Programmieren mit LOGO

Digitale Strukturen – Programmieren als Schlüssel zur Welt

Von Thomas Helmle und Markus Wurster (2008/2020/2023)

Wenn die Digitalisierung ein Grundmuster unserer Lebenswelt darstellt und zur umfassenden Schlüsseltechnologie wurde, dann ist es unsere Aufgabe, Kindern einen „Schlüssel“ im Sinne Montessoris anzubieten, mit dem sie sich diese Welt aufschließen können, mit dem sie selbst auf Entdeckungsreisen gehen und Schlüsselerfahrungen sammeln können.

Wir möchten dazu eine Einführung ins Programmieren mit der Software „Logo“ anbieten. Mit Logo können geometrische Formen und Grafiken programmiert bzw. gezeichnet werden. Logo wurde in den sechziger Jahren entwickelt und wird von den damaligen Autoren und Schülern noch immer in didaktischer Motivation weiter gepflegt. Das heute etwas nostalgisch anmutende Konzept erlaubt uns einen genetischen Zugang zum Thema im Sinne Wagenscheins. Wir gehen experimentierend und selbsttätig vor, erleben intellektuelle und ästhetische Triumpfe und sehen unserer eigenen steilen Lernkurve zu. Ab etwa 8/9 Jahren sind Kinder dazu in der Lage.

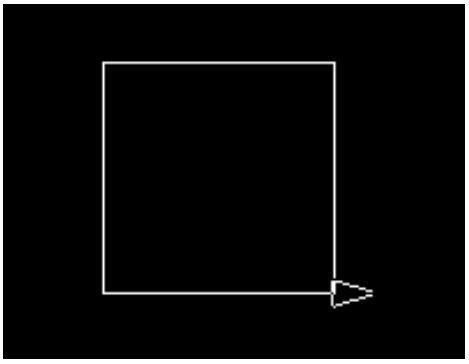
Wir spannen ein Feld auf, in dem sich zwei Botschaften herauskristallisieren: Computer sind keine klugen Zaubermaschinen. Eigentlich sind sie dumm. Alles, was sie können, können sie nur, weil sie von Menschen gut programmiert wurden. Unser menschlicher Geist ist der eigentliche Zauberer. Wir sind es, die mit Algorithmen, mit schrittweisen Lösungsfolgen, zum Ziel kommen.

Die Künstliche Intelligenz lassen wir hier noch beiseite. Erfreuen wir uns dafür an der intelligenten Kunst…

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** (derzeit Version 6.2.3 vom 16.01.2023). Im Internet unter:
www.cs.berkeley.edu/~bh
(Dort im vierten Abschnitt auf den Link „windows“ klicken. Ein komplettes englisches Manual (Hilfe) befindet sich nach der Installation im Verzeichnis C:\ucblogo\docs.)

Eine deutsche Beschreibung von Logo mit Programmier-Beispielen für höhere Klassen findet sich z. B. hier: www.cadae.de/ucblogo/

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. Man zeigt dann neue Befehle eher auf Nachfrage. 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 (siehe Kopiervorlage). Das ergibt ein eigenes „Handbuch“. Für alle gelungenen Formen, die die Kinder programmiert haben, schreiben sie die „Programme“ auf. Die Bildschirm-Grafiken können als Grafik exportiert, gespeichert oder gedruckt werden („Print Turtle Graphics). Dabei wird die Grafik schwarz auf weiß ausgegeben.

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.

Vorbemerkung zum Programmstart: Das Logo-Fenster öffnet sich nur verkleinert. Es kann – windowsüblich – mit Klick rechts oben maximiert werden. (Oder für die dauerhafte Einstellung einmalig Logosymbol auf dem Desktop mit rechter Maustaste anklicken – Eigenschaften – Ausführen Maximiert.) Nach dem ersten Befehl schaltet die Darstellung auf den schwarzen Grafikmodus um.

Tipp: Einmal geschriebene Zeilen können bei LOGO leider nicht mehr verändert bzw. korrigiert werden. Fehler sind dadurch etwas frustrierend, weil man immer wieder von vorne anfangen muss. Bewährt hat es deshalb, den gesamten Code zunächst in einem Editor oder Textverarbeitungsprogramm zu schreiben. Diesen Code kann man per „copy and paste“ ins Logo-Fenster übertragen. Fehler lassen sich so verbessern und verbesserte Codes einfach neu einfügen.

