Ende der Datei ist, sonst `falsch`.

Hier ist ein Beispiel für die Verwendung von erlaubten Primitiven, um von einer Datei zu lesen und zu schreiben. Ich werde dieses Beispiel im Rahmen von Windows geben. Andere Benutzer sollten das folgende Beispiel anpassen können.

Das Ziel dieser Datei ist es, die Datei `c:\example` zu erzeugen, die die folgenden drei Zeilen enthält:

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ
Abcdefghijklmnopqrstuvwxyz
0123456789

# Sie können einen Fluss für die gewünschte Datei öffnen.
# Diesem Fluss wird die Zahl 2 gegeben.

# funk nicht? wechsleordner "..
# funk nicht: wechsleordner "c:\\Program\ files\\
wechsleordner "c:\\Program\ files
; wechsleordner "beispiel-57
öffnefluss 2 "beispiel.txt

# Sie tippen die gewünschten Zeilen

schreibezeilenfluss 2 [ABCDEFGHIJKLMNOPQRSTUVWXYZ]
schreibezeilenfluss 2 [abcdefghijklmnopqrstuvwxyz]
schreibezeilenfluss 2 [0123456789]

# Sie schließen den Fluss, um das Schreiben zu beenden
schließefluss 2
```

Jetzt können Sie sehen, dass die Schreibprozedur richtig lief:

```
# Sie öffnen einen Fluss für die Datei, die Sie lesen wollen.
# Diesem Fluss wird die Zahl ``0`` gegeben.

öffnefluss 0 "beispiel.txt

# Sie lesen jede Zeile eine nach der anderen aus der Datei

dz lesezeilenfluss 0
dz lesezeilenfluss 0
dz lesezeilenfluss 0

# Sie schließen den Fluss:

schließefluss 0
```

Wenn Sie die Zeile 'Großartig!' hinzufügen wollen:

```
wechsleordner "c:\\
öffnefluss 1 "Beispiel]
```

```
hängezeileanfluss 1 [Großartig!]
schließefluss 1
```

5.8 Die fortgeschrittene Fülle−Funktion

Zwei Primitive erlauben, eine Figur zu färben. Die Primitive **`fülle`** und **`fülleform`**.

Diese Primitive erlauben eine Form anzumalen. Diese Primitive können mit den verfügbaren 'Fülle'−Eigenschaften von vielen Bildretuscheprogramme verglichen werden. Dieses Merkmal kann sich auf die Ränder des Designbereichs ausdehnen. Es gibt zwei Regeln, die in dieser Reihenfolge eingehalten werden müssen, um dieses Primitiv richtig zu benutzen:

- Der Schreibstift muss abgesenkt werden (`stiftab`).
- Der Igel darf sich auf keinem Bildelement der Farbe befinden, mit der die Form gefüllt werden soll. (Wenn Sie Figuren `rot` anmalen wollen, darf er nicht auf `rot` sitzen...)

  Nehmen wir ein Beispiel, um den Unterschied zwischen `fülle` und `fülleform` zu sehen:

Das Bildelement unter der Schildkröte ist in diesem Augenblick `weiß`. Das Primitiv `fülle` wird alle benachbarten Bereiche anmalen

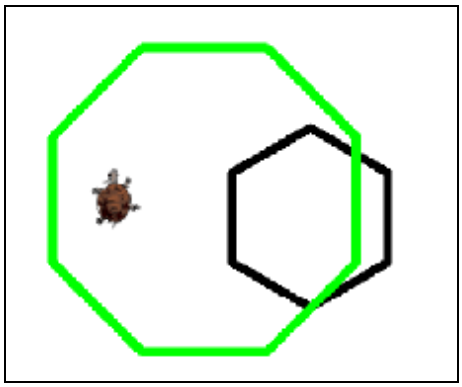


Abbildung 5.1: Am Anfang weiße Bildelemente mit der aktuellen Stiftfarbe an. Wenn Sie zum Beispiel tippen: `setzestiftfarbe 1 fülle` . Gehen wir jetzt zurück zu dem ersten Fall.

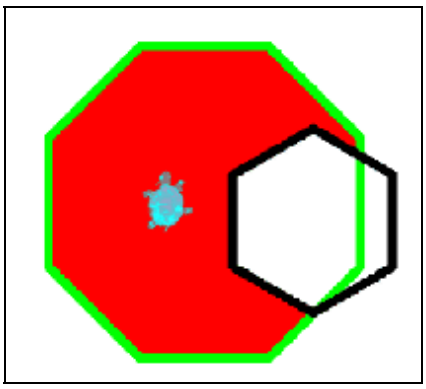
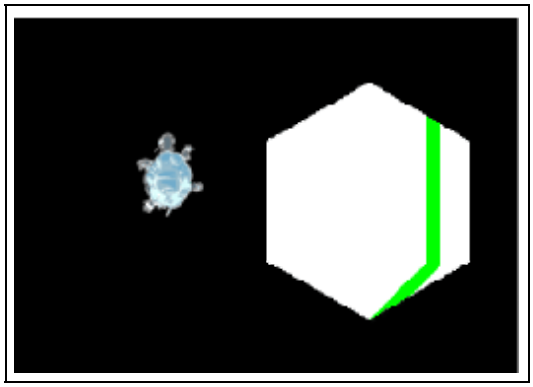


Abbildung 5.2: Mit dem Primitiv `Fülle`

Wenn die Stiftfarbe des Igel `schwarz` ist, wird das Primitiv `füllform` alle Bildelemente färben bis es die aktuelle Farbe (hier `schwarz`) findet.



Dies ist ein gutes Beispiel für den Gebrauch dieses Primitivs:

```
lerne halbkreis :c
  # ziehe einen Halbkreis vom Durchmesser :c
  wiederhole 180 [vw :c*tan 0.5 re 1]
  vw :c*tan 0.5
  re 90 vw :c
  re 90 # bringt Igel wieder in die Anfangsrichtung
Ende
```

□

Abbildung 5.3: Mit dem Primitiv `füllform` , wenn Sie tippen: `setzestiftfarbe 0 füllform`

`tan` gibt es schon als Primitiv:

```
lerne tangens :winkel
  # gibt den Tangens des Winkels wieder
  rückgabe (sin :winkel)/cos :winkel
Ende

lerne regenbogen :c
  wenn :c<100 [stoppe]
  halbkreis :c re 180 vw 20 li 90
  regenbogen :c-40
Ende

lerne dep
  sh re 90 vw 20 li 90 sa
Ende

lerne rbogen
  vi regenbogen 400 rd li 90 vw 20 rw 120 ns sh re 90 vw 20 sa
  setzestiftfarbe 0 fülle dep
  setzestiftfarbe 1 fülle dep
  setzestiftfarbe 2 fülle dep
  setzestiftfarbe 3 fülle dep
  setzestiftfarbe 4 fülle dep
  setzestiftfarbe 5 fülle dep
  setzestiftfarbe 6 fülle dep
Ende
```

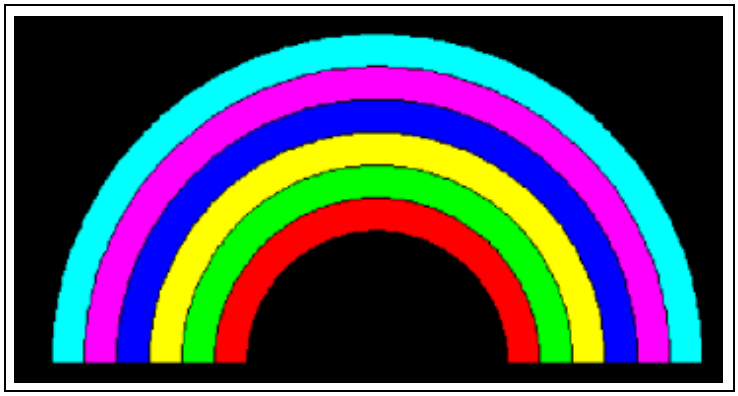


Abbildung 5.4: Bogen herein−LOGO

5.9 Unterbrechungs−Befehle

Logo hat drei Abbruchbefehle: `stoppe`, `stoppealles` und `rg, Rückgabe`.

`Stoppe` kann zwei Ergebnisse haben. Wenn es in eine `Wiederhole−` oder `Während`−Schleife eingeschlossen ist, springt das Programm heraus aus der Schleife. Wenn es in einer Prozedur vorkommt, bricht das Programm sofort aus der Prozedur aus.

`Stoppealles` bricht das Programm aus der ganzen Prozedur heraus sofort ab und stoppt.

`Rückgabe, rg` bricht aus einer Prozedur aus mit einem zurückzugebenden Wert. Sehen Sie die zahlreichen Fälle zum Gebrauch dieser zwei Primitive in den Beispielen am Ende von diesem Handbuch.

5.10 Der Viele−Igel−Modus

Es ist möglich mehrere aktive Igel auf dem Schirm zu haben. Standardmäßig ist beim Start von XLogo nur ein Igel vorhanden. Seine Nummer ist die 0. Wenn Sie einen neuen Igel "erschaffen" wollen, können Sie das Primitiv `si setzeigel` gefolgt von der Anzahl der Igel verwenden. Um Obstruktion zu verhindern, wird der Igel am Ursprung geschaffen und ist unsichtbar (Sie müssen `zi zeigeeigel` benutzen, um ihn zu zeigen). Dann ist der neue Igel der aktive Igel, er gehorcht allen klassischen Primitiven, während Sie den aktiven Igel mit `si setzeigel` nicht ändern. Die größtmögliche Anzahl von verfügbaren Igel kann gesetzt werden im Menü Hilfsmittel−Einstellungen − Tab Optionen. Das Primimtiv dazu heißt `Setzeigelmax`.

Hier sind die Primitive für den Viele−Igel−Modus:

Deutsch	Englisch	Argumente	Verwendung
sigel, setzeigel	sturtle, setturtle	a: Zahl	Der Igel mit Nummer a ist nun der aktive Igel. Beim Start von Xlogo ist der aktive Igel der mit der Nummer 0.
igel	turtle	keine	Ergibt die Nummer des aktiven Igel.
igelliste	turtles	keine	Ergibt eine Liste, die alle Nummern der Igel enthält, die tatsächlich auf dem Schirm sind.
	killturtle	a: Zahl	Löscht den Igel mit der Nummer a

ligel, löscheigel			
setzeigelmax	setTM, setturtlesmax	Ganzzahl	Setzt die maximale Anzahl Igel für den Viele−Igel−Modus.
igelmax	tm, turtlesmax	none	Ergibt die maximale Anzahl Igel für den Viele−Igel−Modus.

5.11 Musik spielen

Deutsch	Englisch	Argumente	Verwendung
fol, folge	seq, sequence	a: Liste	Erzeugt eine Sequenz der Liste im Speicher. Lesen Sie nach dieser Tabelle, um zu lernen wie eine Sequenz zu erzeugen ist.
spiele	play	keine	Spielt die Sequenz im Speicher.
instr, instrument	instr, instrument	keine	Ergibt die Zahl, die dem gewählten Instrument entspricht.
sinstr, setzeinstrument	sinstr, setinstrument	a: Zahl	Das gewählte Instrument ist nun das Instrument mit der Nummer a. Sie können die Liste aller verfügbaren Instrumente sehen im Menü Hilfsmittel−Einstellungen−Tab Sound.
indfol, indexfolge	indseq, indexsequence	keine	Ergibt die Position des Lesezeigers in der aktuellen Sequenz.
sindfol, setzeindexfolge	sindseq, setindexsequence	a: Zahl	Setzt den Lesezeiger auf den Index a in der aktuellen Sequenz im Speicher.
lfol, löschefolge	delseq, deletesequence	keine	Löscht die aktuelle Sequenz im Speicher.

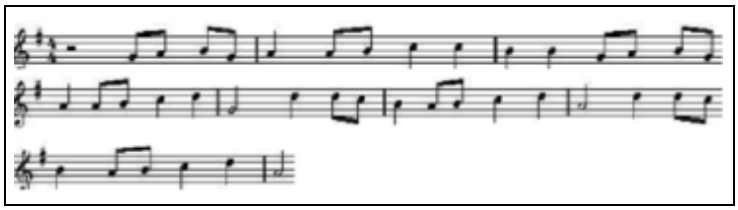
Wenn Sie Musik spielen wollen, müssen Sie die Noten im Speicher in eine Sequenz genannte Liste stellen. Um diese Folge zu erzeugen, können Sie das Primitiv `fol` oder `folge` benutzen. Dies sind die Regeln, die einzuhalten sind, um eine gültige Sequenz zu schaffen:

`do re mi fa sol la si` : die üblichen Noten der ersten Oktave.

Um ein spitzes re zu erzeugen, schreiben wir `re +` Um ein flaches re zu erzeugen, schreiben wir `re −`

Wenn Sie eine Oktave hinauf oder hinunter gehen wollen, benutzen Sie das Symbol ":" gefolgt von + oder −. Z. B. werden nach `:++` alle Noten der Sequenz zwei Oktaven höher gespielt (zwei ++). Standardmäßig werden Noten für eine Dauer von `1` gespielt. Wenn Sie sie verlängern oder vermindern wollen, schreiben Sie die Zahl, die der Dauer der Noten entspricht. Z. B. `folge [sol 0.5 la si]` wird `sol` mit einer Dauer `1` und `la si` mit einer Dauer `0.5` spielen (zweimal schneller).

Wenn Sie dieses Beispiel spielen wollen:



```
lerne tabac
  # erzeugt die Sequenz von Noten
  fol [0.5 sol la si sol 1 la 0.5 la si 1 :+ do do :− si si 0.5 sol la si sol
     1 la 0.5 la si 1 :+ do re 2 :− sol ]
  fol [:+ 1 re 0.5 re do 1 :− si 0.5 la si 1 :+ do re 2 :− la ]
  fol [:+ 1 re 0.5 re do 1 :− si 0.5 la si 1 :+ do re 2 :− la ]
  fol [0.5 sol la si sol 1 la 0.5 la si 1 :+ do do :− si si 0.5 sol la si sol
     1 la 0.5 la si 1 :+ do re 2 :− sol ]
Ende
```

tabac spiele

Um Musik zu hören, beginnen Sie mit dem Kommando: `tabac play` Jetzt können wir eine interessante Anwendung des Primitivs `sindseq` sehen. Schreiben Sie diese Kommandos:

```
delseq    # Lösche die Sequenz aus dem Speicher
tabac     # Lege die Sequenz in den Speicher
sindfol 2 # Setze die Schreibmarke auf das zweite "la".
tabac     # Setze die gleiche Sequenz in den Speicher, aber übersetzt von 2.
play      # großartig!
```

Sie können das Instrument mit dem Primitiv `sinstr, setzeinstrument` oder mit dem Menü Hilfsmittel−Einstellungen auf dem Tab Sound wählen. Sie werden dort die Liste von allen verfügbaren Instrumenten mit ihrer dazugehörigen Zahl finden.

5.12 Schleifen

In XLogo gibt es drei Primitive, die die Konstruktion von Schleifen erlauben: `Wiederhole`, `Für` und `Solange` .

5.12.1 Eine Schleife mit Wiederhole

Dies ist die Syntax für **`Wiederhole`**:

`Wiederhole n List_von_Kommandos`

n ist eine ganze Zahl und `Liste_von_Kommandos` enthält eine Liste der auszuführenden Befehle. Der Logo−Interpreter wird die Befehle in der Liste n mal ausführen: So braucht das Kommando nur einmal geschrieben zu werden!

Z.B.:

```
wiederhole 4 [vw 100 li 90]  # ein Quadrat mit der Seitenlänge 100
wiederhole 6 [vw 100 li 60]  # ein Sechseck mit der Seitenlänge 100
wiederhole 360 [vw 2 li 1]   #−A−Uh... 360−Eck mit Seitenlänge 2
                                  # kurz gesagt: fast ein Kreis!
```

In eine Wiederhole−Schleife eingeschlossen gibt die interne Variable **Whzahl** die Anzahl der laufenden Iteration zurück. (Dabei ist die erste Iteration die Zahl 1).

```
wiederhole 3 [dz whzahl]
1
2
3
```

5.12.2 Eine Schleife mit Für

damit weisen Sie einer Variablen einige mit einer Schrittweite aufeinander folgende Werte aus einem festen Bereich zu. Hier ist die Syntax von **Für**:

```
Für Liste1 Liste2
```

Liste1 enthält drei Argumente: der Variablenname, der Anfangswert und der Endwert. Ein viertes Argument ist das Inkrement(die optionale Schrittweite zwischen zwei aufeinander folgenden Werten). Standardwert ist 1 . Hier sind ein paar Beispiele:

```
für [i 1 4][dz :i*2]
```

```
2
4
6
8
```

Jetzt wird i von 7 nach 2 verringert, jedesmal jeweils um 1.5 # Schauen Sie sich das negative Inkrement an # dessen Quadrat angezeigt wird.

```
für [i 7 2 −1.5][dz (liste :i potenz :i 2)]
```

```
7 49
5.5 30.25
4 16
2.5 6.25
```

5.12.3 Eine Schleife mit Solange

Dies ist die Syntax für **Solange**:

```
Während Liste_istauszuwerten Liste_von_Kommandos
```

Liste_istauszuwerten enthält eine Liste von Befehlen, die zu einem booleschen Wert ausgewertet werden kann. Liste_von_Kommandos enthält eine Liste auszuführender Befehle. Der Logo−Interpreter wird die Liste_von_Kommandos auswerten, **solange** die Liste_istauszuwerten Wahr zurückgibt.

Z.B.:

```
Solange [wahr] [re 1] # dreht den Igel um sich
```

Nun ein Beispiel, das uns erlaubt das Alphabet rückwärts zu buchstabieren

```
setze "liste "abcdefghijklmnopqrstuvwxyz
solange [nicht leer? :liste] [dz letztes :liste  setze "liste ohneletztes :liste ]
```

5.13 Eingaben vom Benutzer empfangen

5.13.1 Interagieren Sie mit der Tastatur

Gegenwärtig kann Text vom Benutzer während der Programmausführung hauptsächlich über 3 Primitive akzeptiert werden: `Taste?`, `lesezeichen` und `lese` .

`Taste?`: Wird als `wahr` oder `falsch` gelesen, abhängig davon, ob seit dem Beginn der Programmausführung eine Taste gedrückt worden ist oder nicht.

`lesezeichen`:

- Wenn `Taste?` den Wert `Falsch` ergibt, pausiert das Programm bis der Benutzer eine Taste drückt.
- Wenn Taste? `wahr` ist, gibt es die Taste aus, die zuletzt gedrückt war. Dies sind die Werte für besondere Tasten:

A −−> 65	B −−> 66	C −−> 67	etc ...	Z −−> 90
−−> −37 or −226 (NumPad)	−−> −38 or −224	−−> −39 or −227	−−> −40 or −225	.
Echap −−> 27	F1 −−> −112	F2 −−> −113	....	F12 −−> −123
Shift −−> −16	Espace −−> 32	Ctrl −−> −17	Enter −−> 10	.

Tabelle 5.11: Werte für besondere Tasten

Wenn Sie unsicher über den von einer Taste zurückgegebenen Wert sind, können Sie tippen:

```
dz lesezeichen
```

Der Interpreter wird dann warten, bis Sie auf eine Taste tippen, bevor er den Wert zurückgibt.

