# Programmieren mit LOGO

*von Thomas Helmle und Markus Wurster, Juni 2008*

Die Programmiersprache LOGO entstand 1968 und wurde bewusst so konzipiert, dass sie leicht zu erlernen ist. Deshalb konnte sich diese Sprache bis heute halten und findet in didaktischen Zusammenhängen immer noch Anwendungen. Sie ist besonders dafür geeignet, dass Kinder und Jugendliche damit ein **Grundverständnis für die Arbeitsweise des Computers** entwickeln können. Autor ist Seymour Papert, Mathematiker und Psychologe in den USA. Er gilt als bedeutendster Schüler Jean Piagets.

Die **Grundaussage** bei der Verwendung von LOGO könnte man Schülern gegenüber so formulieren: Computer sind keine klugen Zaubermaschinen. Eigentlich sind sie dumm. Alles, was sie können, können sie nur, weil sie von Menschen gut programmiert wurden. Beim Programmieren muss man dem Computer *ganz genau* sagen, was er machen muss...

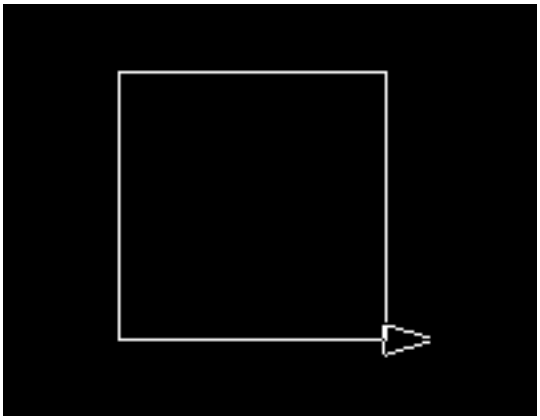
Die Genauigkeit, mit der man programmieren muss, zwingt die Kinder zu einer exakten Arbeitsweise, erfordert hohe Konzentration, fordert logische Planung. Die Arbeit mit LOGO ist also pädagogisch relevant. Sie hat viel mit **mathematischem Denken** zu tun und greift das **geometrische Wissen** der Kinder von einem neuen bzw. anderen Ansatzpunkt her auf. Dazu sind Kinder nach unserer Erfahrung ab etwa 8/9 Jahre in der Lage.

Außerdem erhält die geometrische Arbeit eine starke **ästhetische Komponente**. Vor allem Figuren, die mit Hilfe des Befehls *repeat* programmiert wurden, rufen beim Betrachter ein Staunen hervor. Kinder holen gern ihre FreundInnen zum Bildschirm und sagen stolz: „Schau' mal, wie schön!“

LOGO kann durch verschiedene Interpreter-Programme dargestellt werden. Bekannt wurde die Turtle-Grafik, eine virtuelle Schildkröte, die entsprechend der Programmierung eine farbige Linie hinter sich herzieht.


Wir empfehlen das **UCB-Logo**. Es ist eine englische Version, die an der University of California at Berkeley entstand. Wir haben die Erfahrung gemacht, dass Kinder ganz selbstverständlich mit den originalen englischen Computerbefehlen umgehen können. Das Programm ist gegenüber anderen Versionen extrem reduziert gestaltet und konzentriert sich somit auf das Wesentliche. Der schwarze Bildschirmhintergrund wirkt gut. Statt der berühmten „Schildkröte“ markiert hier ein kleines Dreieck den „Zeichenstift“.
UCB-Logo ist **Freeware**. Im Internet unter: www.cs.berkeley.edu/~bh
(Dort im zweiten Abschnitt auf den Link „windows“ klicken. Ein komplettes englisches Manual (Hilfe) befindet sich nach der Installation im Programmverzeichnis UCBLogo\DOCS.)
Eine deutsche Beschreibung von Logo findet sich z. B. auf den Seiten der Uni Hamburg.
www.informatik.uni-hamburg.de/~schaetti/LOGOeinstieg.html

**Ein elementares Beispiel – ein Quadrat:**



Das Programm für diese Grafik lautet:

| **Befehl** | **Erklärung** |
|---|---|
| *forward 100* | Zeichne vorwärts eine Linie mit 100 Längeneinheiten. |
| *left 90* | Drehe dich um 90° nach links. |
| *forward 100* | Zeichne vorwärts eine Linie mit 100 Längeneinheiten. |
| *left 90* | Drehe dich um 90° nach links. |
| *forward 100* | Zeichne vorwärts eine Linie mit 100 Längeneinheiten. |
| *left 90* | Drehe dich um 90° nach links. |
| *forward 100* | Zeichne vorwärts eine Linie mit 100 Längeneinheiten. |

Jede einzelne Vorwärtsbewegung wie auch jede einzelne Drehbewegung wird in separaten Schritten programmiert.