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

LOGO-Befehle 1. Stufe – Einführung (Figur)

Kursiv in der Tabelle bedeutet, dass hier ein Zahlenwert eingegeben werden muss.
„Pixel“ für Länge; „Grad“ für Winkel

Elementare Befehle

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.
cleanscreen	cs	Alles wird gelöscht – Ausgangszustand.

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

Weiterführende Befehle

Befehl	**Kurzform**	**Beschreibung**
back *Länge*	bk *Länge*	Der Zeiger bewegt sich um eine bestimmte Anzahl von Einheiten *(Pixel)* zurück.
arc *Kreisbogen Radius*	-	*Kreisbogen (Grad)*, beginnend über Zeiger-Spitze mit bestimmtem *Radius*.
repeat *n* [*L W B*]	-	Befehl [*Länge Winkel Bogen*] wird *n* mal wiederholt. Statt Länge und Winkel können in der Klammer auch andere Befehle stehen.
setxy [*x y*]	-	Bewegt den Zeiger an die Koordinaten *x* und *y*.
clean	-	Der Bildschirm wird gelöscht. Die Position des Zeigers ändert sich nicht.
home	-	Der Zeiger bewegt sich zur Mitte des Bildschirms mit Ausrichtung nach oben.

Befehle zur Gestaltung (Bildschirm und Linien)

Befehl	**Kurzform**	**Beschreibung**
hideturtle	ht	Der Zeiger wird unsichtbar.
showturtle	st	Der Zeiger wird sichtbar.
setpencolor *Z*	setpc *Z*	Stiftfarben*: 0* schwarz*, 1* blau*, 2* hellgrün*, 3* cyan*, 4* rot*, 5* magenta*, 6* gelb*, 7* weiß*, 8* braun*, 9* ocker*, 10 d*unkelgrün*, 11* hellblau*, 12* rosa*, 13* lila*, 14* orange*, 15* grau…
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.
setbackground *Z*	setbg *Z*	Hintergrundfarbe: *siehe setpencolor*
label [...]	-	Text innerhalb der eckigen Klammer wird angezeigt.
fullscreen	-	Maximaler Grafikbereich auf dem Bildschirm.
textscreen	-	Maximaler Textbereich auf dem Bildschirm.
splitscreen	-	Grafik- und Textbereich (geteiltes Fenster)

LOGO-Befehle 2. Stufe – Hinführung zur Programmierung

Befehl	**Kurzform**	**Beschreibung**
to *Name* ... 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.
save *"Name*	-	Speichert das geschriebene Programm im Verzeichnis „Eigene Dateien“ bzw. „Dokumente“. Die Anführungszeichen zu Beginn des Namens legen fest, dass es sich um einen Namen handelt.
load "*Name* load *"Name.txt*	-	Lädt das Programm aus dem Verzeichnis „Dokumente“ in den temporären Speicher von LOGO.
Name	-	Startet das geschriebene und benannte Programm aus dem temporären Speicher.
savepict "*Name*	-	Speichert die Grafik in Windows unter „Eigene Dateien“. Die Datei ist nur innerhalb des Programms LOGO darstellbar.
loadpict "*Name*	-	Lädt die gespeicherte Grafik auf den Bildschirm.
:a :b ...	-	Variablen nach Doppelpunkt hinter einem Namen
+ - * /	-	Rechenoperationen, auch in Verbindung mit Variablen

Einführung unpluggt, Spiele

1. Schatzkarte (Kopiervorlage): Nur „vorwärts“, „rechts“, „links“. Einheit Kästchen. Geheimcode verstehen, Weg einzeichnen.
2. Einführungsspiel: Mit Partner, Augen zu oder verbunden. Raum mit Hindernissen.
   Angabe von „Katzendepperle“ und Winkel-Maß; Gehen und Drehen sind je eigene Befehle.

Hinweis: Dieser Algorithmus („Programm“) hat ca. zehn Zeilen. Im Vergleich: Windows 11 hat ca. 50 Millionen Zeilen.