```
lese liste_titel Wort:
```

Gibt einen Dialogkasten, dessen Titel `liste_titel` ist. Der Benutzer kann dann in einem Textfeld eine Antwort eingeben und die Antwort wird dann in Gestalt von einem Wort oder einer Liste, wenn der Benutzer mehrere Wörter geschrieben hat, in der Variablen `:word` gespeichert werden. Ausgewertet wird, wenn der OK−Knopf gedrückt wird.

5.13.2 Einige Beispiele zum Gebrauch

```
lerne alter
  lese [Was ist Ihr Alter?] "Alter
  # setze "alter erstes :alter
  wenn :alter<18 [dz [Sie sind ein Kind.]]
  wenn oder :alter=18 :alter>18 [dz [Sie sind ein Erwachsener.]]
# so nicht:
# wenn :alter>=18 [dz [Sie sind ein Erwachsener.]]
  wenn :alter>99 [dz [Respekt, Respekt!!]]
  dz "
Ende

lerne rallye
  wenn taste? [
```

```
    setze "auto lesezeichen
    wenn :auto=-37 [li 90]
    wenn :auto=-39 [re 90]
    wenn :auto=-38 [vw 10]
    wenn :auto=-40 [rw 10]
    wenn :auto=27 [stoppe]
  ]
  rallye
Ende

# Sie können den Igel mit den Pfeiltasten steuern und mit Esc stoppen.
```

5.13.3 Interagieren Sie mit der Maus

Gegenwärtig können Mausereignisse vom Benutzer während der Programmausführung über drei Primitive akzeptiert werden:

`lesemaus, mauspos` und `Maus?` .

`lesemaus`: Das Programm pausiert bis der Benutzer die Maus drückt. Dann gibt es eine Zahl zurück, die das Ereignis repräsentiert.

- Dies sind die unterschiedlichen Werte:
  - **0** Die Maus wurde bewegt
  - **1** Der Knopf 1 ist gedrückt gewesen
  - **2** Der Knopf 2 ist gedrückt gewesen

    Der Knopf 1 ist der linke Knopf, der Knopf 2 ist der nächste rechts...

`mauspos, mausposition`: Gibt eine Liste zurück, die die Position der Maus enthält.

`Maus?`: Gibt ein `Wahr` zurück, wenn wir seit Programmbeginn die Maus anfassen, sonst ein `Falsch`.

5.13.4 Einige Beispiele zum Gebrauch

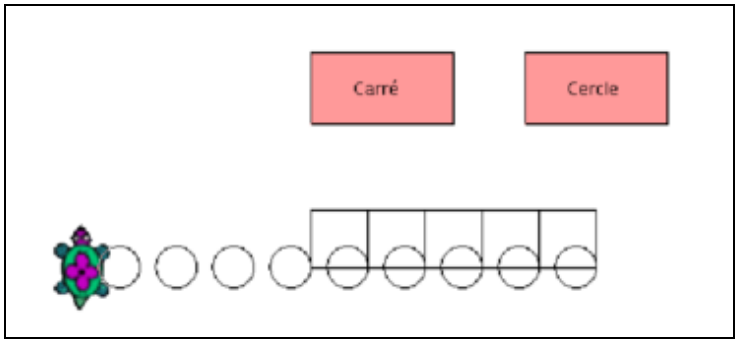
In der ersten Prozedur folgt der Igel der Maus, wenn er sich auf dem Bildschirm bewegt.

```
lerne beispiel
  # wenn die Maus bewegt wird, geht der Igel zur nächsten Position
  wenn lesemaus=0 [aufpos mauspos]
  beispiel
Ende
```

In dieser zweiten Prozedur ist es wie vorher, nun aber müssen Sie mit dem linken Knopf der Maus klicken, wenn Sie wollen, dass sich der Igel bewegt.

```
lerne beispiel2
  wenn lesemaus=1 [setzepos mauspos]
  beispiel2
Ende
```

In diesem dritten Beispiel schaffen wir zwei rosafarbene Schaltflächen. Wenn wir links klicken auf die linke Schaltfläche, zeichnen wir ein Quadrat mit einer Seite von `40`. Klicken wir links auf die rechte Schaltfläche, zeichnen wir einen kleinen Kreis. Zuletzt klicken wir rechts auf die rechte Schaltfläche und das Programm stoppt.




```
lerne knopf
  # erzeugt eine rosafarbene, rechteckige Schaltfläche (Höhe 50, Breite 100)
  wiederhole 2 [vw 50 re 90 vw 100 re 90]
  re 45 stifthoch vw 10 stiftab setzestiftfarbe [255 153 153]
  fülle rw 10 li 45 stiftab setzestiftfarbe 0
Ende

lerne lance
  lb knopf stifthoch setzepos [150 0] stiftab Knopf
  sh setzepos [30 20] sa igeltext "Quadrat
  sh setzepos [180 20] sa igeltext "Kreis
  sh setzepos [0 −100] sa
  maus
Ende

lerne maus
  # wir legen den Wert von lesemaus in die Variable ev
  setze "ev lesemaus
  # wir legen die erste Koordinate der Maus in die Variable x
  setze "x element 1 mauspos
  # wir legen die zweite Koordinate der Maus in die Variable y
  setze "y element 2 mauspos
  # Wenn wir den linken Knopf Schaltfläche klicken
  wenn :ev=1 & :x>0 & :x<100 & :y>0 & :y<50 [Quadrat]
  # Wenn wir den rechten Knopf | Schaltfläche klicken
  wenn :x>150 & :x<250 & :y>0 & :y<50 [
     wenn :ev=1 [meinkreis]
     wenn :ev=3 [stoppe]
  ]
maus
Ende

lerne Kreis2
  wiederhole 90 [vw 1 li 4] li 90 sh
  vw 40 re 90 sa
Ende

lerne Quadrat
  wiederhole 4 [vw 40 re 90]
Ende
```

lance startet

5.13.5 Graphische Komponenten gebrauchen

Mit XLogo können Sie mehrere graphische Komponenten auf der Zeichenfläche hinzufügen (Schaltfläche, Kombo−Menü). Alle Primitive, die dem Benutzer erlauben jene Komponenten zu manipulieren, fangen mit dem Präfix Gui an (für Graphische Anwenderschnittstelle).

Schaffen Sie sich eine Komponente

Zuerst müssen Sie jene graphischen Objekte erstellen, dann können Sie einige ihrer Eigenschaften ändern und zuletzt auf der Zeichenfläche zeigen.

- Um eine Schaltfläche zu schaffen, müssen wir das Primitiv **`guitaste`** benutzen. Hier ist die Syntax:

  # Dieser Befehl schafft eine Schaltfläche mit dem Namen b # auf die ""Klicke"" geschrieben wird:

  ```
  guitaste "B "Klicke
  ```

- Um ein Kombo−Menü zu schaffen, müssen wir das Primitiv **`guimenü`** benutzen. Hier ist die Syntax:

  # Dieser Befehl erschafft ein Kombomenü mit dem Namen m # und die Liste enthält 3 Elemente : Element1, Element2 und Element3

  ```
  guimenü "m [Element1 Element2 Element3]
  ```

Modifizieren Sie einige Eigenschaften von graphischen Komponenten

`guiposition`: Positioniert das graphische Element an einem bestimmten Ort mit seiner Koordinate. Zum Beispiel, wenn Sie die Schaltfläche an den Punkt mit den Koordinaten (20; 100) stellen wollen, werden Sie schreiben:

```
guiposition "b [20 100]
```

Wenn Sie keinen Ort für die Komponente spezifizieren, wird er standardmäßig auf die obere linke Ecke der Zeichenfläche gesetzt werden.

`guientferne`: Beseitigt eine graphische Komponente. Zum Beispiel um die Schaltfläche zu entfernen:

```
guientferne "b
```

`guiaktion`: Definiert eine Aktion für die Komponente, wenn der Benutzer damit interagiert.

Bewegt den Igel um 100 vor, wenn wir auf die Schaltfläche "b klicken

```
guiaktion "b [vw 100]
```

Für das Kombomenü hat jedes Element seine eigene Aktion

```
guiaktion "m [[dz Element1] [dz "Element2] [dz "Element3]]
```

`guizeichne`: Zeigt die graphische Komponente auf der Zeichenfläche. Zum Beispiel, um die Schaltfläche zu zeigen:

```
guizeichne "b
```

5.14 Zeit und Datum

XLogo hat mehrere Primitive für Datum, Zeit und Countdown.

Deutsch	Englisch	Argumente	Verwendung

warte	wait	n: Ganzzahl	Hält das Programm an, und daher auch den Igel, für die Dauer von n/60 Sekunden.
stopuhr	chrono, chronometre	n: Ganzzahl	Startet einen Countdown von n Sekunden. Wir wissen, wann dieser Countdown beendet ist, wenn wir das Primitiv endecountdown? benutzen.
endecountdown?	endcountdown?	keine	Ergibt "wahr, wenn es keinen aktiven Countdown gibt, sonst "false wenn der Countdown aktiv ist.
datum	date	keine	Ergibt eine Liste die drei Ganzzahlen enthält, die das Datum darstellen. Die erste Ganzzahl zeigt den Tag, die zweite den Monat und die letzte das Jahr. —> [tag monat jahr]
zeit	time	aucun	Ergibt eine Liste die drei Ganzzahlen enthält, die die Zeit darstellen. Die erste Ganzzahl zeigt die Stunde, die zweite die Minute und die letzte die Sekunden. —> [stunde minute sekunde]
vergangenezeit	pasttime	keine	Ergibt die vergangene Zeit in Sekunden seit XLogo gestartet wurde.

Der Unterschied zwischen Warte und Countdown ist, dass Countdown das Programm nicht anhält.

Hier ist ein Beispiel:

```
lerne uhr
  # zeigt die Zeit im nummerischen Format
  # wir frischen die Zeit alle fünf Sekunden auf
  wenn endecountdown? [
    lb
    sfont 75 vi
    setze "Heu zeit
    setze "h erstes :heu
    setze "m element 2 :heu
    # Wir zeigen zwei Zahlen für Sekunden und Minuten. (wir müssen eine 0 hinzufügen)
    wenn :m-10<0 [setze "m Wort 0 :m]
    setze "s zuletzt :heu
    # Wir zeigen zwei Zahlen für Sekunden und Minuten. (wir müssen eine 0 hinzufügen)
    wenn :s-10<0 [setze "s Wort 0 zu :s]
    igeltext wort wort wort wort :h ": :m ": :s
    countdown 5
  ]
  uhr
Ende
```

5.15 XLogo im Netzwerk benutzen

5.15.1 Das Netz − Wie geht das?

Zuerst müssen wir in die Grundlagen für Netzwerkkommunikation einführen, bevor wir die XLogo−Primitive benutzen können.

Zwei Computer (oder mehr) können über ein Netz kommunizieren, wenn in beiden Ethernet−Karten eingebaut sind.

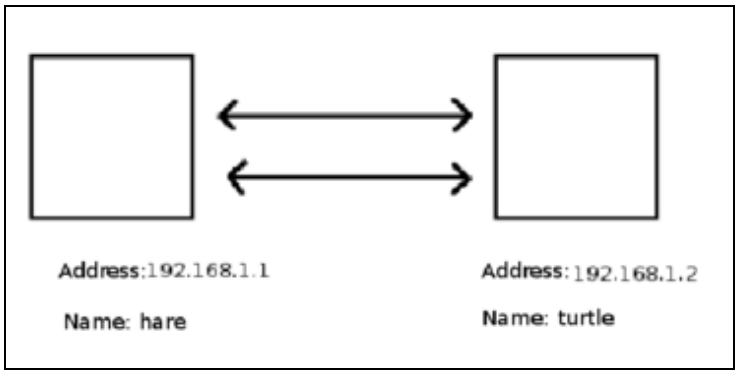


Abbildung 5.5: Ein einfaches Netz

Jeder Computer wird durch eine persönliche IP−Adresse identifiziert. Diese IP−Adresse besteht aus vier ganzen Zahlen, jede zwischen 0 und 255 und getrennt durch einen Punkt. Zum Beispiel wird der erste Computer in der Abbildung adressiert durch die IP **192.168.1.1**

Weil es nicht leicht ist, sich an diese Zahlen zu erinnern, ist es auch möglich, jeden Computer über einen gewohnteren Namen zu identifizieren. Wie wir in der Abbildung sehen, können wir mit dem rechten Computer mit Hilfe seiner IP−Adresse **192.168.1.2** kommunizieren oder mit seinem Namen **Turtle** .

Hier möchte ich eine Sache anmerken: Der lokale Computer, an dem Sie arbeiten, besitzt die Adresse **127.0.0.1** . Sein allgemeiner Name ist **localhost** . Wir werden dies später in der Praxis sehen.

5.15.2 Primitive für das Netz

XLogo hat 4 Primitive, die ihm ermöglichen, über ein Netz zu kommunizieren: `höretcp`, `ausführetcp`, `chattcp` und `sende`. In allen künftigen Beispielen werden wir den einfachen Fall von zwei Computern in der bisherigen Abbildung nehmen.

`höretcp`: Dieses Primitiv ist die Grundlage für alle Netzkommunikation. Es benötigt kein Argument. Wenn Sie dieses Primitiv auf einem Computer ausführen, wird der Computer nach von anderen Computern auf das Netz geschickte Anweisungen achten.

`ausführetcp`: Dieses Primitiv erlaubt die Ausführung von Anweisungen von einem Computer auf dem Netz.

Syntax: `ausführetcp Wort Liste`

`Wort` ist die angerufene IP−Adresse oder der Computername, `Liste` enthält auszuführende Anweisungen.

Beispiel: Ich bin auf Hase, ich will ein Quadrat mit einer Seite von 100 auf dem anderen Computer zeichnen.

Dazu muss ich auf dem Computer **Turtle** mit dem Befehl `höretcp` beginnen. Dann, auf **Hase** schreibe ich :

```
ausführetcp "192.168.1.2 [Wiederhole 4 [vw 100 re 90]]
```

oder

```
ausführetcp "Turtle [Wiederhole 4 [vw 100 re 90]]
```

chattcp: Erlaubt den Plausch zwischen zwei Computern auf einem Netz. Auf jedem Computer zeigt es ein Chatfenster.

Syntax: `Chattcp Wort Liste`

`Wort` ist die angerufene IP−Adresse oder der Computername, `Liste` enthält den zu zeigenden Satz.

Beispiel: Hase will mit Turtle reden.

Zuerst führt **Turtle** `höretcp` aus und wartet damit auf Anweisungen von Netzcomputern. Dann schreibt **Hase**: `chattcp "192.168.1.2 [hallo Turtle]` . Chat−Fenster werden auf beiden Computern aufgehen, das erlaubt ihnen miteinander zu reden.

sendetcp: Schicken Sie Daten zu einem Computer auf dem Netz und geben Sie seine Antwort zurück.

Syntax: `Sendetcp Wort Liste`

`Wort` ist die angerufene IP−Adresse oder der Computername, `Liste` enthält die zu schickenden Daten. Wenn Xlogo auf dem anderen Computer begonnen wird, wird er mit "In Ordnung" antworten.

Mit diesem Primitiv ist es möglich mit einem Roboter über seinen Netzzugang zu kommunizieren. Dann könnte die Antwort des Roboters unterschiedlich sein.

Beispiel: Turtle will Hase den Satz "3.14159 ist wirklich Pi" schicken.

Zuerst führt **Hase** `höretcp` aus, dass er auf den anderen Computer wartet. Dann schreibt **Turtle**:

```
druckezeile sendetcp "Hase [3.14159 ist wirklich Pi]
```

Eine kleine Andeutung: Starten Sie zwei Instanzen von XLogo auf dem gleichen Computer.

- Führen Sie im ersten Fenster `höretcp` aus.
- Schreiben Sie in das zweite `ausführetcp "127.0.0.1 [vw 100 re 90]`

  Sie können den Igel im anderen Fenster bewegen! (heh, heh, das ist möglich, weil 127.0.0.1 Ihre lokale Adresse bezeichnet, denn es ist Ihr eigener Computer...)

XLogo Handbuch (siehe Quelle)

Übersetzt von Michael Malien	Erzeugt durch Txt2Tags
Geändert am: 07/28/08 11:35:30	am 2008−07−28 um 12:15:08

XLogo Handbuch

6 Programm−Beispiele
geändert am 28.07.2008 11:35

6 Programm−Beispiele

6.1 Häuser zeichnen

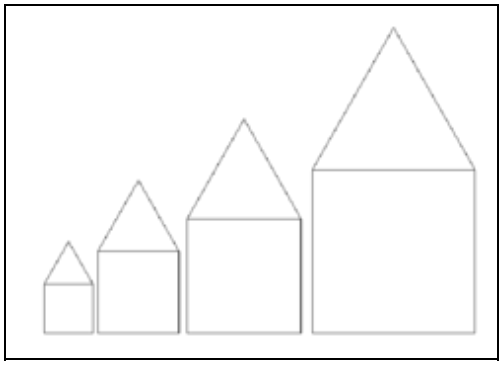


Abbildung 6.1: Häuser

```
lerne häuser
  löschebild stifthoch links 90 vorwärts 200 rechts 90 stiftab versteckeigel
  haus 3 platz 3 haus 5 platz 5 haus 7 platz 7 haus 10
Ende

lerne haus :c
  wiederhole 4 [vorwärts 20*:c rechts 90]
  vorwärts 20*:c rechts 30
  wiederhole 3 [vorwärts 20*:c rechts 120]
Ende


lerne platz :c
  stifthoch links 30 rückwärts :c*20 rechts 90 vorwärts :c*22 links 90 stiftab
Ende
```

6.2 Ein ganzes Rechteck zeichnen

Abbildung 6.2: Rechteck

```
lerne rechteck :lo :la
  wenn :lo=0 | :la=0 [stoppe]
  wiederhole 2[vorwärts :lo rechts 90 vorwärts :la rechts 90]
  rechteck :lo −1 :la −1
Ende
```

6.3 Fakultät

Gedächtnishilfe

$$5! = 5 \times 4 \times 3 \times 2 \times 1 = 120$$

```
lerne fac :n
  wenn :n=1[rückgabe 1][rückgabe :n*fac :n-1]
Ende

dz fac 5
120.0

dz fac 6
720.0
```

6.4 Die Schneeflocke (mit Dank an Georges No¨ el))

```
lerne koch :order :len
  wenn :order < 1 | :len <1 [vorwärts :len stoppe]
  koch :order-1 :len/3
  links 60
  koch :order-1 :len/3
  rechts 120
  koch :order-1 :len/3
  links 60
  koch :order-1 :len/3
Ende
```

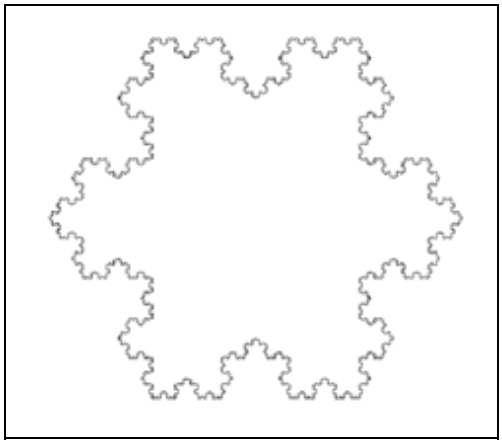


Abbildung 6.3: Die Schneeflocke

```
lerne kochflocke :order :len
  wiederhole 3 [rechts 120 koch :order :len]
Ende
```

`kochflocke 5 450`

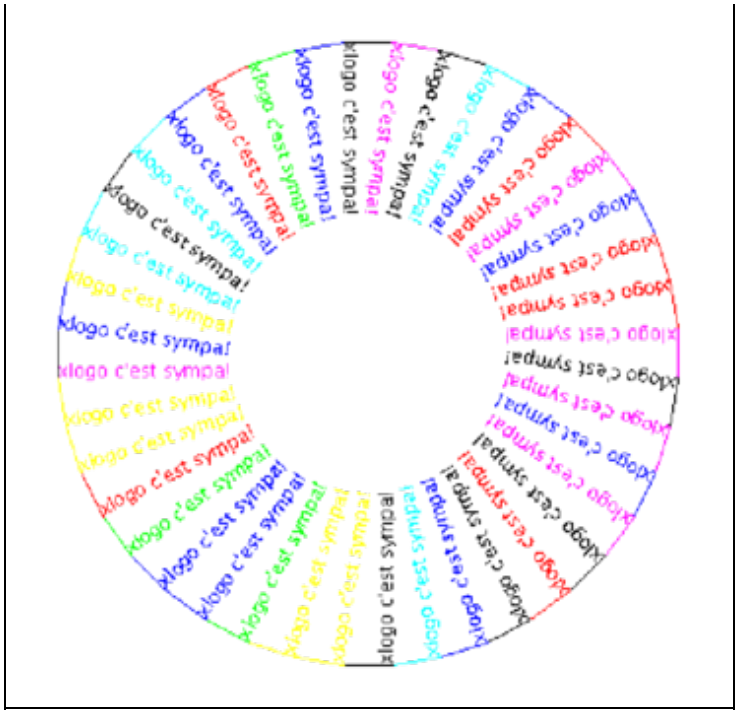


Abbildung 6.4: XLogo ist toll!

6.5 Ein wenig Schrift...