Die Programmiersprache umfasst sehr viele Befehle. Aber bereits mit einigen wenigen Befehlen kann man interessante Ergebnisse erzielen. Nach unserer Erfahrung ist es gut, wenn man den Kindern anfänglich lediglich die einführenden Befehle zeigt (siehe unten). Man sollte den Kindern viel Zeit zum Ausprobieren und Entdecken lassen. So können sie selbst erleben, dass eine zusätzliche Funktion sinnvoll wäre und einen neuen Befehl in ihre Arbeiten integrieren. Die „Spezialisten“ unter den Kindern werden möglicherweise schnell ein Niveau erreichen, das die Lehrerkenntnisse übersteigt...

Bewährt hat sich, dass die Kinder ein Heft/einen Ordner anlegen, in dem sie alle neuen Befehle, die sie ausprobiert haben, von Hand notieren. Das ergibt ein eigenes „Handbuch“. Für alle gelungenen Formen, die die Kinder programmiert haben, schreiben sie die „Programme“ auf. Die Bildschirm-Grafiken können abgezeichnet, ausgedruckt oder fotografiert werden.

Viele der Logo-Befehle können in Kurzform eingegeben werden. Es scheint aber sinnvoll zu sein, wenn man die Abkürzungen nicht zu früh zeigt, damit das Verständnis der englischen Befehle besser geübt werden kann.

Die folgenden Listen sind – im Sinne des oben Dargestellten – für die Hand des begleitenden Erwachsenen gedacht.

**LOGO-Befehle 1. Stufe – Einführung**

Vorbemerkung: Beim Programmstart öffnet sich das Logo-Fenster nur verkleinert. Es kann – windowsüblich – mit Klick rechts oben maximiert werden. (Oder einmalig Logosymbol auf dem Desktop – Rechtsklick – Eigenschaften – Ausführen Maximiert.) Nach dem ersten Befehl schaltet die Darstellung auf den schwarzen Grafikmodus um.
*Kursiv* in der Tabelle bedeutet, dass hier ein Zahlenwert eingegeben werden muss.

| **Befehl** | **Kurzform** | **Beschreibung** |
|---|---|---|
| forward *Länge* | fd *Länge* | Der Zeiger bewegt sich um eine bestimmte Anzahl von Einheiten *(Pixel)* nach vorne. |
| right *Winkel* | rt *Winkel* | Der Zeiger dreht sich um einen bestimmten Winkel *(Grad)* nach rechts. |
| left *Winkel* | lt *Winkel* | Der Zeiger dreht sich um einen bestimmten Winkel *(Grad)* nach links. |
| home | | Der Zeiger bewegt sich zur Mitte des Bildschirms mit Ausrichtung nach oben. |
| clean | | Der Bildschirm wird gelöscht. Die Position des Zeigers ändert sich nicht. |
| cleanscreen | cs | Alles wird gelöscht – Ausgangszustand. |

Die geometrischen Grundformen (Polygone) können bereits mit diesem Repertoire an Befehlen gezeichnet werden.

**LOGO-Befehle 2. Stufe – Weiterführung und Gestaltung**

| **Befehl** | **Kurzform** | **Beschreibung** |
|---|---|---|
| back *Länge* | bk *Länge* | Der Zeiger bewegt sich um eine bestimmte Anzahl von Einheiten *(Pixel)* zurück. |
| setpencolor *Z* | setpc *Z* | Stiftfarben:<br>*0* schwarz, *1* blau, *2* grün, *4* rot, *6* gelb, *7* weiß, |
| hideturtle | ht | Der Zeiger wird unsichtbar. |
| showturtle | st | Der Zeiger wird sichtbar. |

| | | |
|---|---|---|
| setbackground *Z* | setbg *Z* | Hintergrundfarbe: *siehe setpencolor* |
| setpensize *Z* | | Linienbreite *Z* in Pixel. |
| penup | pu | Der Stift wird von der Zeichenfläche genommen. |
| pendown | pd | Der Stift wird auf die Zeichenfläche gesetzt. |
| penerase | pe | Stift löscht. |
| penpaint | ppt | Stift schreibt. |