Aufgaben-Vorschläge für den Anfang

1. Quadrate in verschiedenen Größen, z. B. vier Mal wiederholt:

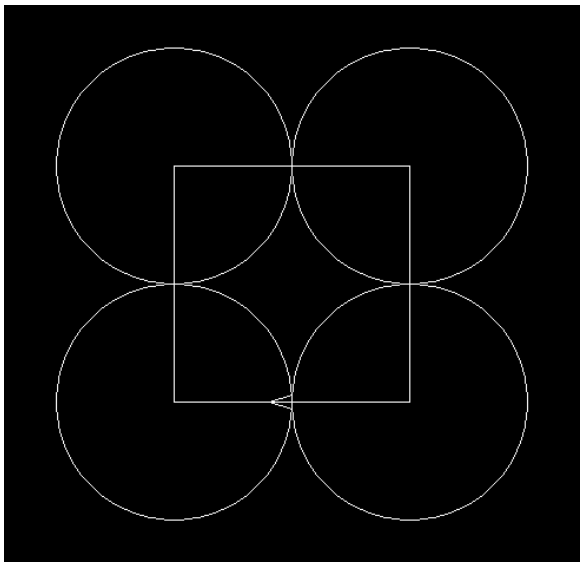
   ```
   forward 200
   right 90
   ```

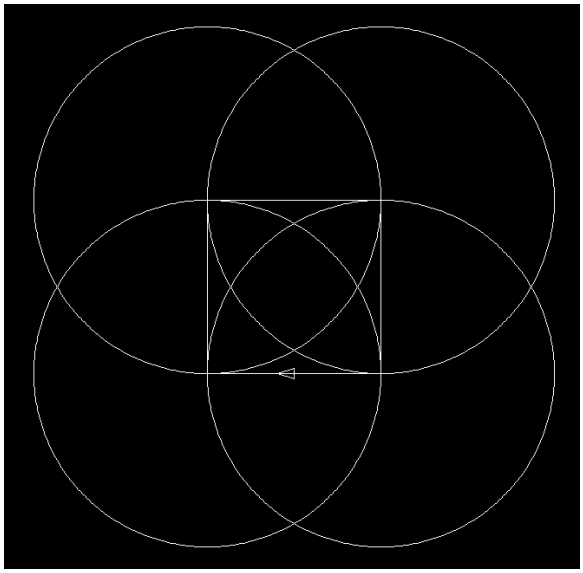
   Achtung: Längeneinheit sind „Pixel“ (winzige Punkte auf Bildschirm) und Grad
2. Gekipptes Quadrat (mit „right 45“ beginnen)
3. Mehrere gekippte Quadrate – jeweils um z. B. 15° verschoben
4. Dreiecke in verschiedenen Größen, z. B.

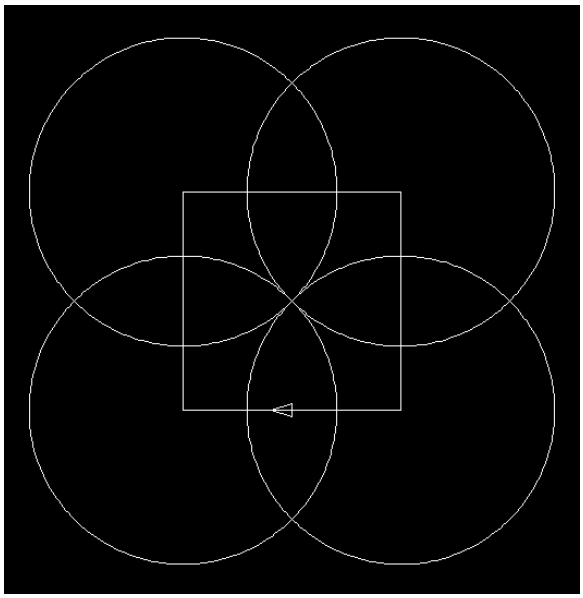
   ```
   left  90
   forward 200
   right 120
   forward 200
   right 120
   forward 200
   ```

   Achtung: Diese Winkel sind keine Innenwinkel des Dreiecks, sondern bezeichnen die Richtungsänderung; hier sind es also die Außenwinkel des Dreiecks (bzw. die Ergänzungswinkel der Dreiecks-Innenwinkel).
5. Andere Vierecke: Rechteck, Raute, Drachen, Parallelogramm...
6. Regelmäßige Vielecke (Polygone) aus der „Geometrischen Kommode“ nachkonstruieren
7. Haus mit Fenster und Tür.
8. Gegenständliche Bilder, Auto…