```
lerne schrift
  versteckeigel
    wiederhole 40[
      vorwärts 30 rechts 9 setzestiftfarbe zufallszahl 7 igeltext [XLogo ist toll!]
      ]
Ende
```

6.6 Und Konjugation...

6.6.1 Erste Version

```
lerne fr_zukunft :wort
  druckezeile satz "je wort :wort "ai
  druckezeile satz "tu wort :wort "as
  druckezeile satz "il wort :wort "a
  druckezeile satz "nous wort :wort "ons
  druckezeile satz "vous wort :wort "ez
  druckezeile satz "elles wort :wort "ont
Ende

Fr_zukunft "parler

  je parlerai
  tu parleras
  il parlera
  nous parlerons
  vous parlerez
  elles parleront
```

6.6.2 Zweiter Versuch

```
  lerne fut :wort
    setze "pronomen [je tu il nous vous elles]
    setze "endungen [ai as a ons ez ont]
    setze "i 0
    wiederhole 6[
      setze "i :i+1 dz satz element :i :pronomen wort :wort element :i :endungen
    ]
  Ende

fut "parler

  je parlerai
  tu parleras
  il parlera
  nous parlerons
  vous parlerez
  elles parleront
```

6.6.3 Oder sogar: Ein wenig Rekursion!

```
  lerne fu :verb
    setze "pronomen [je tu il nous vous elles]
    setze "endungen [ai as a ons ez ont]
    konjugiere :verb :pronomen :endungen
  Ende

  lerne konjugiere :verb :pronomen :endungen
    wenn leer? :pronomen [stoppe]
    druckezeile satz erstes :pronomen wort :verb erstes :endungen
    konjugiere :verb ohneerstes :pronomen ohneerstes :endungen
  Ende

fu "parler

  je parlerai
  tu parleras
  il parlera
  nous parlerons
  vous parlerez
  elles parleront
```

6.7 Alles über Farben

6.7.1 Einführung

Zunächst ein paar Erklärungen: Sie werden zur Kenntnis nehmen, dass das Befehl `Setpc` entweder eine Liste oder eine Zahl als Parameter haben kann. Hier sind wir an der Codierung von RGB−Werten interessiert.

Jede Farbe in XLogo wird durch drei Werte kodiert: `Rot`, `grün` und `blau`, woher der Name RGB−Verschlüsselung. Die drei Zahlen im Listenparameter nach dem Primitiv `setzestiftfarbe` repräsentieren deswegen die roten, grünen und blauen Bestandteile einer Farbe. Dieses Verschlüsseln ist nicht wirklich intuitiv, und daher können eine Idee von der Farbe bekommen, die durch diese Kodierung gegeben ist, durch die Benutzung des Dialogkastens in **Hilfsmittel−Wähle Stiftfarbe**.

Bei der Benutzung dieser Kodierung ist es sehr leicht, ein Bild zu transformieren. Zum Beispiel wenn Sie ein Farbfoto in ein Grautonbild verwandeln wollen, können Sie die Farbe jedes Bildelements des Bildes zum Durchschnitts−Wert der drei Bestandteile `[r g b]` verändern. Stellen Sie sich vor, dass die Farbe von einem Punkt des Bildes gegeben wird durch die Kodierung `[0 100 80]`. Nun berechnen Sie den Durchschnitt von diesen drei Zahlen:

`( 0+100+80 ) / 3 = 60`, und weisen dann die Farbe `[60 60 60]` diesem Bildelement zu. Diese Operation muss auf jedes Bildelement des Bildes durchgeführt werden.

6.7.2 Werden wir praktisch!

Wir werden ein `100 × 100` Bildelemente großes Bild in ein Grauwert−Bild transformieren.

Dies heißt, dass es `100 × 100 = 10000` zu ändernde Bildelemente gibt. Sie können auf das in diesem Beispiel benutzte Bild an der folgenden Adresse zugreifen:

http://xlogo.tuxfamily.org/pics/transfo.png

Dies ist, wie wir fortfahren werden: Zuerst werden wir uns auf die obere linke Ecke vom Bild beziehen, nämlich 0. Dann wird der Igel die ersten 100 Bildelemente der Anfangszeile, von den ersten 100 der zweiten Zeile gefolgt, prüfen und so weiter. Jedes Mal wird die Farbe des Bildelements mit `findefarbe`, und die Farbe wird dann verändert zu dem Durchschnitt von den drei `[r g b]` Werten. Hier ist der dazugehörige Code: (Vergessen Sie nicht, den `dateipfad` in der Prozedur zu verändern!)

```
lerne pixel :list
  # gebe den Durchschnitt der drei Zahlen [r g b] zurück
  setze "r erstes :list
  setze "list ohneerstes :list
  setze "g erstes :list
  setze "list ohneerstes :list
  setze "b erstes :list
  setze "b runde (:r+:g+:b)/3
  rückgabe satz liste :b :b :b
Ende
```

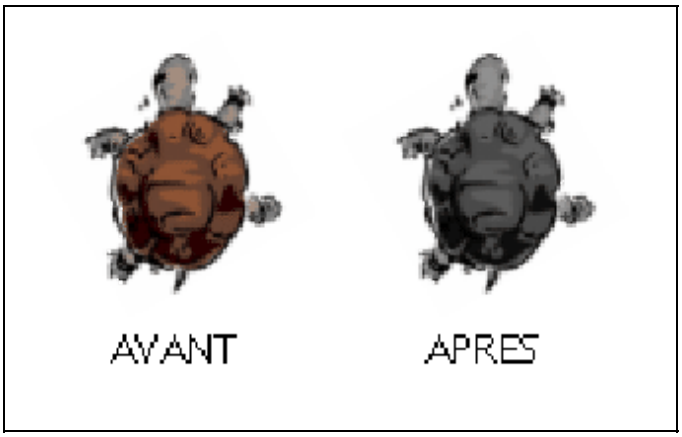


Abbildung 6.5: XLogo macht eine Bildretusche....

```
lerne grauwert :c
  wenn :y=−100 [stoppe]
  wenn :c=100 [setze "c 0 setze "y :y−1]
  # wir weisen die "Durchschnittsfarbe" des Pixel dem Stift zu
  setzestiftfarbe pixel findefarbe liste :c :y
  # wir verwandeln den Pixel in einen Grauwert
```

```
  pixel liste :c :y
  grauwert :c+1
Ende

lerne transform
  # Sie müssen den Pfad zu dem Bild transfo.png ändern
  # z.B. mit setzeordner "c:\\my_images ladebild "transfo.png]
  löschebild versteckeigel setzeordner "/heim/loic
  ladebild "transfo.png setze "y 0 grauwert 0
Ende
```

6.7.3 Und wenn Sie es negativ wollen?

Um aus einem Bild das Negativbild zu bekommen, können Sie den gleichen Prozess benutzen, außer dass an Stelle den Durchschnitt der Zahlen `R G B` zu ermitteln, Sie die Zahl durch die ersetzen, die Sie bekommen wenn Sie sie von 255 subtrahieren. Z. B.: Wenn ein Bildelement die Farbe `[2 100 200]` hat, ersetzen Sie sie durch die Farbe `[253 155 55]`.

Nur die Prozedur `pixel` muss zum folgenden Programm geändert werden:

```
lerne grauwert2 :c
  wenn :y=-100 [stoppe]
  wenn :c=100 [setze "c 0 setze "y :y-1]
  setzestiftfarbe pixel2 findefarbe liste :c :y
  pixel2 liste :c :y
  grauwert2 :c+1
Ende
```

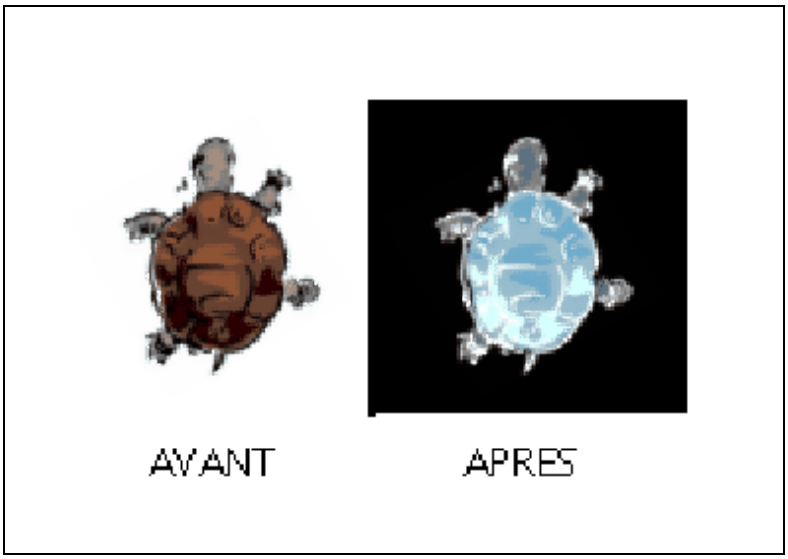


Abbildung 6.6: XLogo täuscht vor GIMP zu sein... (anmaßend? :-))

```
lerne transform2
  # Sie müssen den Pfad zu dem Bild transfo.png ändern
  # eg: setzeordner "c:\\my_images ladebild "transfo.png]
  versteckeigel löschebild setzeordner "/heim/loic
  ladebild "transfo.png setze "y 0 grauwert2 0
Ende

lerne pixel2 :list
  setze "r erstes :list
  setze "list ohneerstes :list
  setze "g erstes :list
  setze "liste ohneerstes :list
  setze "b erstes :list
```

```
  rückgabe satz liste 255-:r 255-:g 255-:b
Ende
```

6.8 Ein gutes Beispiel für die Benutzung von Listen (Dank an Olivier SC)

Ich hoffe, dass Sie dieses wunderbare Palindrom−Programm schätzen werden:

```
lerne umkehrew :w
  wenn leer? :w [rückgabe "]
  rückgabe wort letztes :w umkehrew ohneletztes :w
Ende

lerne palindrom :w
  wenn gleich? :w umkehrew :w [rückgabe "true] [rückgabe "false]
Ende

lerne palin :n
  wenn palindrom :n [druckezeile :n stoppe]
  druckezeile satz satz satz satz :n "mehr umkehrew :n "gleich summe :n umkehrew :n
  palin :n + umkehrew :n
Ende
```

```
palin 78
```

```
78 mehr 87 gleich 165
165 mehr 561 gleich 726
726 mehr 627 gleich 1353
1353 mehr 3531 gleich 4884
4884
```

6.9 Eine reizende Rose

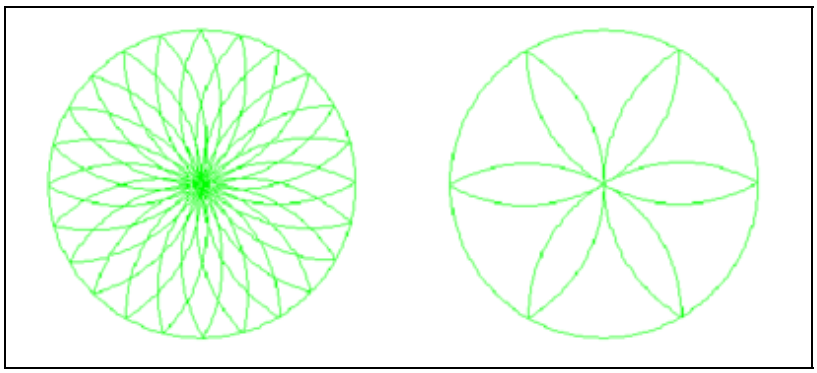


Abbildung 6.7: Besser als die Benutzung eines Kompass!

```
lerne rose
  wiederhole 6[ wiederhole 60[vorwärts 2 rechts 1] rechts 60
  wiederhole 120 [vorwärts 2 rechts 1] rechts 60]
Ende

lerne liebreizende_rose
  rose
  wiederhole 30[vorwärts 2 rechts 1]
  rose
  wiederhole 15[vorwärts 2 rechts 1]
  rose
  wiederhole 30[vorwärts 2 rechts 1]
```