**LOGO-Befehle 3. Stufe – Hinführung zur Programmierung**

| **Befehl** | **Kurzform** | **Beschreibung** |
|---|---|---|
| fullscreen | | Maximaler Grafikbereich auf dem Bildschirm. |
| textscreen | | Maximaler Textbereich auf dem Bildschirm. |
| repeat *n* [*L W*] | | Befehl [*Länge Winkel*] wird *n* mal wiederholt.<br>Statt Länge und Winkel können in der Klammer auch andere Befehle stehen. |
| arc *Bogen Radius* | | Kreisbogen *(Grad)*, beginnend über Zeiger-Spitze mit bestimmtem *Radius*. |
| setxy [*x y*] | | Bewegt den Zeiger an die Koordinaten *x* und *y*. |
| savepict "*Name* | | Speichert die Grafik in Windows unter „Eigene Dateien“. Die Anführungszeichen zu Beginn des Namens legen fest, dass es sich um einen Namen handelt. |
| loadpict "*Name* | | Lädt die gespeicherte Grafik auf den Bildschirm. |
| to *Name*<br>...<br>end | | Mit *Name* wird ein komplettes Programm benannt. Mit „to“ und „end“ wird das Programm begonnen und beendet. Das Programm befindet sich im temporären Speicher; es wird nicht dauerhaft gespeichert. |
| *Name* | | Startet das geschriebene und benannte Programm aus dem temporären Speicher. |
| edit "*Name* | ed "*Name* | Öffnet den Windows-Editor und bereitet ein zu schreibendes Programm mit „to ... end“ vor.<br>Diese Datei im Editor unter „Eigene Dateien“ (mit Endung .txt) abspeichern. Die Datei kann im Editor beliebig verändert werden. |
| load "*Name.txt* | | Lädt das Programm in den temporären Speicher. Aufgerufen wird es durch *Name* (s. o.). |
| :a :b ... | | Variablen nach Doppelpunkt hinter einem Namen |
| label [...] | | Text innerhalb der eckigen Klammer wird angezeigt. |

**Aufgaben-Vorschläge für den Anfang**

1. Quadrate in verschiedenen Größen, z. B. vier Mal
   ```
   forward 200
   right 90
   ```
2. Gekipptes Quadrat (mit „right 45“ beginnen)
3. Mehrere gekippte Quadrate – jeweils um z. B. 15° verschoben
4. Andere Vierecke: Rechteck, Raute, Drachen, Parallelogramm...
5. Das „Haus vom Nikolaus“
6. Regelmäßige Vielecke (Polygone)

**Vorschläge für weiterführende Aufgaben**

1. Regelmäßige Polygone mit „repeat“ programmieren.
   ```
   repeat 6 [fd 200 rt 60]
   ```
   Dabei jeweils die Außenwinkel, Innenwinkel und Winkelsummen aufschreiben:
   *„Das regelmäßige Sechseck (Hexagon) hat einen Außenwinkel von 60°, einen Innenwinkel von 120° und eine Winkelsumme von 720°.“*
2. Andere – möglicherweise auch nicht geschlossene – Figuren erzeugen, indem man andere Außenwinkel angibt, als die regelmäßigen Vielecke haben, z. B.
   ```
   repeat 9 [fd 300 rt 80]
   ```
3. Regelmäßige Polygone ineinander darstellen. Das Programm für Dreieck bis Zehneck:
   ```
   pu
   bk 250
   pd
   lt 90
   ht
   repeat 3 [fd 180 rt 120]
   repeat 4 [fd 180 rt 90]
   repeat 5 [fd 180 rt 72]
   repeat 6 [fd 180 rt 60]
   repeat 8 [fd 180 rt 45]
   repeat 9 [fd 180 rt 40]
   repeat 10 [fd 180 rt 36]
   repeat 7 [fd 180 rt 51.42858]
   ```
4. Kreise, Kreisbögen ...
5. Mit „to ... end“ richtige Programme schreiben und speichern.
6. Geometrische Figuren mit der Eingabe von x/y-Koordinaten konstruieren.

7. Verwendung von Variablen:

   Ein benanntes Programm kann mit Variablen nach Doppelpunkt ergänzt werden – z. B.

   ```
   to Stern :a
   ```

   Diese Variablen können ihrerseits bestimmt werden, z. B.

   ```
   setpc :a
   ```

   Der Aufruf „Stern 2“ zeichnet dann die Grafik in grüner Farbe.