9. Kreise im Viereck

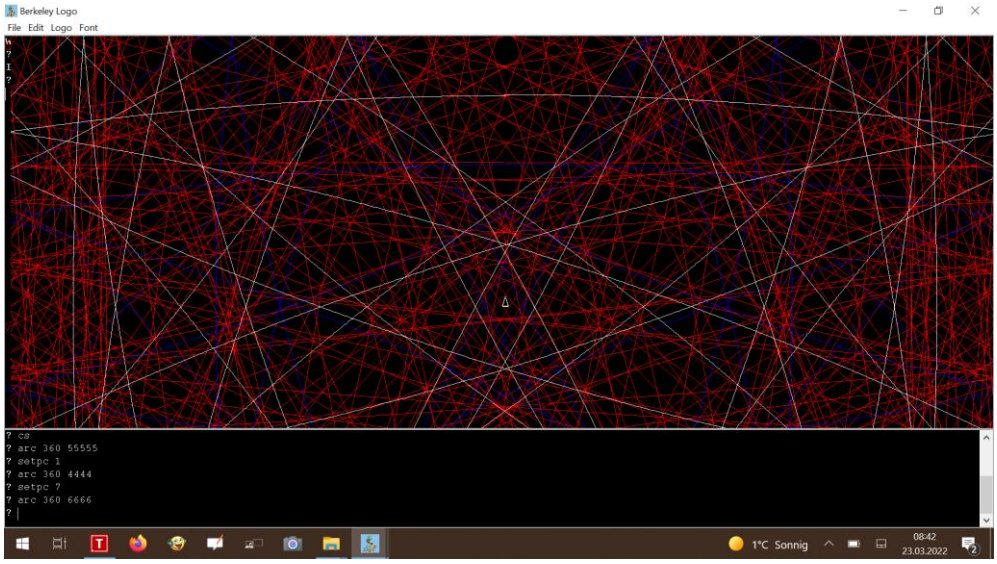






(Achtung: Bei dieser Form muss man die Radien – Diagonalen – mit Wurzel aus 2 berechnen!)

10. Solche Kreise in einem anderen Polygon
11. Abstrakte Kunst (sehr viele Wiederholungen oder sehr lange Linien, die den Bildschirm „überlaufen“ lassen). Diesen Effekt entdecken sehr viele Kinder ganz von alleine.



Vorschläge für weiterführende Aufgaben

1. Regelmäßige Polygone mit „repeat“ programmieren. (Programmierer müssen Arbeit sparen!)

   repeat 6 [fd 200 rt 60]

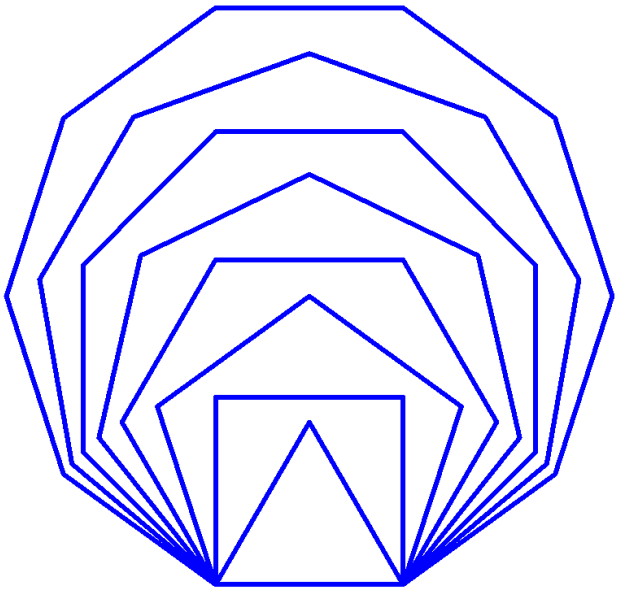
   Dabei jeweils die Außenwinkel, Innenwinkel und Winkelsummen aufschreiben:

   „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.