```
    rose
  Ende

setzebildfarbe 0 löschebild setzestiftfarbe 5 versteckeigel rose
stifthoch aufpos [−300 0] stiftab aufkurs 0 liebreizende_rose
```

XLogo Handbuch (siehe Quelle)

Übersetzt von Michael Malien	Erzeugt durch Txt2Tags
Geändert am: 07/28/08 11:35:58	am 2008−07−28 um 12:15:20

XLogo Handbuch

7 Deinstallation und Lesezeichen
geändert am 27.03.2008 13:15

7 Deinstallation und Lesezeichen

7.1 Deinstallation

Um XLogo zu deinstallieren, ist alles, was gemacht werden muss, die Datei `XLogo.jar` und die Konfigurationsdatei `.xlogo` zu löschen, die in Ihrem Home−Verzeichnis zu finden ist (`/Home/Ihr_Anmeldename`) für Linux−Benutzer, oder `c:\windows\.xlogo` für Windows−Benutzer.

7.2 Lesezeichen

Für die aktuelle Version und Fehlerkorrekturen besuchen Sie ab und zu die XLogo−Site.

Fühlen Sie sich frei, sich mit dem Verfasser in Verbindung zu setzen, wenn Sie ein Problem bei der Installation oder Benutzung haben. Alle Vorschläge sind willkommen.

XLogo Handbuch (siehe Quelle)

Übersetzt von Michael Malien	Erzeugt durch Txt2Tags
Geändert am: 03/27/08 13:15:38	am 2008−07−28 um 12:15:31

XLogo Handbuch

8 FAQ − Tricks und wissenswerte Dinge
geändert am 27.03.2008 13:15

8 FAQ − Tricks und wissenswerte Dinge

8.1 Obwohl ich eine Prozedur aus dem Editor lösche, läuft sie weiter

Wenn Sie aus dem Editor gehen, wird gerade gespeichert oder aktualisiert, was im Editor steht. Der einzige Weg in XLogo eine Prozedur zu löschen geht mit dem Primitiv `lösche` oder `er`.

Beispiel: `lösche "toto # löscht die Prozedur toto`

8.2 Ich benutze die Version in Esperanto, aber ich kann nicht mit Sonderzeichen schreiben!

Wenn Sie in die Befehlszeile oder in den Editor tippen, wird, wenn Sie mit dem rechten Knopf klicken, ein rollender Bildschirm erscheinen. In diesem Menü können Sie die traditionellen Editierfunktionen finden (ausschneiden, kopieren, einfügen) und die Esperanto−Sonderzeichen, wenn diese Sprache ausgewählt wird.

8.3 Im Sound Tab der Hilfsmittel−Dialogbox kann kein Instrument gefunden werden.

Traurig ist, dass dieses Problem schon erkannt worden ist. Es ist wegen der Virtuellen Maschine von Java. Dieses Problem tritt völlig zufällig auf. In meinem Fall habe ich zum Beispiel einen Computer, der mit Linux und Windows 98 arbeitet.

Mit Windows 98 erscheint die Liste nicht und mit Linux tut sie es!! Und ich benutze die gleiche JRE und habe kein Hardware−Problem. Dies kann sich von einer JRE−Version zur anderen ändern.

8.4 Ich habe Probleme den Schirm zu aktualisieren, wenn der Igel zeichnet.

Dies ist auch ein bekanntes Problem des JRE. Ich werde versuchen mich damit in der Zukunft zu befassen. Ich könnte in der Lage sein etwas zu tun. Bis jetzt gibt es zwei Lösungen:

- Das Fenster zu minimieren und seine Größe wieder zu verhöhen.
- Das auf der Website vorgeschlagene JRE 1.4.1_07 zu benutzen. Mit JRE > 1.5 geht es besser.

8.5 Wie tippt man schnell ein früher benutztes Kommando?

- Erste Methode: Klicken Sie mit der Maus auf die Zeile der Befehlsgeschichte, wird das Kommando sofort auf der Kommandozeile wieder erscheinen.

- Zweite Methode: Mit der Tastatur erlauben die Auf− und Abwärtspfeile durch die Liste der letzten Kommandos zu navigieren, die getippt worden sind (sehr praktisch).

8.6 Wie kann Ihnen geholfen werden?

- Indem jeder beobachteter Fehler berichtet wird. Wenn Sie ein beobachtetes Problem reproduzieren können, ist es umso besser.
- Jeder Vorschlag zum Verbessern des Programms ist willkommen.
- Durch Hilfe beim Übersetzen: besonders in Englisch ...
- Ein bisschen moralische Unterstützung wird immer gerne gesehen!

XLogo Handbuch (siehe Quelle)

Übersetzt von Michael Malien	Erzeugt durch Txt2Tags
Geändert am: 03/27/08 13:15:38	am 2008−07−28 um 12:15:38

XLogo Handbuch

5 Listen von Grundwörtern
geändert am 28.07.2008 11:35

5 Listen von Grundwörtern

5.1 Bewegung des Igel, Stift− und Farbeinstellungen

- Wie oben schon gesagt wurde, wird der Igel mittels interner Befehle kontrolliert, die **Primitive** heißen. Die folgenden Abschnitte zeigen diese Primitive:

5.1.1 Den Igel bewegen

Diese erste Tabelle zeigt die Primitive auf, die die Bewegung des Igel bestimmen.

Deutsch	Englisch	Argumente	Verwendung
vw, vorwärts	fd, forward	n : Anzahl Schritte	Bewegt den Igel n Schritte vorwärts in die Richtung in die er gerade zeigt.
rw, rückwärts	bk, back	n: Anzahl Schritte	Bewegt den Igel n Schritte rückwärts in die Richtung in die er gerade zeigt.
re, rechts	rt, right	n: Winkel	Dreht den Igel n Grad nach rechts relativ zu der Richtung in die er gerade zeigt.
li, links	lt, left	n: Winkel	Dreht den Igel n Grad nach links relativ zu der Richtung in die er gerade zeigt.
kreis	circle	R: Zahl	Zeichnet einen Kreis vom Radius R um den Igel.
arc, bogen	arc	R cap1 cap2: Zahlen	Zeichnet einen Bogen vom Radius R um den Igel. Dieser Bogen wird eingeschrieben zwischen cap1 und cap2.
heim	home	keine	Setzt den Igel auf die Anfangsposition, d.h. auf die Koordinaten [0 0] mit der Richtung nach 0 Grad.
auf	setpos, setposition	[x y] Liste zweier Zahlen	Bewegt den Igel zu den Koordinaten, die durch die zwei Zahlen in der Liste angegeben sind (x bezeichnet die x−Achse und y die y−Achse)
sx, setzex	setx	x: x−Achse	Bewegt den Igel horizontal zu dem Punkt X auf der x−Achse.
sy, setzey	sety	y: y−Achse	Bewegt den Igel horizontal zu dem Punkt Y auf der y−Achse.
sxy, setzexy	setxy	x y: x−Koordinate gefolgt von der y−Koordinate	Identisch zu `auf [x y]`
ak, aufkurs	setheading	n: Richtung	Orientiert den Igel in die angegebene Richtung. 0 bedeutet, dass der Igel nach oben zeigt.

Deutsch	Englisch	Argumente	Verwendung

igeltext	label	a: Wort oder Liste	Zeichnet das angegebene Wort oder Liste an der Position des Igel, entsprechend der Richtung in die er zeigt. Z.B.: schreibt `igeltext [Hallo Welt!]` den Satz "Hallo Welt!" wo immer der Igel ist, entsprechend seiner Lage oder Richtung.
litext, längeigeltext	labellength	a: Wort oder Liste	Ergibt die mit dem Primitiv `igeltext` in der aktuellen Schriftart zum Darstellen erforderliche Länge eines Wortes oder einer Liste.
punkt	dot	a: Liste	Der Punkt mit den Koordinaten in der Liste wird hervorgehoben (in der Stiftfarbe).|

Die zweite Tabelle zeigt die Primitive auf, die Eigenschaften des Igel einstellen. Sie steuern zum Beispiel, ob der Igel auf dem Schirm sichtbar sein soll oder in welcher Farbe er zeichnen soll, wenn er sich bewegt.

Deutsch	Englisch	Argumente	Verwendung
zi, zeigeigel	st, showturtle	keine	Macht den Igel auf dem Schirm sichtbar.
vi, versteckeigel	ht, hideturtle	keine	Macht den Igel auf dem Schirm unsichtbar.
lb, löschebild	cs, clearscreen	keine	Leert den Zeichenbereich.
wasche	wash	keine	Löscht den Zeichenbereich aber belässt den Igel an dem Ort.
reset	resetall	keine	Initialisiert die XLogo Schnittstelle mit den Standardwerten (Stift−Farbe: schwarz, Schirmfarbe: weiß, Animationsmodus: aus, Text und Graphik Schriftart: Dialog 12 Punkt, Stift−Form: Quadrat, Zeichnungsqualität: normal, Igel erlaubt: 16, Einzelschrittmodus: aus, Schirmgröße [1000 1000]) und leert die Zeichenfläche.
sa, stiftab	pd, pendown	keine	Der Igel wird eine Linie zeichnen, wenn er sich bewegt.
sh, stifthoch	pu, penup	keine	Der Igel wird eine Linie zeichnen, wenn er sich bewegt.
rd, radiere	pe, penerase	keine	Der Igel wird alle Markierungen radieren auf die er trifft.
umkehrstift	px, penreverse	keine	Senke den Stift und setze den Igel in Invertierungsmodus.
ns, normalstift	ppt, penpaint	keine	Senke den Stift und setze den Igel in Zeichnungsmodus.
sstf, setzestiftfarbe	setpc, setpencolor	a: Ganzzahl oder Liste [r g b]	Setzt die Stiftfarbe. Siehe Seite 22 .
sbf setzebildfarbe	setsc, setscreencolor	a: Ganzzahl oder Liste [r g b]	Setzt die Schirmfarbe. Siehe S.22 .

pos	pos, position	keine	Ergibt die aktuelle Position des Igel, z.B: `pos` ergibt [10 −100]
kurs	heading	keine	Ergibt the Lage oder Richtung des Igel (siehe `aufkurs`)
ri, richtung	towards	a: Liste	Die Liste muss zwei Zahlen als Koordinaten enthalten. Ergibt die Richtung, welcher der Igel folgen muß, um den Punkt zu erreichen, der in der Liste angegeben ist.

Deutsch	Englisch	Argumente	Verwendung
abst, abstand	distance	a: Liste	Die Liste muß zwei Zahlen als Koordinaten enthalten. Ergibt die Anzahl von Schritten zwischen der aktuellem Position und dem Punkt, der durch die Koordinaten in der Liste definiert ist.
sf, stiftfarbe	pc, pencolor	a: Liste	Ergibt die aktuelle Farbe des Stiftes. Die Farbe ist angegeben durch die Liste [r g b] , wo `r` die rote, `b` die blaue und `g` die grüne Komponente ist.
bf, bildfarbe	sc, screencolor	a: Liste	Ergibt die aktuelle Schirmfarbe (Hintergrund). Diese Farbe wird angegeben durch die Liste `[r g b]` , wo `r` die rote, `b` die blaue und `g` die grüne Komponente ist.
fen, fenster	window	keine	Der Igel kann sich außerhalb der Zeichenfläche bewegen, kann aber dort natürlich nicht zeichnen.
rspr, randsprung	wrap	keine	Wenn der Igel die Zeichenfläche verläßt, wird er auf der gegenüber liegenden Seite erscheinen!
perspektivisch	perspective	keine	Der Igel kann sich durch den 3D−Raum bewegen. (Siehe Abschnitt 5.1.1 für diesen Modus). Um diesen Modus zu

			verlassen, benutzen Sie einen der Primitive fenster, randsprung oder zaun
zaun	fence	keine	Der Igel ist auf die Zeichenfläche begrenzt. Wenn er dabei ist sie zu verlassen, wird eine Fehlermeldung angezeigt und die maximale Anzahl von Schritten ausgegeben, die der Igel sich bewegen kann bevor er den Zaun erreicht (bis 1 oder 2 Schritte ...).
ff, findefarbe	fc, findcolor	a: Liste	Ergibt die Farbe eines Pixel. Diese Farbe wird bestimmt durch eine [r g b] Liste, wo r die rote, b die blaue und g die grüne Komponente ist.
ssb, setzestiftbreite	setpw, setpenwidth	n: Zahl	Definiert die Dicke der Stiftspitze in Pixel. Standard ist 1. Der Stift hat eine quadratische Spitze. (Andere Formen werden in zukünftigen Versionen zur Verfügung gestellt.)
sb, stiftbreite	pw, penwidth	keine	Ergibt die Dicke der Stiftspitze in Pixel.
sstform, setzestiftform	setps, setPenShape	0 oder 1	Setzt die Stiftform. 0 Quadrat. 1 Oval
sf, stiftform	ps, penshape	keine	Ergibt die Stiftform. 0 Quadrat. 1 Oval
szq, szeichnungsqualität	setdq, setDrawingQuality	0, 1 oder 2	Setzt die Zeichnungsqualität: 0 normal, 1 hoch und 2 niedrig
zq, zeichnungsqualität	dq, DrawingQuality	keine	Ergibt die Zeichnungsqualität 0 normal, 1 hoch und 2 niedrig
sbg, setzebildgröße	setscreensize	[Breite Höhe]	Setzt die Schirmgröße auf die Dimension, die in der Liste enthalten ist, z.B. `setzebildgröße [1000 1000]`

bg, bildgröße	screensize	Liste	Ergibt die aktuelle Schirmgröße als Liste, z.B. `setzebildgröße [1000 1000] bildgröße`
sform, setzeform	setshape	n: Zahl	Sie können den bevorzugten Igel auf dem zweiten Reiter im Menü Hilfsmittel − Einstellungen setzen. Der bevorzugte Igel kann auch gesetzt werden mit `setshape`. Die Zahl n kann zwischen 0 und 6 liegen. 0 ist die dreieckige Form.
form	shape	keine	Ergibt die Zahl, die für die Form des Igel steht.
setzefontgröße	setfs, setfontsize	n: Zahl	Setzt Größe und Schriftart, wenn Sie auf dem Schirm mit dem Primitiv `igeltext` schreiben. Standardgröße der Schriftart ist 12 Punkt.

5.1.2 Ein Wort zu Farben

Farben werden in XLogo durch eine Liste dreier Zahlen `[r g b]` zwischen 0 und 255 definiert. Die Zahl `r` ist die `rote` Komponente, `b` das `Blau` und `g` das `Grün` . XLogo hat 16 vordefinierte Farben: Sie können sie einstellen, entweder mit ihrer RGB−Liste, mit Hilfe ihrer Nummer oder mit einem Primitiv. Sehen Sie sich diese Tabelle an:

Zahl	Deutsch	Englisch	R G B	Farbe
0	schwarz	black	0 0 0	
1	rot	red	255 0 0	
2	grün	green	0 255 0	
3	gelb	yellow	255 255 0	
4	blau	blue	0 0 255	
5	magenta	magenta	255 0 255	
6	cyan	cyan	0 255 255	
7	weiß	white	255 255 255	

8	grau	gray	128 128 128	
9	hellgrau	lightgray	192 192 192	
10	dunkelrot	darkred	128 0 0	
11	dunkelgrün	darkgreen	0 128 0	
12	dunkelblau	darkblue	0 0 128	
13	orange	orange	255 200 0	
14	rosa	pink	255 175 175	
15	violett	purple	128 0 255	
16	braun	brown	153 102 0	

Diese drei Anweisungen sind die gleichen:

```
setzebildfarbe orange sbildfarbe 13 sbf [255 200 0]
```

5.1.3 Animations−Modus

Es gibt zwei Primitive, die den Igel Befehle ausführen lassen ohne dass sie angezeigt werden: **`animation`** und **`stoppeanimation`**

Deutsch	Englisch	Argumente	Verwendung
anim, animation	anim, animation	keine	Sie gehen in den Animationsmodus. Der Igel zeichnet nicht mehr auf den Schirm, folgt aber der Linie. Um die Zeichnung auf dem Schirm zu aktualisieren, benutzen Sie das Primitiv `auffrischen`. Das ist sehr nützlich zum Erzeugen einer Animation oder um eine Linie schneller zu zeichnen.
stoppeanim, stoppeanimation	stopanim, stopanimation	keine	Animationsmodus ist beendet: Sie schalten zurück in den klassischen Modus. Sie können die Bewegungen des Igel auf dem Schirm sehen.
af, auffrischen	refresh	keine	Aktualisiert den Schirm im Animationsmodus: Das Bild des Zeichengebiets wird aktualisiert.

Damit Sie den Animationsmodus bemerken, erscheint ein Kamera−Icon in der Befehlsgeschichte. Wenn Sie auf das Icon klicken, wird der Animationsmodus stoppen. Das ist gleichwertig zu dem Primitiv `stoppeanimation` .	

5.1.4 Text in der Textbereichs schreiben

Diese Tabelle zeigt die Primitive auf, die die Eigenschaften des Textbereichs einstellen. Jene Primitive, die die Farbe und die Größe der Textbereichs kontrollieren, sind nur verfügbar für die Primitive **`druckezeile`** und **`schreibe`**

Deutsch	Englisch	Argumente	Verwendung
lt, löschetext	ct, cleartext	keine	Leert das Gebiet, das Kommando− und Kommentargeschichte enthält.
dz, druckezeile	pr, print	Wort, Liste oder Zahl	Zeigt das Argument in der Geschichtszone an. `pr "abcd` −−−−−−−−> abcd ; `pr [1 2 3 4]` −−−−> 1 2 3 4 ; `pr 4` −−−−−−−−−−−> 4
schreibe	write	Wort, Liste oder Zahl	Dasselbe für das Primitiv `druckezeile`, aber ohne Zeilenvorschub.
stg, setzetextgröße	setTS, setTextSize	a: Zahl	Definiert die Schriftartgröße im Kommandofenster. Nur gültig mit dem Primtiv `druckezeile`.
stf, stfarbe, setzetextfarbe	setTC, setTextColor	a: Zahl oder Liste	Definiert die Schriftfarbe im Kommandofenster. Nur gültig mit dem Primtiv `druckezeile`. Siehe Seite 22.
stn, setzetextname	setTN, setTextName	n: Zahl	Wählt die Schrift mit Zahl n, wenn Sie im Kommandofenster mit dem Primitiv `druckezeile` schreiben. Sie können die Verbindung zwischen der Nummer und der Schrift im Menü Hilfsmittel−Einstellungen auf dem Reiter Schriftart finden.
sstil, setzestil	setsty, setstyle	Liste oder Wort	Setzt das Format of the police im Textgebiet. Sie können zwischen sieben Stilen wählen: `kein`, `fett`, `kursiv`, `durchgestrichen`, `unterstrichen`, `hochgestellt`, `heruntergestellt`. Wenn Sie mehrere Stile zusammen wollen, schreiben Sie sie in einer Liste. Sehen Sie sich die Beispiele nach dieser Tabelle an.
stil	sty, style	keine	Ergibt eine Liste, welche die unterschiedlichen Stile für das Primitiv `druckezeile` enthält.

Ein paar Beispiele für das Formatieren von Text:

```
setzestil [Fett Unterstrichen] druckezeile "Hallo
```

hallo

```
sstil "Durchgestrichen schreibe [Text durchgestrichen] sstil "Kursiv schreibe "\ x sstil
```

~~gestrichen~~ x^2

Deutsch	Englisch	Argumente	Verwendung

fontgröße	fontsize	keine	Ergibt die Größe der Schriftart, wenn Sie mit dem Primitiv `igeltext` auf den Schirm schreiben.
setzefontname	setfn, setfontname	n: Zahl	Wählt die Nummer n der Schriftart, wenn Sie mit dem Primitiv `igeltext` auf den Schirm schreiben. Sie können die Verbindung zwischen Zahl und Schriftart im Menü Hilfsmittel, Einstellungen auf dem Reiter Schriftart finden.
fontname	fontname	keine	Ergibt eine Liste mit zwei Elementen. Das erste Element ist eine Zahl, die die Schriftart angibt, wenn Sie auf den Schirm mit dem Primitiv `igeltext` schreiben. Das letzte Element ist eine Liste, welche den Namen der Schriftart enthält.
setzetrenner	setsep, setseparation	a: Zahl	Bestimmt das Verhältnis zwischen Graphikschirm und history zone. Die Zahl a muss zwischen 0 und 1 liegen. Wenn a 1 ist, wird das Zeichengebiet den ganzen Raum einnehmen, wenn a gleich 0 ist, wird die history zone das ganze Fenster einnehmen.
trenner	sep, separation	keine	Ergibt das aktuelle Verhältnis zwischen Zeichengebiet und history zone.
gitter	grid	a, b Ganzzahl	Zeichnet ein Gitter. Jedes Quadrat hat die Dimension a und b.
stoppegitter	stopgrid	keine	Löscht das Gitter.
sgf, setzegitterfarbe	setgridcolor	Farbe	Erlaubt den Benutzer eine eigene Farbe für das Gitter zu wählen, z.B: `setzegitterfarbe rot`
gitterfarbe	gridcolor	keine	Ergibt die aktuelle Farbe des Gitter.
gitter?	grid?	keine	Ergibt wahr, wenn das Gitter gezeichnet wird,

			sonst falsch.
achsen	axis	n: Ganzzahl	Zeichnet horizontale und vertikale Achsen. Der Abstand zwischen zwei Teilstrichen beträgt n Schritte.
xachse	xaxis	n: Ganzzahl	Zeichnet nur die horizontale Achse. Der Abstand zwischen zwei Teilstrichen beträgt n Schritte.
yachse	yaxis	n: Ganzzahl	Zeichnet nur die vertikale Achse. Der Abstand zwischen zwei Teilstrichen beträgt n Schritte.
stoppeachsen	stopaxis	keine	Löscht beide Achsen.
saf, setzeachsenfarbe	setaxiscolor	Farbe	Erlaubt den Benutzer eine Farbe für die Achsen zu verwenden, z.B. `setzeachsenfarbe grün`
achsenfarbe	axiscolor	keine	Ergibt die aktuelle Farbe der Achse.
xachse?	xaxis?	keine	Ergibt `wahr`, wenn die horizontale Achse gezeichnet wird, sonst `falsch`.
yachse?	yaxis?	keine	Ergibt `wahr`, wenn die vertikale Achse gezeichnet wird, sonst `falsch`.
zoom	zoom	a	Zoome den Zeichenschirm. Die Zahl a stellt die Skalierung zur originalen Bildgröße dar, wie festgelegt in den Einstellungen.
fenstergröße	zonesize	keine	Ergibt eine Liste aus vier Zahlen. Diese Ganzzahlen sind die Koordinaten der linken oberen Ecke der Zeichenzone und die Koordinaten der rechten unteren Ecke.
nricht, nachricht	msg, message	a: Liste	Zeigt die Meldung der Liste in einer Dialogbox, das Programm stoppt bis der Benutzer den Button "OK" klickt.

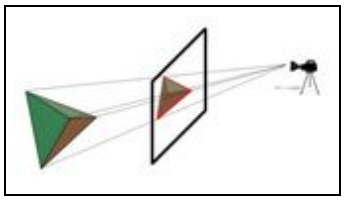
5.1.5 Der Igel in 3D

Beginnend mit Version 0.9.92 kann unser Igel die Ebene verlassen und sich im 3D−Raum bewegen. Wir benutzen das Primitiv `perspektivisch`, um dorthin zu schalten. Willkommen in der 3D−Welt!

Die perspektivische Projektion

Um einen 3D−Raum in einer 2D−Ebene darzustellen, benutzt XLogo eine perspektivische Projektion. Eine Kamera sieht auf die 3D−Szene und angezeigt wird das Bild des Projektionsschirms. Hier ist ein kleines Schema, das dies erklären soll:

Einige Primitive erlauben die Kameraposition zu setzen, der Projektionsschirm liegt auf halber Entfernung zur Kamera.



Orientierung in 3D−Welt verstehen

In der Ebene wurde die Orientierung des Igels nur durch seinen Kurs (**`kurs`**) definiert. In der 3D−Welt ist die Orientierung des Igels durch drei Winkel gegeben:

- **`Rollwinkel`**: Die Steigung des Igel um die Achse Oy.
- **`Neigung`**: Die Steigung des Igel um die Achse Ox.
- **`Kurs`**: Die Steigung des Igel um die Achse Oz.

Tatsächlich ist der Igel sehr ähnlich zu einem Jet, wenn er sich selbst in der 3D−Welt bewegt. Hier ist eine kleine Illustration, welche diese drei Werte darstellt:

Es erscheint zunächst sehr schwierig zu sein, aber Sie werden sehen, dass die Dinge sehr ähnlich bleiben zu den Bewegungen in der 2D−Ebene. Hier sind die Grund−Primitive für das Bewegen in der 3D−Welt:

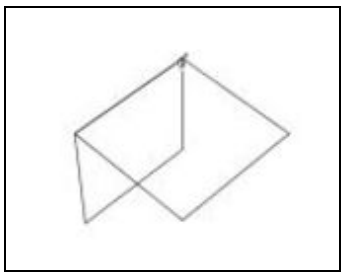
- **`vorwärts`** : Gleiches Verhalten wie in der 2D−Ebene
- **`rechts`** : Gleiches Verhalten wie in der 2D−Ebene
- **`rollerechts`** : Der Igel dreht sich um n Grad rechts um die longitudinale Achse
- **`rollelinks`** : Der Igel dreht sich um n Grad links um die longitudinale Achse
- **`rauf`** : Der Igel geht um n Grad hoch um seine transversale Achse
- **`runter`** : Der Igel geht um n Grad hinab um seine transversale Achse

Wenn wir in der 2D−Ebene ein 200 Schritte breites Quadrat zeichnen wollen, schreiben wir:

```
wiederhole 4 [vorwärts 200 rechts 90]
```

Diese Anweisungen sind auch in der 3D−Welt verfügbar und das Quadrat wird in dem Modus `perspektivisch` gezeichnet. Wenn der Igel `90` Grad herunter geht, können wir ein weiteres Quadrat zeichnen und wir erhalten:

```
löschebild
wiederhole 4 [vorwärts 200 rechts 90]
runter 90
wiederhole 4 [vorwärts 200 rechts 90]
```



Sie müssen gerade einmal ein paar Beispiele probieren, um diese Orientierung zu verstehen und werden ein Experte! Dann werden Sie erkennen, wie die drei Rotations−Primitive verbunden sind. Zum Beispiel versuchen Sie das:

```
löschebild
rollelinks 90 rauf 90 rollerechts 90
```

Das Bewegen des Igel ist äquivalent zu `links 90`.

Verfügbare Primitive im 2D− und 3D−Modus

Die folgenden Primitive sind verfügbar in der Ebene und in der 3D−Welt. Der einzige Unterschied sind die Argumente, die diese Primitive empfangen. Zum Beispiel warten die Primitive `setpos` und `setposition` immer noch auf ein Listenelement als Argument. In 3D muss diese Liste aus drei Zahlen bestehen, welche die drei Punktkoordinaten darstellen. Hier sind diese Primitive:

`kreis`	`bogen`	`heim`	`richtung`
`abstand`	`setzeposition`	`setzex`	`setzey`
`setzerichtung`	`igeltext`	`längeigeltext`	`punkt`
`pos`	`richtung`	.	.

Nur im 3D−Modus verfügbare Primitive

- **`setzexyz`** : Dieses Primitiv bewegt den Igel zum gewählten Punkt. Dieses Primitiv wartet auf drei Argumente, die die Punktkoordinaten darstellen. `setzexyz` ist sehr ähnlich zu `setzeposition` aber die Koordinaten stehen dort nicht als Liste.

  Zum Beispiel bewegt `setzexyz minus 100 200 50` den Igel zu dem Punkt x=−100; y=200; z=50

- **`setzez`** : Dieses Primitiv bewegt den Igel zu dem Punkt mit dem gültigen Wert für z. `setzez` wartet auf eine Zahl als Argument. Dieses Primitiv ist vergleichbar zu `setzex` und `setzey`.

- **`setzeorientierung`** : Setzt die Orientierung des Igel. Dieses Primitiv wartet auf eine Liste, welche drei Zahlen enthält: den Rollwinkel, die Neigung und die Kurs.

Zum Beispiel setzt `setzeorientierung [100 0 58]` den Rollwinkel `100`, die Neigung auf `0` und den Kurs auf `58` Grad.

- **`Orientierung`** : Gibt die Orientierung des Igel in einer Liste zurück: `Rollwinkel , Neigung , Kurs`. Beachten Sie die Zahlenreihenfolge. Zum Beispiel ergibt `100 , 20 , 90` dieselbe Orientierung, wenn nach der Anweisung `löschebild` von der Ursprungsposition ausgegangen wird:

```
rollerechts 100 rauf 20 rechts 90
```

Wenn Sie die Reihenfolge der Anweisungen ändern, erhalten Sie nicht die gültige Orientierung!

- **`setzerollwinkel`** : Der Igel dreht sich um die longitudinale Achse und erhält den gewählten Winkel.
- **`rollwinkel`** : Ergibt den aktuellen Wert des Rollwinkel.
- **`setzeneigung`** : Der Igel dreht sich um die transversale Achse und erhält die gewählte Neigung.
- **`neigung`** : Ergibt den aktuellen Wert der Neigung.

3D Betrachter

Ein 3D−Betrachter ist in XLogo enthalten und erlaubt es Ihre Zeichnungen in 3D darzustellen. Dieses Modul verwendet die Java3D Bibliotheken, so dass es notwendig ist Java3D voll installiert zu haben.

Hier sind die Regeln zur Verwendung des 3D−Betrachters:

Wenn wir geometrische Figuren im Zeichenbereich erschaffen wollen, müssen wir den 3D−Betrachter anweisen, welche Formen wir für zukünftige Darstellungen aufzeichnen wollen. Es ist möglich Vielecke/Vieleckflächen, Linien, Punkte und Text aufzuzeichnen.

Hier sind die Primitive:

- **`vieleckanfang`** : Die als nächstes folgenden Igelbewegungen werden gespeichert, um ein Vieleck zu erzeugen.

- **`vieleckende`** : Seit dem letzten Aufruf von `vieleckanfang` hat der Igel mehrere Ecken angesteuert. Dieses neue Vieleck wird aufgezeichnet, seine Farbe wird bestimmt durch alle Farben der Eckpunkte. Dieses Primitiv vollendet das Vieleck.

- **`linienanfang`** : Die als nächstes folgenden Igelbewegungen werden gespeichert, um eine Linie zu erzeugen.

- **`linienende`** : Seit dem letzten Aufruf von `linienanfang` hat der Igel mehrere Ecken angesteuert. Diese neue Linie wird aufgezeichnet, seine Farbe wird bestimmt durch alle Farben der Eckpunkte. Dieses Primitiv vollendet die Linie.

- **`punkteanfang`** : Die als nächstes folgenden Igelbewegungen werden gespeichert, um eine Punktemenge zu erzeugen.

- **`punkteende`** : Dieses Primitiv beendet die Punktemenge.

- **`textanfang`** : Jedesmal wenn der Benutzer mit dem Primitiv `igeltext` einen Text im Zeichenbereich anzeigt, wird er aufgezeichnet und dann im 3D−Betrachter angezeigt.

- **`textende`** : Beendet die Texteaufzeichnung.

- **`view3d vieleckansicht`** : Startet den 3D−Betrachter. Alle aufgezeichneten Objekte werden in dem neuen Fenster gezeichnet. Sie können die Kamera steuern:
  - ♦ Sie können die Szene drehen, in dem Sie auf die linke Maustaste klicken.
- Sie können die Szene übersetzen, in dem Sie auf die rechte Maustaste klicken.
- Sie können die Szene zoomen, in dem Sie das Mausrad verwenden.

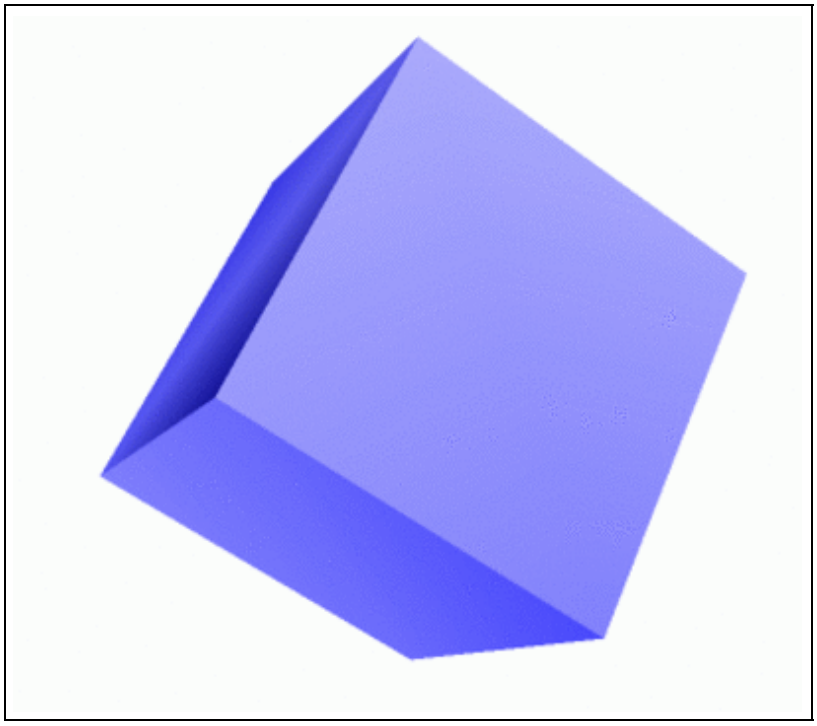
Zeichnen eines Würfels

Alle Seitenflächen bestehen aus 400 Schritten breiten Quadraten. Hier ist das Programm:

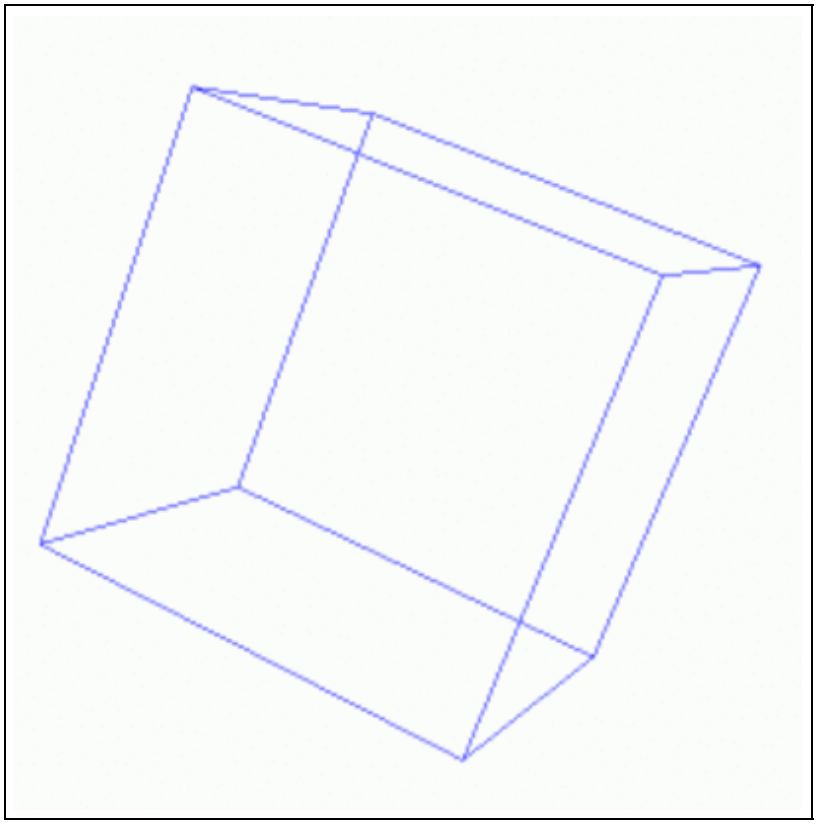
```
lerne quadrat
  # we record the vertice square
  vieleckanfang wiederhole 4[vorwärts 400 rechts 90] vieleckende
Ende

lerne einfacherWürfel
  # gelb cube
  löschebild perspektivisch setzestiftfarbe gelb
  # lateral faces
  wiederhole 4[quadrat stifthoch rechts 90 vorwärts 400 links 90 rollerechts 90 stiftab]
  # bottom face
  runter 90 quadrat rauf 90
  # upper face
  vorwärts 400 runter 90 quadrat
  # visualization
  vieleckansicht
Ende
```

Wir starten das Kommando `einfacherWürfel`:



Wenn wir in der Prozedur `Quadrat` `vieleckanfang` durch `linienanfang` und `vieleckende` durch `linienende` ersetzen:



Hätten wir `punkteanfang` und `punkteende` statt `linienanfang` und `linienende` verwendet, sollten wir auf dem Schirm nur die acht Würfelecken sehen. Diese Primitive sind sehr wichtig zum Darstellen von Punktemengen in 3D.

5.2 Arithmetische und logische Operationen

Deutsch	Englisch	Argumente	Verwendung
summe	sum	a b: Zahlen	Addiert die zwei Zahlen a und b und gibt das Ergebnis zurück, z.B.: `summe 40 60` ergibt 100
differenz	difference	a b: Zahlen	Ergibt a `-` b, z.B.: `differenz 100 20` ergibt 80
minus	minus	a : Zahl	Ergibt den negativen Wert von a, z.B.: `minus 5` ergibt −5. Siehe die Anmerkung am Ende dieser Tabelle.
produkt	product	a b: Zahlen	Ergibt das Ergebnis der Multiplikation von a und b.
teile	div, divise	a b: Zahlen	Ergibt das Ergebnis der Division von a und b, z.B.: `teile 3 6` ergibt 0.5
quotient	quotient	a b: Zahlen	Ergibt den Quotient von a und b, z.B.: `quotient 15 6` ergibt 2
rest	mod, modulo	a b: ganze Zahlen	Ergibt den Rest der Division von a und b.

runde	rnd, round	a: Zahl	Ergibt die nächste ganze Zahl zur Zahl a, z.B.: `round 6.4` ergibt 6
ganzzahl	integer	a: Zahl	Ergibt den ganzzahligen Teil der Zahl, z.B.: `ganzzahl 8.9` ergibt 8 ; `integer 6.8` ergibt 6
potenz	power	a b: Zahlen	Ergibt die Potenz 'a hoch b', z.B.: `power 3 2` ergibt 9
qw	sqrt, squareroot	n: Zahl	Ergibt die Quadratwurzel von n.
log10	log10	n : Zahl	Ergibt den Zehnerlogarithmus von n
sin, sinus	sin, sine	a: Zahl	Ergibt den Sinus von a. (a wird in Grad angegeben)
cos, cosin	cos, cosine	a: Zahl	Ergibt den Cosinus von a. (a wird in Grad angegeben)
tan	tan, tangent	a: Zahl	Ergibt den Tangens von a. (a wird in Grad angegeben)
acos, arccos	acos, arccosine	a: Zahl	Ergibt den Winkel im Bereich 0 bis 180 dessen Cosinus a ist.
asin, arcsin	asin, arcsine	a: Zahl	Ergibt den Winkel dessen Sinus a ist.
atan, arctan	atan, arctangent	a: Zahl	Ergibt den Winkel dessen Tangens a ist.
pi	pi	aucun	Ergibt die Zahl 3.141592653589793
zz, zufallszahl	random, ran	n: ganze Zahl	Ergibt eine Zufallszahl zwischen `0` und `n - 1`.
ovz, ohnevorzeichen, betrag	absolute, abs	n: Zahl	Ergibt den Absolutbetrag (den Zahlenwert ohne Vorzeichen) einer Zahl.

Wichtig: Seien Sie sorgfältig mit den Primitiven, die zwei Parameter benötigen!

```
Z.B.:
  ``setzexy a b`` , wenn b negativ ist,
  zum Beispiel bei ``setxy 200-10``
```

Der Logointerpreter wird die Operation `200-10` durchführen (d. h. es wird `10` von `200` subtrahiert). Er wird deswegen schlussfolgern, dass es nur einen Parameter (`190`) gibt, aber zwei benötigt, und wird daher eine Fehlermeldung erzeugen. Um diese Art von Problem zu vermeiden, benutzen Sie das Primitiv `minus` , um die negative Zahl zu spezifizieren:

- `setxy 200 minus 10` .

  Dies ist eine Liste von logischen Operatoren:

Deutsch	Englisch	Argumente	Verwendung
oder, eines?	or	b: boolesch	Ergibt `wahr`, wenn a oder b `wahr` sind, sonst `falsch`
und, alle?	and	b: boolesch	Ergibt `wahr`, wenn a und b `wahr` sind, sonst `falsch`

nicht	not	a :boolesch	Ergibt die Negation von a. Wenn a `wahr` ist, `falsch`. Wenn a `falsch` ist, `wahr`.

5.3 Operationen auf Listen und Wörter

Deutsch	Englisch	Argumente	Verwendung
wort	word	a b	Verbindet zwei Wörter a und b, z.B.: `dz word "a 1` ergibt a1
liste	list	a b	Ergibt eine Liste bestehend aus a und b, zum Beispiel, `liste 3 6` ergibt [3 6]. `liste "a "Liste` ergibt [a Liste]
satz	se, sentence	a b	Ergibt eine Liste bestehend aus a und b. Wenn a oder b Listen sind , dann wird jedes Element von a und b ein Element der Ergebnisliste (Quadratische Klammern werden entfernt), z.B.: `satz [4 3] "Hallo` ergibt [4 3 Hallo] ; satz [Wo sind] "Dinge ergibt [Wo sind Dinge]
me, miterstem	fput	a b: a irgendwas, b Liste	Fügt a vor dem ersten Element von Liste b ein, z.B: `me "cocoa [2]` ergibt [cocoa 2]
ml, mitletztem	lput	a b: a irgendwas, b Liste	Fügt a hinter dem letzten Element von Liste b ein, z.B.: `ml 5 [7 9 5]` ergibt [7 9 5 5]
umdrehliste	reverse	a : Liste	Kehrt die Reihenfolge der Elements in List a um, z.B.: `umdrehliste [1 2 3]` ergibt [3 2 1]
nehme, nehmewas	pick	a : ein Wort oder Liste	Wenn a ein Wort ist, gibt sie einen zufälligen Buchstaben von a zurück. Wenn a eine Liste ist, gibt sie ein zufälliges Element von a zurück.
entferne	remove	a b: a irgendwas, b Liste	Entfernt Element a von Liste b, wenn a vorkommt, z.B.: `entferne 2 [1 2 3 4 2 6 ]` ergibt [1 3 4 6]

Deutsch	Englisch	Argumente	Verwendung
element	item	a b: a ganze Zahl, b List oder Wort	Wenn b ein Wort ist, ergibt sie den Buchstaben an Position a im Wort (1 stellt den ersten Buchstaben dar.). Wenn b eine Liste ist, ergibt sie das Element an Position n in der Liste.
ol, ohneletztes	bl, butlast	a: Liste oder Wort	Wenn a eine Liste ist, ergibt sie die ganze Liste außer ihr letztes Element. Wenn a ein Wort ist, ergibt sie das Wort

			minus seinen letzten Buchstaben.
oe, ohneerstes	bf, butfirst	a: Liste oder Wort	Wenn a eine Liste ist, ergibt sie die ganze Liste außer ihr erstes Element. Wenn a ein Wort ist, ergibt sie das Wort minus seinen ersten Buchstaben.
lz, letztes	last	a: Liste oder Wort	Wenn a eine Liste ist, ergibt sie das letzte Element der Liste. Wenn a ein Wort ist, ergibt sie den letzten Buchstaben des Wortes.
erstes	first	a: List oder Wort	Wenn a eine Liste ist, ergibt sie das erste Element der Liste. Wenn a ein Wort ist, ergibt sie den ersten Buchstaben des Worts.
selem, setzeelement	setitem, replace	li1 n li2: li1 Liste, n Ganzzahl, li2 Wort oder Liste	Ersetzt Element n in der Liste li1 durch das Wort oder die Liste li2. `setzelement [a b c] 2 8` −−> [a 8 c]
fez, fügeelementzu	additem	li1 n li2: li1 Liste, n Ganzzahl, li2 Wort oder Liste	Fügt an der Position n in the List li das Wort oder die Liste li2 ein, z.B. `fügeelementzu [a b c] 2 8` −−> [a 8 b c]
zähle	count	a: Liste oder Wort	Wenn a ein Wort ist, ergibt sie die Anzahl von Buchstaben in a. Wenn a eine Liste ist, ergibt sie die Anzahl von Elementen in a.
unicode	unicode	a: Wort	Ergibt den Unicode−Wert von Zeichen "a", z.B.: `dz unicode "A` ergibt 65
zeichen	character, char	a: Zahl	Ergibt das Zeichen dessen Unicode−Wert "a" ist, z.B.: `dz zeichen 65` ergibt "A

5.4 Boolesche Funktionen

Eine boolesche Funktion ist ein Primitiv, das **`Wahr`** oder **`Falsch`** zurückgibt. Diese Primitive enden hier mit einem Fragezeichen. Sie heißen auch Prädikate (engl. predicates), vergleichbar mit den Informationsfunktionen in Excel (Anmerkung des Übersetzers).

Deutsch	Englisch	Argumente	Verwendung
wahr	true	keine	Ergibt `wahr`.
falsch	false	keine	Ergibt `falsch`.

wort?	word?	a	Ergibt `wahr`, wenn a ein Wort ist, sonst `falsch`.
zahl?	number?	a	Ergibt `wahr`, wenn a eine Zahl ist, sonst `falsch`.
ganzzahl?	integer?	a	Ergibt `wahr`, wenn a eine ganze Zahl ist, sonst `falsch`.
liste?	list?	a	Ergibt `wahr`, wenn a eine Liste ist, sonst `falsch`.
leer?	empty?	a	Ergibt `wahr`, wenn a eine leeres Wort oder eine leere Liste ist, sonst `falsch`.
gleich?	equal?	a b	Ergibt `wahr`, wenn a und b gleich sind, sonst `falsch`.

Deutsch	Englisch	Argumente	Verwendung
vorher?	before?	a b : Wörter	Ergibt `wahr`, wenn a alphabetisch vor b liegt, sonst `falsch`.
element?	member?	a b	Ergibt `wahr`, wenn b eine Liste ist und a ein Element von b ist. Ergibt `wahr`, wenn b ein Wort ist und a ein Buchstabe von b ist.
elementab	member	a b	Sucht das Element a in dieser Liste, wenn b eine Liste ist.

Hier gibt es zwei mögliche Fälle:

- Wenn a in b ist, ergibt sie eine Teilliste aller Listenelemente beginnnend mit der ersten Position von a in b.
- Wenn a nicht in b ist, ergibt sie das Wort `falsch`. Wenn b ein Wort ist, sucht sie nach einem Buchstaben a in diesem Wort. Da gibt es zwei Möglichkeiten:

- Wenn a in b ist, ergibt sie den letzten Teil von dem Wort, beginnend mit a.
- Sonst ergibt sie das Wort `falsch`. `elementab "o "cocoa` ergibt `ocoa` ; `elementab 3 [1 2 3 4]` liefert `[3 4]` |

sa? stiftab?	pd?, pendown?	irgendwas	Ergibt das Wort `wahr`, wenn der Stift abgesetzt ist, sonst `falsch`.
sichtbar?	visible?	irgendwas	Ergibt das Wort `wahr`, wenn der Igel sichtbar ist, sonst `falsch`.
grundwort?	prim?, primitive?	a: Wort	Ergibt `wahr`, wenn das Wort ein XLogo−Primitiv ist, sonst `falsch`.
proz?, prozedur?	proc?, procedure?	a: Wort	Ergibt `wahr`, wenn das Wort eine benutzerdefinierte Prozedur ist, sonst `falsch`.

5.5 Einen Ausdruck mit dem Primitiv Wenn testen

Wie in jeder Programmiersprache auch erlaubt es Ihnen auch Logo zu überprüfen, ob einer Bedingung entsprochen wird, und dann den gewünschten Code auszuführen, wenn sie `wahr` oder `falsch` ist.

Mit dem Primitiv **Wenn** ermöglichen Sie solche Tests. Hier ist die Syntax:

```
Wenn ausdruck_test [Liste1] [Liste2]
```

Wenn `ausdruck_test` wahr ist, werden die in `Liste1` eingeschlossenen Anweisungen ausgeführt.

Wenn `ausdruck_test` falsch ist, werden die Anweisungen in `Liste2` ausgeführt. Die zweite Liste ist optional.

Beispiele:

- `Wenn 1+2=3 [druckezeile "wahr][druckezeile "falsch]`
- `Wenn (erstes "XLOGO)="Y [vw 100 re 90] [dz [XLOGO fängt mit einem X an!]]`
- `Wenn (3*4)=6+6 [dz 12]`

5.6 Sich mit Prozeduren und Variablen befassen

5.6.1 Prozeduren

Prozeduren sind eine Art von "Programm". Wenn eine Prozedur aufgerufen wird, werden die Anweisungen im Körper von der Prozedur ausgeführt. Eine Prozedur wird mit dem Schlüsselwort **lerne** definiert.

```
lerne Name_von_Prozedur :v1 :v2 :v3.... [:v4 ....] [:v5 ....]
  Körper der Prozedur
Ende
```

- `Name_von_Prozedur` ist der Name, der der Prozedur gegeben wird.
- `:v1 :v2 :v3` stehen für die Variablen, die innerhalb der Prozedur benutzt werden. (lokale Variablen).
- `[:v4 ... ], [:v5...]` sind optionale Variablen, die wir der Prozedur hinzufügen können (gehen Sie nach unten für mehr Erklärungen).
- `Körper der Prozedur` stellt die Befehle dar, die ausgeführt werden, wenn diese Prozedur gerufen wird.

  Z.B.:

  ```
  lerne quadrat :s
    wiederhole 4 [vw :s re 90]
  Ende
  ```

  Die Prozedur wird `quadrat` genannt und nimmt einen s genannten Parameter. `quadrat 100` wird deswegen ein Quadrat produzieren, das eine Seitenlänge von `100` hat. (Betrachten Sie die Beispiele von Prozeduren am Ende vom Handbuch.)

Seit Version 0. 7 c, ist es möglich, Bemerkungen nach dem durch # eingeleiteten Code einzufügen.

```
lerne quadrat :s
  # diese Prozedur erlaubt es, ein Quadrat zu zeichnen dessen Seite :s ist.
  wiederhole 4 [vw :s re 90] # praktisch, nicht wahr?
Ende
```

Optionale Variablen

Es ist nun möglich, einer Prozedur von XLogo optionale Argumente hinzu zu fügen. Schauen Sie sich das Beispiel unten an:

```
lerne vieleck :n [:l 10]
  wiederhole :n [vw :l re 360/:n]
Ende

# mit diesem Befehl wird ein reguläres Vieleck gezeichnet
# 20 Seiten der Länge 10
vieleck 20
```

Während der Interpretation ist die Variable `:l` durch ihren Standardwert ersetzt worden, ich meine die `10` . Wenn wir diesen Wert ändern wollen, müssen wir die Prozedur Vieleck zwischen Klammern aufrufen, um dem Interpreter zu sagen, dass wir optionale Argumente benutzen werden.

```
# Dieser Befehl wird ein reguläres Polygon mit Länge 20 zeichnen
#−Seiten. Jede Seite ist lang 5.
(vieleck 20 5)

# Dies ist ein Quadrat mit jeder Seite der Länge 100
(Vieleck 4 100)
```

5.6.2 Das Konzept der Variablen

Es gibt zwei Arten von Variablen:

- Globale Variablen: Diese sind immer zugänglich von irgendeinem Ort im Programm.
- Lokale Variablen: Diese sind nur zugänglich in den Prozeduren, wo sie definiert werden. In dieser Logo−Version sind lokale Variable nicht zugänglich in Unterprozeduren. Am Ende der Prozedur werden die lokalen Variablen gelöscht.

5.6.3 Das Primitiv verfolge

Es ist möglich, die Arbeit eines Programms zu verfolgen, um sich die Prozeduren zeigen zu lassen, wenn sie arbeiten. Dieser Modus zeigt, dank des Primitivs `rückgabe`, was die Prozeduren ausgeben. Um diesen Modus zu verwenden, tippen Sie **`verfolge`**.

`stoppeverfolge` wird den `Verfolge`−Modus deaktivieren. Ein kleines Beispiel mit der Fakultät (siehe Seite 52).

```
Verfolge dz fak 4
```

ergibt:

```
fak 4
  fak 3
    fak 2
      fak 1
      fak ergibt 1
    fak ergibt 2
  fak ergibt 6
fak ergibt 24
24
```

Deutsch	Englisch	Argumente	Verwendung
setze	make		

		a b: a Wort, b irgendwas	Wenn die lokale Variable a existiert, weist sie den Wert b zu. Wenn nicht, erzeugt sie eine globale Variable a und weist ihr den Wert b zu. Z.B.: `setze "a 100` weist den Wert 100 der Variablen a zu.
lokal	local	a: Wort	Erzeugt eine Variable namens a. Beachte, diese wird nicht initialisiert. Um einen Wert zuzuweisen, siehe `setze`.
lokalsetze	localmake	a b: a Wort, b irgendwas	Erzeugt eine neue lokale Variable und weist ihr den Wert b zu.
def, definiere	def, define	Wort1 Liste2 Liste3	Definiert eine neue Prozedur namens Wort1, welche die Variablen in Liste2 erfordert. Liste3 enthält die Anweisungen der Prozedur. Z.B. `def "Vieleck [nb länge][wiederhole :nb [vw :länge re 360/:nb]]`

−−−> Dieses Kommando definiert eine Prozedur namens Vieleck mit zwei Variablen :nb und :länge. Diese Prozedur zeichnet ein reguläres Vieleck, wir können die Anzahl der Seiten und ihre Längen wählen. ||

Deutsch	Englisch	Argumente	Verwendung
wert	thing	a: Wort	Ergibt den Wert der Variablen :a. `wert "a` ist ähnlich wie `:a`
vg, vergesse	er, erase	a: Wort	Entfernt die Prozedur namens a.
vgv, vergessevar	kill	a: Wort	Löscht die Variable a.
vga, vergessealles	erall, eraseall	keine	Entfernt alle aktuellen Variablen und aktuellen Prozeduren.
zga, zeigealles	poall, printoutall	keine	Listet alle aktuell definierten Prozeduren.
starte	run	a :Liste	Führt die Liste von Instruktionen aus, die in Liste a stehen.
variablen	lvars, listvariables	keine	Ergibt eine Liste die alle definierten Variablen enthält.

5.6.4 Eigenschaftslisten

Nun können Sie in XLogo Eigenschaftslisten definieren. Jede Liste hat einen bestimmten Namen und enthält einige Schlüssel−Wert−Paare.

Zum Beispiel können wir eine Eigenschaftsliste namens "Auto" betrachten. Sie soll einen Schlüssel "Farbe" mit dem Wert "rot" assoziieren, und den Schlüssel "Typ" mit dem Wert "4x4".

Um diese Listen zu handhaben, können wir folgende Primitive benutzen:

- **`setzeeg`**

  Syntax: setzeeg Listenname Schlüssel Wert

  Fügt zur Eigenschaftsliste namens Listenname eine Eigenschaft hinzu. Auf den Wert kann mit dem Schlüssel zugegriffen werden. Existiert keine Eigenschaftsliste namens

Listenname, wird sie erzeugt.

- **`gebeeg`**

  Syntax: gebeeg Listenname Schlüssel

  Gibt den Wert zurück, der mit dem Schlüssel Schlüssel in der Eigenschaftsliste namens Listenname assoziiert ist. Wenn diese Eigenschaft nicht existiert oder kein gültiger Schlüssel angegeben ist, wird eine leere Liste zurückgegeben.

- **`entferneeg`**

  Syntax: entferneeg Listenname Schlüssel

  Entfernt die zusammengehörigen Schlüsselwert−Paare aus der Eigenschaftsliste Listenname

- **`egliste`**

  Syntax: egliste Listenname

  Zeigt alle Schlüssel−Wert−Paare an, die in der Eigenschaftsliste namens Listenname stehen.

Nun zurück zur Eigenschaftsliste "Auto":

```
# Füllen einer Eigenschaftsliste
setzeeg "Auto "Farbe "rot
setzeeg "Auto "Typ "Polo
setzeeg "Auto "Hersteller "VW

# Zeige einen Wert an
druckezeile gebeeg "Auto "Farbe
```

rot

```
# Zeige alle Elemente an
druckezeile egliste "Auto
Hersteller VW Farbe rot Typ Polo
```

5.7 Dateien handhaben

Deutsch	Englisch	Argumente	Verwendung
listedateien	ls, listfiles	keine	Listet den Inhalt eines Verzeichnisses. (Äquivalent zu dem ls Kommando für Linux−Benutzer und dem dir Kommando für DOS−Benutzer)
ladebild	li, loadimage	a: Liste	Ladet eine Bilddatei. Ihre obere linke Ecke wird an der Position des Igel gesetzt. Die einzigen unterstützten Formate sind .png und .jpg. Der Pfad muss relativ zum

			aktuellen Ordner angegeben werden. Z.B.: `setzeordner "C:\\meine_bilder ladebild "igel.jpg`
sordner, setzeordner	setdir, setdirectory	l: Liste	Setzt das aktuelle Verzeichnis. Der Pfad muss absolut angegeben werden. Das Verzeichnis muss mit einem Wort bezeichnet werden.
wo, wechsleordner	cd, changedirectory	m: Wort	Erlaubt es das aktuelle Verzeichnis zu wählen. Der Pfad ist relativ zum aktuellen Verzeichnis. Sie können die '..' Schreibweise verwenden, um sich auf das Eltern−Verzeichnis zu beziehen.
ord, ordner	dir, directory	keine	Ergibt das das aktuelle Verzeichnis. Das Standard Homeverzeichnis des Benutzers lautet `/home/your_login` für Linux−Benutzer, `C:\WINDOWS` für Benutzer von Windows.
speichere	save	w: Wort l: Liste	Ein gutes Beispiel das zu erklären ist: `speichere "test.lgo [proc1 proc2 proc3]` speichert in der Datei test.lgo im Verzeichnis die Prozeduren proc1, proc2 und proc3. Wenn die Erweiterung `.lgo` fortgelassen wird, wird sie standardmäßig hinzugefügt. Das Wort gibt einen relativen Pfad beginnend vom aktuellen Verzeichnis. Dieses Kommando wird nicht mit einem absoluten Pfad arbeiten.
gespeichert	saved	w: Wort	`gespeichert "test.lgo` speichert in der Datei test.lgo im aktuellen Verzeichnis die aktuell definierten Prozeduren. Wenn die Erweiterung `.lgo` fortgelassen wird, wird sie standardmäßig hinzugefügt. Das angegebene Wort gibt den relativen Pfad beginnend mit dem aktuellen

			Verzeichnis. Dieses Kommando wird nicht mit einem absoluten Pfad arbeiten.
lade	load	w: Wort	Öffnet und liest die Datei `w`. Um zum Beispiel alle definierten Prozeduren zu löschen und die Datei test.lgo" zu laden, würden Sie schreiben: `efns lade "test.lgo`. Das angegebene Wort gibt den relativen Pfad beginnend mit dem aktuellen Verzeichnis. Dieses Kommando wird nicht mit einem absoluten Pfad arbeiten.
öffnefluss	openflow	id Datei	Wenn Sie von einer Datei lesen oder in sie schreiben wollen, müssen Sie sie zuerst öffnen. Das Argument Datei" muss der Name der Datei sein. Sie müssen ein Wort geben, um die Datei im aktuellen Verzeichnis zu bezeichnen. Das id Argument ist die Zahl, die dem Fluss gegegen wird, um die Datei zu identifizieren.
listefluss	listflow	keine	Zeigt die Liste der verschiedenen offenen Flüsse mit ihren Bezeichnern.

Deutsch	Englisch	Argumente	Verwendung
lesezeilenfluss	readlineflow	id	Öffnet einen Fluß dessen Bezeichner der Zahl der Datei entspricht and dann eine Zeile einliest.
lesebuchstabenfluss	readcharflow	id	Öffnet einen Fluß dessen Bezeichner der Zahl entspricht, die als Argument benutzt wird und dann ein Zeichen in diese Datei einliest. Dieses Primitiv sendet eine Zahl zurück, die den Wert eines Zeichens darstellt (ähnlich zu `lesezeichen`).
schreibezeilenfluss	writelineflow	id Liste	Schreibt die Textzeile in der Liste an den Anfang der Datei bezeichnet durch id. Seien Sie vorsichtig: Das Schreiben funktioniert nur, wenn der Fluss durch das Primitiv `closeflow` geschlossen wurde.
hängezeileanfluss	appendlineflow	id Liste	

			Schriebt die Textzeile in der Liste an das Ende der Datei bezeichnet durch id. Seien Sie vorsichtig: Das Schreiben funktioniert nur, wenn der Fluss durch das Primitiv `closeflow` geschlossen wurde.
schließefluss	closeflow	id	Schliesst den Fluss mit dem Bezeichner als Argument.
endefluss?	endflow?	id	Sendet `wahr`, wenn es das Ende der Datei ist, sonst `falsch`.

Hier ist ein Beispiel für die Verwendung von erlaubten Primitiven, um von einer Datei zu lesen und zu schreiben. Ich werde dieses Beispiel im Rahmen von Windows geben. Andere Benutzer sollten das folgende Beispiel anpassen können.

Das Ziel dieser Datei ist es, die Datei `c:\example` zu erzeugen, die die folgenden drei Zeilen enthält:

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ
Abcdefghijklmnopqrstuvwxyz
0123456789

# Sie können einen Fluss für die gewünschte Datei öffnen.
# Diesem Fluss wird die Zahl 2 gegeben.

# funk nicht? wechsleordner "..
# funk nicht: wechsleordner "c:\\Program\ files\\
wechsleordner "c:\\Program\ files
; wechsleordner "beispiel-57
öffnefluss 2 "beispiel.txt

# Sie tippen die gewünschten Zeilen

schreibezeilenfluss 2 [ABCDEFGHIJKLMNOPQRSTUVWXYZ]
schreibezeilenfluss 2 [abcdefghijklmnopqrstuvwxyz]
schreibezeilenfluss 2 [0123456789]

# Sie schließen den Fluss, um das Schreiben zu beenden
schließefluss 2
```

Jetzt können Sie sehen, dass die Schreibprozedur richtig lief:

```
# Sie öffnen einen Fluss für die Datei, die Sie lesen wollen.
# Diesem Fluss wird die Zahl ``0`` gegeben.

öffnefluss 0 "beispiel.txt

# Sie lesen jede Zeile eine nach der anderen aus der Datei

dz lesezeilenfluss 0
dz lesezeilenfluss 0
dz lesezeilenfluss 0

# Sie schließen den Fluss:

schließefluss 0
```

Wenn Sie die Zeile 'Großartig!' hinzufügen wollen:

```
wechsleordner "c:\\
öffnefluss 1 "Beispiel]
```

```
hängezeileanfluss 1 [Großartig!]
schließefluss 1
```

5.8 Die fortgeschrittene Fülle−Funktion

Zwei Primitive erlauben, eine Figur zu färben. Die Primitive **`fülle`** und **`fülleform`**.

Diese Primitive erlauben eine Form anzumalen. Diese Primitive können mit den verfügbaren 'Fülle'−Eigenschaften von vielen Bildretuscheprogramme verglichen werden. Dieses Merkmal kann sich auf die Ränder des Designbereichs ausdehnen. Es gibt zwei Regeln, die in dieser Reihenfolge eingehalten werden müssen, um dieses Primitiv richtig zu benutzen:

- Der Schreibstift muss abgesenkt werden (`stiftab`).
- Der Igel darf sich auf keinem Bildelement der Farbe befinden, mit der die Form gefüllt werden soll. (Wenn Sie Figuren `rot` anmalen wollen, darf er nicht auf `rot` sitzen...)

  Nehmen wir ein Beispiel, um den Unterschied zwischen `fülle` und `fülleform` zu sehen:

Das Bildelement unter der Schildkröte ist in diesem Augenblick `weiß`. Das Primitiv `fülle` wird alle benachbarten Bereiche anmalen

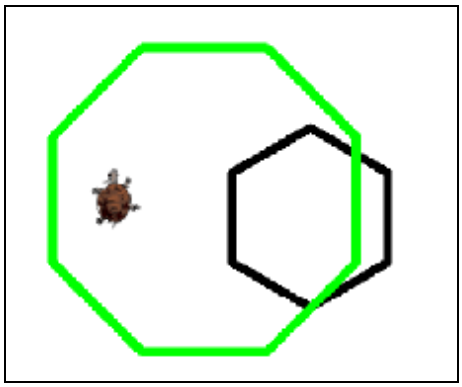


Abbildung 5.1: Am Anfang weiße Bildelemente mit der aktuellen Stiftfarbe an. Wenn Sie zum Beispiel tippen: `setzestiftfarbe 1 fülle` . Gehen wir jetzt zurück zu dem ersten Fall.

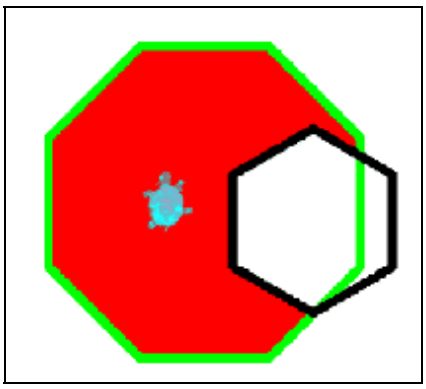
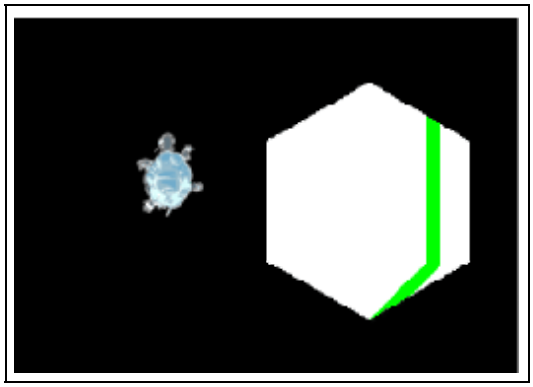


Abbildung 5.2: Mit dem Primitiv `Fülle`

Wenn die Stiftfarbe des Igel `schwarz` ist, wird das Primitiv `füllform` alle Bildelemente färben bis es die aktuelle Farbe (hier `schwarz`) findet.



Dies ist ein gutes Beispiel für den Gebrauch dieses Primitivs:

```
lerne halbkreis :c
  # ziehe einen Halbkreis vom Durchmesser :c
  wiederhole 180 [vw :c*tan 0.5 re 1]
  vw :c*tan 0.5
  re 90 vw :c
  re 90 # bringt Igel wieder in die Anfangsrichtung
Ende
```

□

Abbildung 5.3: Mit dem Primitiv `füllform` , wenn Sie tippen: `setzestiftfarbe 0 füllform`

`tan` gibt es schon als Primitiv:

```
lerne tangens :winkel
  # gibt den Tangens des Winkels wieder
  rückgabe (sin :winkel)/cos :winkel
Ende

lerne regenbogen :c
  wenn :c<100 [stoppe]
  halbkreis :c re 180 vw 20 li 90
  regenbogen :c-40
Ende

lerne dep
  sh re 90 vw 20 li 90 sa
Ende

lerne rbogen
  vi regenbogen 400 rd li 90 vw 20 rw 120 ns sh re 90 vw 20 sa
  setzestiftfarbe 0 fülle dep
  setzestiftfarbe 1 fülle dep
  setzestiftfarbe 2 fülle dep
  setzestiftfarbe 3 fülle dep
  setzestiftfarbe 4 fülle dep
  setzestiftfarbe 5 fülle dep
  setzestiftfarbe 6 fülle dep
Ende
```

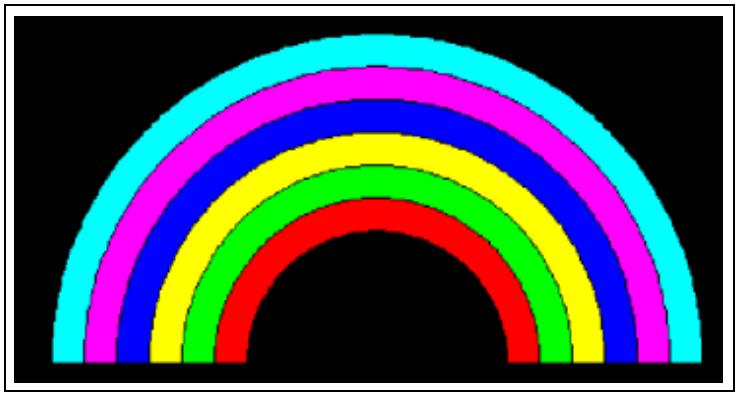


Abbildung 5.4: Bogen herein−LOGO

5.9 Unterbrechungs−Befehle

Logo hat drei Abbruchbefehle: `stoppe`, `stoppealles` und `rg, Rückgabe`.

`Stoppe` kann zwei Ergebnisse haben. Wenn es in eine `Wiederhole−` oder `Während`−Schleife eingeschlossen ist, springt das Programm heraus aus der Schleife. Wenn es in einer Prozedur vorkommt, bricht das Programm sofort aus der Prozedur aus.

`Stoppealles` bricht das Programm aus der ganzen Prozedur heraus sofort ab und stoppt.

`Rückgabe, rg` bricht aus einer Prozedur aus mit einem zurückzugebenden Wert. Sehen Sie die zahlreichen Fälle zum Gebrauch dieser zwei Primitive in den Beispielen am Ende von diesem Handbuch.

5.10 Der Viele−Igel−Modus

Es ist möglich mehrere aktive Igel auf dem Schirm zu haben. Standardmäßig ist beim Start von XLogo nur ein Igel vorhanden. Seine Nummer ist die 0. Wenn Sie einen neuen Igel "erschaffen" wollen, können Sie das Primitiv `si setzeigel` gefolgt von der Anzahl der Igel verwenden. Um Obstruktion zu verhindern, wird der Igel am Ursprung geschaffen und ist unsichtbar (Sie müssen `zi zeigeeigel` benutzen, um ihn zu zeigen). Dann ist der neue Igel der aktive Igel, er gehorcht allen klassischen Primitiven, während Sie den aktiven Igel mit `si setzeigel` nicht ändern. Die größtmögliche Anzahl von verfügbaren Igel kann gesetzt werden im Menü Hilfsmittel−Einstellungen − Tab Optionen. Das Primimtiv dazu heißt `Setzeigelmax`.

Hier sind die Primitive für den Viele−Igel−Modus:

Deutsch	Englisch	Argumente	Verwendung
sigel, setzeigel	sturtle, setturtle	a: Zahl	Der Igel mit Nummer a ist nun der aktive Igel. Beim Start von Xlogo ist der aktive Igel der mit der Nummer 0.
igel	turtle	keine	Ergibt die Nummer des aktiven Igel.
igelliste	turtles	keine	Ergibt eine Liste, die alle Nummern der Igel enthält, die tatsächlich auf dem Schirm sind.
	killturtle	a: Zahl	Löscht den Igel mit der Nummer a

ligel, löscheigel			
setzeigelmax	setTM, setturtlesmax	Ganzzahl	Setzt die maximale Anzahl Igel für den Viele−Igel−Modus.
igelmax	tm, turtlesmax	none	Ergibt die maximale Anzahl Igel für den Viele−Igel−Modus.

5.11 Musik spielen

Deutsch	Englisch	Argumente	Verwendung
fol, folge	seq, sequence	a: Liste	Erzeugt eine Sequenz der Liste im Speicher. Lesen Sie nach dieser Tabelle, um zu lernen wie eine Sequenz zu erzeugen ist.
spiele	play	keine	Spielt die Sequenz im Speicher.
instr, instrument	instr, instrument	keine	Ergibt die Zahl, die dem gewählten Instrument entspricht.
sinstr, setzeinstrument	sinstr, setinstrument	a: Zahl	Das gewählte Instrument ist nun das Instrument mit der Nummer a. Sie können die Liste aller verfügbaren Instrumente sehen im Menü Hilfsmittel−Einstellungen−Tab Sound.
indfol, indexfolge	indseq, indexsequence	keine	Ergibt die Position des Lesezeigers in der aktuellen Sequenz.
sindfol, setzeindexfolge	sindseq, setindexsequence	a: Zahl	Setzt den Lesezeiger auf den Index a in der aktuellen Sequenz im Speicher.
lfol, löschefolge	delseq, deletesequence	keine	Löscht die aktuelle Sequenz im Speicher.

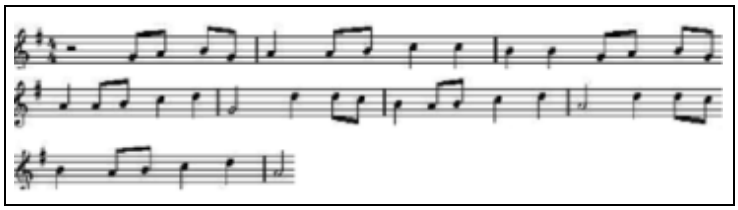
Wenn Sie Musik spielen wollen, müssen Sie die Noten im Speicher in eine Sequenz genannte Liste stellen. Um diese Folge zu erzeugen, können Sie das Primitiv `fol` oder `folge` benutzen. Dies sind die Regeln, die einzuhalten sind, um eine gültige Sequenz zu schaffen:

`do re mi fa sol la si` : die üblichen Noten der ersten Oktave.

Um ein spitzes re zu erzeugen, schreiben wir `re +` Um ein flaches re zu erzeugen, schreiben wir `re −`

Wenn Sie eine Oktave hinauf oder hinunter gehen wollen, benutzen Sie das Symbol ":" gefolgt von + oder −. Z. B. werden nach `:++` alle Noten der Sequenz zwei Oktaven höher gespielt (zwei ++). Standardmäßig werden Noten für eine Dauer von `1` gespielt. Wenn Sie sie verlängern oder vermindern wollen, schreiben Sie die Zahl, die der Dauer der Noten entspricht. Z. B. `folge [sol 0.5 la si]` wird `sol` mit einer Dauer `1` und `la si` mit einer Dauer `0.5` spielen (zweimal schneller).

Wenn Sie dieses Beispiel spielen wollen:



```
lerne tabac
  # erzeugt die Sequenz von Noten
  fol [0.5 sol la si sol 1 la 0.5 la si 1 :+ do do :− si si 0.5 sol la si sol
     1 la 0.5 la si 1 :+ do re 2 :− sol ]
  fol [:+ 1 re 0.5 re do 1 :− si 0.5 la si 1 :+ do re 2 :− la ]
  fol [:+ 1 re 0.5 re do 1 :− si 0.5 la si 1 :+ do re 2 :− la ]
  fol [0.5 sol la si sol 1 la 0.5 la si 1 :+ do do :− si si 0.5 sol la si sol
     1 la 0.5 la si 1 :+ do re 2 :− sol ]
Ende
```

tabac spiele

Um Musik zu hören, beginnen Sie mit dem Kommando: `tabac play` Jetzt können wir eine interessante Anwendung des Primitivs `sindseq` sehen. Schreiben Sie diese Kommandos:

```
delseq    # Lösche die Sequenz aus dem Speicher
tabac     # Lege die Sequenz in den Speicher
sindfol 2 # Setze die Schreibmarke auf das zweite "la".
tabac     # Setze die gleiche Sequenz in den Speicher, aber übersetzt von 2.
play      # großartig!
```

Sie können das Instrument mit dem Primitiv `sinstr, setzeinstrument` oder mit dem Menü Hilfsmittel−Einstellungen auf dem Tab Sound wählen. Sie werden dort die Liste von allen verfügbaren Instrumenten mit ihrer dazugehörigen Zahl finden.

5.12 Schleifen

In XLogo gibt es drei Primitive, die die Konstruktion von Schleifen erlauben: `Wiederhole`, `Für` und `Solange` .

5.12.1 Eine Schleife mit Wiederhole

Dies ist die Syntax für **`Wiederhole`**:

`Wiederhole n List_von_Kommandos`

n ist eine ganze Zahl und `Liste_von_Kommandos` enthält eine Liste der auszuführenden Befehle. Der Logo−Interpreter wird die Befehle in der Liste n mal ausführen: So braucht das Kommando nur einmal geschrieben zu werden!

Z.B.:

```
wiederhole 4 [vw 100 li 90]  # ein Quadrat mit der Seitenlänge 100
wiederhole 6 [vw 100 li 60]  # ein Sechseck mit der Seitenlänge 100
wiederhole 360 [vw 2 li 1]   #−A−Uh... 360−Eck mit Seitenlänge 2
                                  # kurz gesagt: fast ein Kreis!
```

In eine Wiederhole−Schleife eingeschlossen gibt die interne Variable **Whzahl** die Anzahl der laufenden Iteration zurück. (Dabei ist die erste Iteration die Zahl 1).

```
wiederhole 3 [dz whzahl]
1
2
3
```

5.12.2 Eine Schleife mit Für

damit weisen Sie einer Variablen einige mit einer Schrittweite aufeinander folgende Werte aus einem festen Bereich zu. Hier ist die Syntax von **Für**:

```
Für Liste1 Liste2
```

Liste1 enthält drei Argumente: der Variablenname, der Anfangswert und der Endwert. Ein viertes Argument ist das Inkrement(die optionale Schrittweite zwischen zwei aufeinander folgenden Werten). Standardwert ist `1` . Hier sind ein paar Beispiele:

```
für [i 1 4][dz :i*2]
```

```
2
4
6
8
```

Jetzt wird i von 7 nach 2 verringert, jedesmal jeweils um 1.5 # Schauen Sie sich das negative Inkrement an # dessen Quadrat angezeigt wird.

```
für [i 7 2 −1.5][dz (liste :i potenz :i 2)]
```

```
7 49
5.5 30.25
4 16
2.5 6.25
```

5.12.3 Eine Schleife mit Solange

Dies ist die Syntax für **Solange**:

```
Während Liste_istauszuwerten Liste_von_Kommandos
```

`Liste_istauszuwerten` enthält eine Liste von Befehlen, die zu einem booleschen Wert ausgewertet werden kann. `Liste_von_Kommandos` enthält eine Liste auszuführender Befehle. Der Logo−Interpreter wird die `Liste_von_Kommandos` auswerten, **solange** die `Liste_istauszuwerten` Wahr zurückgibt.

Z.B.:

```
Solange [wahr] [re 1] # dreht den Igel um sich
```

Nun ein Beispiel, das uns erlaubt das Alphabet rückwärts zu buchstabieren

```
setze "liste "abcdefghijklmnopqrstuvwxyz
solange [nicht leer? :liste] [dz letztes :liste  setze "liste ohneletztes :liste ]
```

5.13 Eingaben vom Benutzer empfangen

5.13.1 Interagieren Sie mit der Tastatur

Gegenwärtig kann Text vom Benutzer während der Programmausführung hauptsächlich über 3 Primitive akzeptiert werden: `Taste?`, `lesezeichen` und `lese` .

`Taste?`: Wird als `wahr` oder `falsch` gelesen, abhängig davon, ob seit dem Beginn der Programmausführung eine Taste gedrückt worden ist oder nicht.

`lesezeichen`:

- Wenn `Taste?` den Wert `Falsch` ergibt, pausiert das Programm bis der Benutzer eine Taste drückt.
- Wenn Taste? `wahr` ist, gibt es die Taste aus, die zuletzt gedrückt war. Dies sind die Werte für besondere Tasten:

A −−> 65	B −−> 66	C −−> 67	etc ...	Z −−> 90
−−> −37 or −226 (NumPad)	−−> −38 or −224	−−> −39 or −227	−−> −40 or −225	.
Echap −−> 27	F1 −−> −112	F2 −−> −113	....	F12 −−> −123
Shift −−> −16	Espace −−> 32	Ctrl −−> −17	Enter −−> 10	.

Tabelle 5.11: Werte für besondere Tasten

Wenn Sie unsicher über den von einer Taste zurückgegebenen Wert sind, können Sie tippen:

```
dz lesezeichen
```

Der Interpreter wird dann warten, bis Sie auf eine Taste tippen, bevor er den Wert zurückgibt.

```
lese liste_titel Wort:
```

Gibt einen Dialogkasten, dessen Titel `liste_titel` ist. Der Benutzer kann dann in einem Textfeld eine Antwort eingeben und die Antwort wird dann in Gestalt von einem Wort oder einer Liste, wenn der Benutzer mehrere Wörter geschrieben hat, in der Variablen `:word` gespeichert werden. Ausgewertet wird, wenn der OK−Knopf gedrückt wird.

5.13.2 Einige Beispiele zum Gebrauch

```
  lerne alter
    lese [Was ist Ihr Alter?] "Alter
    # setze "alter erstes :alter
    wenn :alter<18 [dz [Sie sind ein Kind.]]
    wenn oder :alter=18 :alter>18 [dz [Sie sind ein Erwachsener.]]
# so nicht:
# wenn :alter>=18 [dz [Sie sind ein Erwachsener.]]
    wenn :alter>99 [dz [Respekt, Respekt!!]]
    dz "
Ende

  lerne rallye
    wenn taste? [
```

```
    setze "auto lesezeichen
    wenn :auto=-37 [li 90]
    wenn :auto=-39 [re 90]
    wenn :auto=-38 [vw 10]
    wenn :auto=-40 [rw 10]
    wenn :auto=27 [stoppe]
  ]
  rallye
Ende

# Sie können den Igel mit den Pfeiltasten steuern und mit Esc stoppen.
```

5.13.3 Interagieren Sie mit der Maus

Gegenwärtig können Mausereignisse vom Benutzer während der Programmausführung über drei Primitive akzeptiert werden:

`lesemaus, mauspos` und `Maus?` .

`lesemaus`: Das Programm pausiert bis der Benutzer die Maus drückt. Dann gibt es eine Zahl zurück, die das Ereignis repräsentiert.

- Dies sind die unterschiedlichen Werte:
  - **0** Die Maus wurde bewegt
  - **1** Der Knopf 1 ist gedrückt gewesen
  - **2** Der Knopf 2 ist gedrückt gewesen

    Der Knopf 1 ist der linke Knopf, der Knopf 2 ist der nächste rechts...

`mauspos, mausposition`: Gibt eine Liste zurück, die die Position der Maus enthält.

`Maus?`: Gibt ein `Wahr` zurück, wenn wir seit Programmbeginn die Maus anfassen, sonst ein `Falsch`.

5.13.4 Einige Beispiele zum Gebrauch

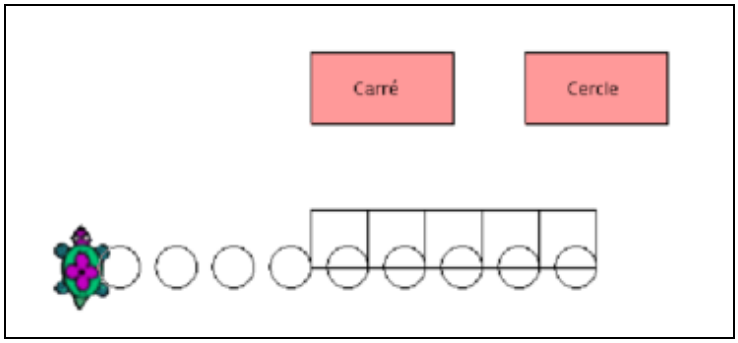
In der ersten Prozedur folgt der Igel der Maus, wenn er sich auf dem Bildschirm bewegt.

```
lerne beispiel
  # wenn die Maus bewegt wird, geht der Igel zur nächsten Position
  wenn lesemaus=0 [aufpos mauspos]
  beispiel
Ende
```

In dieser zweiten Prozedur ist es wie vorher, nun aber müssen Sie mit dem linken Knopf der Maus klicken, wenn Sie wollen, dass sich der Igel bewegt.

```
lerne beispiel2
  wenn lesemaus=1 [setzepos mauspos]
  beispiel2
Ende
```

In diesem dritten Beispiel schaffen wir zwei rosafarbene Schaltflächen. Wenn wir links klicken auf die linke Schaltfläche, zeichnen wir ein Quadrat mit einer Seite von `40`. Klicken wir links auf die rechte Schaltfläche, zeichnen wir einen kleinen Kreis. Zuletzt klicken wir rechts auf die rechte Schaltfläche und das Programm stoppt.




```
lerne knopf
  # erzeugt eine rosafarbene, rechteckige Schaltfläche (Höhe 50, Breite 100)
  wiederhole 2 [vw 50 re 90 vw 100 re 90]
  re 45 stifthoch vw 10 stiftab setzestiftfarbe [255 153 153]
  fülle rw 10 li 45 stiftab setzestiftfarbe 0
Ende

lerne lance
  lb knopf stifthoch setzepos [150 0] stiftab Knopf
  sh setzepos [30 20] sa igeltext "Quadrat
  sh setzepos [180 20] sa igeltext "Kreis
  sh setzepos [0 −100] sa
  maus
Ende

lerne maus
  # wir legen den Wert von lesemaus in die Variable ev
  setze "ev lesemaus
  # wir legen die erste Koordinate der Maus in die Variable x
  setze "x element 1 mauspos
  # wir legen die zweite Koordinate der Maus in die Variable y
  setze "y element 2 mauspos
  # Wenn wir den linken Knopf Schaltfläche klicken
  wenn :ev=1 & :x>0 & :x<100 & :y>0 & :y<50 [Quadrat]
  # Wenn wir den rechten Knopf | Schaltfläche klicken
  wenn :x>150 & :x<250 & :y>0 & :y<50 [
     wenn :ev=1 [meinkreis]
     wenn :ev=3 [stoppe]
  ]
maus
Ende

lerne Kreis2
  wiederhole 90 [vw 1 li 4] li 90 sh
  vw 40 re 90 sa
Ende

lerne Quadrat
  wiederhole 4 [vw 40 re 90]
Ende
```

lance startet

5.13.5 Graphische Komponenten gebrauchen

Mit XLogo können Sie mehrere graphische Komponenten auf der Zeichenfläche hinzufügen (Schaltfläche, Kombo−Menü). Alle Primitive, die dem Benutzer erlauben jene Komponenten zu manipulieren, fangen mit dem Präfix Gui an (für Graphische Anwenderschnittstelle).

Schaffen Sie sich eine Komponente

Zuerst müssen Sie jene graphischen Objekte erstellen, dann können Sie einige ihrer Eigenschaften ändern und zuletzt auf der Zeichenfläche zeigen.

- Um eine Schaltfläche zu schaffen, müssen wir das Primitiv **`guitaste`** benutzen. Hier ist die Syntax:

  # Dieser Befehl schafft eine Schaltfläche mit dem Namen b # auf die ""Klicke"" geschrieben wird:

  ```
  guitaste "B "Klicke
  ```

- Um ein Kombo−Menü zu schaffen, müssen wir das Primitiv **`guimenü`** benutzen. Hier ist die Syntax:

  # Dieser Befehl erschafft ein Kombomenü mit dem Namen m # und die Liste enthält 3 Elemente : Element1, Element2 und Element3

  ```
  guimenü "m [Element1 Element2 Element3]
  ```

Modifizieren Sie einige Eigenschaften von graphischen Komponenten

`guiposition`: Positioniert das graphische Element an einem bestimmten Ort mit seiner Koordinate. Zum Beispiel, wenn Sie die Schaltfläche an den Punkt mit den Koordinaten (20; 100) stellen wollen, werden Sie schreiben:

```
guiposition "b [20 100]
```

Wenn Sie keinen Ort für die Komponente spezifizieren, wird er standardmäßig auf die obere linke Ecke der Zeichenfläche gesetzt werden.

`guientferne`: Beseitigt eine graphische Komponente. Zum Beispiel um die Schaltfläche zu entfernen:

```
guientferne "b
```

`guiaktion`: Definiert eine Aktion für die Komponente, wenn der Benutzer damit interagiert.

Bewegt den Igel um 100 vor, wenn wir auf die Schaltfläche "b klicken

```
guiaktion "b [vw 100]
```

Für das Kombomenü hat jedes Element seine eigene Aktion

```
guiaktion "m [[dz Element1] [dz "Element2] [dz "Element3]]
```

`guizeichne`: Zeigt die graphische Komponente auf der Zeichenfläche. Zum Beispiel, um die Schaltfläche zu zeigen:

```
guizeichne "b
```

5.14 Zeit und Datum

XLogo hat mehrere Primitive für Datum, Zeit und Countdown.

Deutsch	Englisch	Argumente	Verwendung

warte	wait	n: Ganzzahl	Hält das Programm an, und daher auch den Igel, für die Dauer von n/60 Sekunden.
stopuhr	chrono, chronometre	n: Ganzzahl	Startet einen Countdown von n Sekunden. Wir wissen, wann dieser Countdown beendet ist, wenn wir das Primitiv endecountdown? benutzen.
endecountdown?	endcountdown?	keine	Ergibt "wahr, wenn es keinen aktiven Countdown gibt, sonst "false wenn der Countdown aktiv ist.
datum	date	keine	Ergibt eine Liste die drei Ganzzahlen enthält, die das Datum darstellen. Die erste Ganzzahl zeigt den Tag, die zweite den Monat und die letzte das Jahr. —> [tag monat jahr]
zeit	time	aucun	Ergibt eine Liste die drei Ganzzahlen enthält, die die Zeit darstellen. Die erste Ganzzahl zeigt die Stunde, die zweite die Minute und die letzte die Sekunden. —> [stunde minute sekunde]
vergangenezeit	pasttime	keine	Ergibt die vergangene Zeit in Sekunden seit XLogo gestartet wurde.

Der Unterschied zwischen Warte und Countdown ist, dass Countdown das Programm nicht anhält.

Hier ist ein Beispiel:

```
lerne uhr
  # zeigt die Zeit im nummerischen Format
  # wir frischen die Zeit alle fünf Sekunden auf
  wenn endecountdown? [
    lb
    sfont 75 vi
    setze "Heu zeit
    setze "h erstes :heu
    setze "m element 2 :heu
    # Wir zeigen zwei Zahlen für Sekunden und Minuten. (wir müssen eine 0 hinzufügen)
    wenn :m-10<0 [setze "m Wort 0 :m]
    setze "s zuletzt :heu
    # Wir zeigen zwei Zahlen für Sekunden und Minuten. (wir müssen eine 0 hinzufügen)
    wenn :s-10<0 [setze "s Wort 0 zu :s]
    igeltext wort wort wort wort :h ": :m ": :s
    countdown 5
  ]
  uhr
Ende
```

5.15 XLogo im Netzwerk benutzen

5.15.1 Das Netz − Wie geht das?

Zuerst müssen wir in die Grundlagen für Netzwerkkommunikation einführen, bevor wir die XLogo−Primitive benutzen können.

Zwei Computer (oder mehr) können über ein Netz kommunizieren, wenn in beiden Ethernet−Karten eingebaut sind.

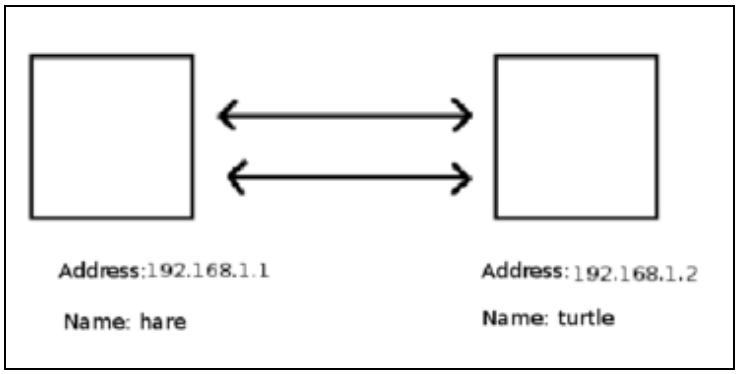


Abbildung 5.5: Ein einfaches Netz

Jeder Computer wird durch eine persönliche IP−Adresse identifiziert. Diese IP−Adresse besteht aus vier ganzen Zahlen, jede zwischen 0 und 255 und getrennt durch einen Punkt. Zum Beispiel wird der erste Computer in der Abbildung adressiert durch die IP **192.168.1.1**

Weil es nicht leicht ist, sich an diese Zahlen zu erinnern, ist es auch möglich, jeden Computer über einen gewohnteren Namen zu identifizieren. Wie wir in der Abbildung sehen, können wir mit dem rechten Computer mit Hilfe seiner IP−Adresse **192.168.1.2** kommunizieren oder mit seinem Namen **Turtle** .

Hier möchte ich eine Sache anmerken: Der lokale Computer, an dem Sie arbeiten, besitzt die Adresse **127.0.0.1** . Sein allgemeiner Name ist **localhost** . Wir werden dies später in der Praxis sehen.

5.15.2 Primitive für das Netz

XLogo hat 4 Primitive, die ihm ermöglichen, über ein Netz zu kommunizieren: `höretcp`, `ausführetcp`, `chattcp` und `sende`. In allen künftigen Beispielen werden wir den einfachen Fall von zwei Computern in der bisherigen Abbildung nehmen.

`höretcp`: Dieses Primitiv ist die Grundlage für alle Netzkommunikation. Es benötigt kein Argument. Wenn Sie dieses Primitiv auf einem Computer ausführen, wird der Computer nach von anderen Computern auf das Netz geschickte Anweisungen achten.

`ausführetcp`: Dieses Primitiv erlaubt die Ausführung von Anweisungen von einem Computer auf dem Netz.

Syntax: `ausführetcp Wort Liste`

`Wort` ist die angerufene IP−Adresse oder der Computername, `Liste` enthält auszuführende Anweisungen.

Beispiel: Ich bin auf Hase, ich will ein Quadrat mit einer Seite von 100 auf dem anderen Computer zeichnen.

Dazu muss ich auf dem Computer **Turtle** mit dem Befehl `höretcp` beginnen. Dann, auf **Hase** schreibe ich :

```
ausführetcp "192.168.1.2 [Wiederhole 4 [vw 100 re 90]]
```

oder

```
ausführetcp "Turtle [Wiederhole 4 [vw 100 re 90]]
```

chattcp: Erlaubt den Plausch zwischen zwei Computern auf einem Netz. Auf jedem Computer zeigt es ein Chatfenster.

Syntax: `Chattcp Wort Liste`

`Wort` ist die angerufene IP−Adresse oder der Computername, `Liste` enthält den zu zeigenden Satz.

Beispiel: Hase will mit Turtle reden.

Zuerst führt **Turtle** `höretcp` aus und wartet damit auf Anweisungen von Netzcomputern. Dann schreibt **Hase**: `chattcp "192.168.1.2 [hallo Turtle]` . Chat−Fenster werden auf beiden Computern aufgehen, das erlaubt ihnen miteinander zu reden.

sendetcp: Schicken Sie Daten zu einem Computer auf dem Netz und geben Sie seine Antwort zurück.

Syntax: `Sendetcp Wort Liste`

`Wort` ist die angerufene IP−Adresse oder der Computername, `Liste` enthält die zu schickenden Daten. Wenn Xlogo auf dem anderen Computer begonnen wird, wird er mit "In Ordnung" antworten.

Mit diesem Primitiv ist es möglich mit einem Roboter über seinen Netzzugang zu kommunizieren. Dann könnte die Antwort des Roboters unterschiedlich sein.

Beispiel: Turtle will Hase den Satz "3.14159 ist wirklich Pi" schicken.

Zuerst führt **Hase** `höretcp` aus, dass er auf den anderen Computer wartet. Dann schreibt **Turtle**:

```
druckezeile sendetcp "Hase [3.14159 ist wirklich Pi]
```

Eine kleine Andeutung: Starten Sie zwei Instanzen von XLogo auf dem gleichen Computer.

- Führen Sie im ersten Fenster `höretcp` aus.
- Schreiben Sie in das zweite `ausführetcp "127.0.0.1 [vw 100 re 90]`

  Sie können den Igel im anderen Fenster bewegen! (heh, heh, das ist möglich, weil 127.0.0.1 Ihre lokale Adresse bezeichnet, denn es ist Ihr eigener Computer...)

XLogo Handbuch (siehe Quelle)

Übersetzt von Michael Malien	Erzeugt durch Txt2Tags
Geändert am: 07/28/08 11:35:30	am 2008−07−28 um 12:15:08

XLogo Handbuch

10 XLogo aus dem Web ausführen
erzeugt am 29.03.2008 20:42

10 XLogo aus dem Web ausführen

10.1 Das Problem

Sie sind Webmaster einer Website auf der Sie über XLogo sprechen und wollen einige Ihrer Programme zur Verfügung stellen, die Sie mit XLogo erschaffen haben. Sie können die Logo−Datei im `.lgo`−Format herausgeben, besser aber wäre es wenn der Benutzer XLogo online starten kann, dass er direkt Ihr Programm testen kann.

Daher werden wir die Technologie namens "**Java Webstart**" verwenden, um XLogo von einer Website zu starten. Tatsächlich brauchen wir dazu auf unserer Site nur einen Link zu einer Datei mit der Erweiterung `.jnlp` anzubringen, die dann XLogo ausführen wird.

10.2 Eine jnlp Datei erzeugen

Hier sehen Sie ein Beispiel für eine solche Datei. Tatsächlich ist das folgende Beispiel das auf der französischen Site im Abschnitt "exemples`` verwendet wird. Diese Datei erlaubt das Programm zu laden, das den Würfel aus dem Abschnitt über 3D zeichnet. Erklärungen zu dem Programm werden nach dem Code gegeben.

```
<?xml version="1.0" encoding="utf−8"?>
<jnlp spec="1.5+" codebase="http://downloads.tuxfamily.
           org/xlogo/common/webstart">
<information>
  <title>XLogo</title>
  <vendor>xlogo.tuxfamily.org</vendor>
  <homepage href="http://xlogo.tuxfamily.org"/>
  <description>Logo Programming Language</description>
  <offline−allowed/>
</information>

<security>
  <all−permissions/>
</security>

<resources>
  <j2se version="1.4+"/>
  <jar href="xlogo.jar"/>
</resources>

<application−desc main−class="Lanceur">
  <argument>−lang</argument>
  <argument>fr</argument>
  <argument>−a</argument>
  <argument>http://xlogo.tuxfamily.org/fr/html/examples−fr/3d/de.lgo</argument>
</application−desc>
</jnlp>
```

Diese Datei ist im XML−Format geschrieben. Der wichtigste Teil sind diese vier Zeilen:

```
  <argument>−lang</argument>
  <argument>fr</argument>
  <argument>−a</argument>
  <argument>http://xlogo.tuxfamily.org/fr/html/examples−fr/3d/de.lgo</argument>
```

wird z.B. zu:

```
  <argument>http://logolei.de/xlogo/common/webstart/examples−de/3d/de.lgo</argument>
```

In diesen Zeilen stehen die Parameter für XLogo beim Startup.

- Zeile 1 und Zeile 2 erzwingen die Sprache Französisch.
- Die letzte Zeile gibt die zu ladende Datei an.
- Zeile 3 zeigt an, dass das Hauptkommando der Datei beim Startup von XLogo ausgeführt wird.

Ein letzter Hinweis:

> Weil der Server von Tuxfamily nicht alle Verbindungen akzeptieren kann, ist es besser die Datei `xlogo.jar` auf Ihrer Site zu speichern. Sie verlinken diese Datei mit der `.jnlp`−Datei, in dem Sie die Adresse in Zeile 2 nach dem Wort `codebase` ändern.

In **xlogo−de.jnlp**:

```
<jnlp spec="1.5+" codebase="http://logolei.de/xlogo/common/webstart" href="xlogo−de.jnlp
```

In **test−de.jnlp**:

<jnlp spec="1.5+" codebase="http://logolei.de/xlogo/common/webstart" href="test−de.jnlp">

Dieser Link wird einfach in HTML eingebunden, ähnlich wie auf der Download−Seite:

<a
href="http://logolei.de/xlogo/common/webstart/xlogo−de.jnlp">http://logolei.de/xlogo/common/webstar

---

XLogo Handbuch (siehe Quelle)

Übersetzt von Michael Malien	Erzeugt durch Txt2Tags
Geändert am: 03/29/08 20:42:38	am 2008−07−28 um 12:15:52

XLogo Handbuch

12 Über dieses Dokument
geändert am 28.07.2008 12:32

12 Über dieses Dokument

XLOGO Benutzerhandbuch

Dieses Dokument entstand durch Übersetzung des englischen Manual und wurde erzeugt unter Verwendung

- der Windows−Version 2005−06−17 (2.3)
- von Txt2Tags

XLogo Handbuch (siehe Quelle)

Übersetzt von Michael Malien	Erzeugt durch Txt2Tags
Geändert am: 07/28/08 12:32:10	am 2008−07−28 um 12:32:29