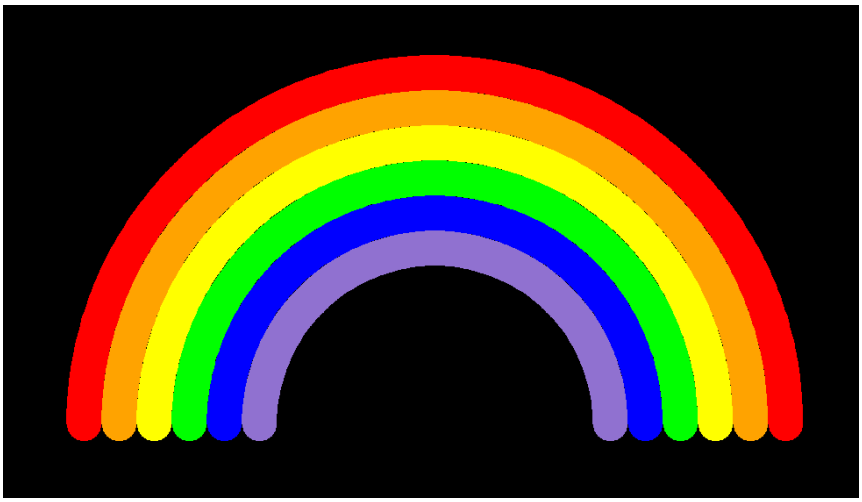
   repeat 9 [fd 300 rt 80]

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]
(Statt der Winkelangaben könnte man [360/n] schreiben)

4. Kreise, Kreisbögen ...
   Regenbogen (Farben, Liniendicke)



```
lt 90
setpensize 40
ht
setpc 13
arc 180 200
setpc 1
arc 180 240
setpc 2
arc 180 280
setpc 6
arc 180 320
setpc 14
arc 180 360
setpc 4
arc 180 400
```

5. Mit „to ... end“ richtige Programme schreiben und speichern.
   Mit Return abgeschlossene Zeilen können später nicht mehr korrigiert werden!
   Deshalb ist es sinnvoll, das Programm (genauer: die „Prozedur“) in einem Editor-Programm zu schreiben und mit der Endung [.txt] unter „Dokumente“ abzuspeichern.

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 Dreiecke :a
```

   Der Aufruf „Dreiecke *100*“ führt die Prozedur aus und setzt für a jeweils den Wert 100 ein.
   Beispiel:

```
to Dreiecke :a
ht
```

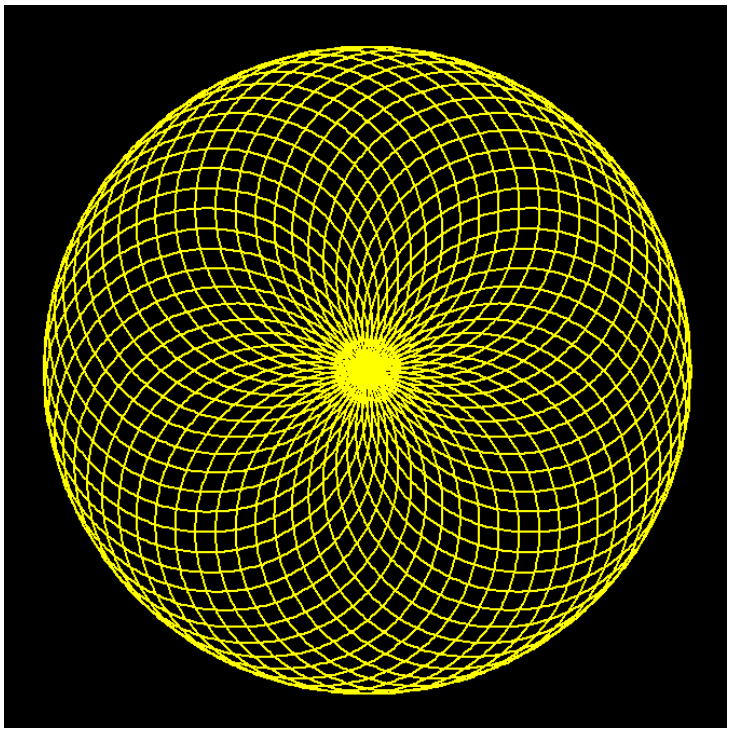
```
lt 90
fd a/2
rt 120
fd a
rt 120
fd a
rt 120
fd a/2
rt 90
end
```

So kann man mit dem Aufruf „Dreiecke *Länge1*“ „Dreiecke *Länge2*“ … verschieden große Dreiecke in einer Zeichnung erzeugen.

Es geht auch mit mehreren Variablen – Beispiel (n=Seitenzahl; l=Seitenlänge):

```
to Polygon :n :l
lt 90
fd l/2
repeat n-1 [rt 360/n fd l]
rt 360/n
fd l/2
rt 90
end
```

Spielerei für Erwachsene und fortgeschrittene Jugendliche



Das Programm für diese Blume:

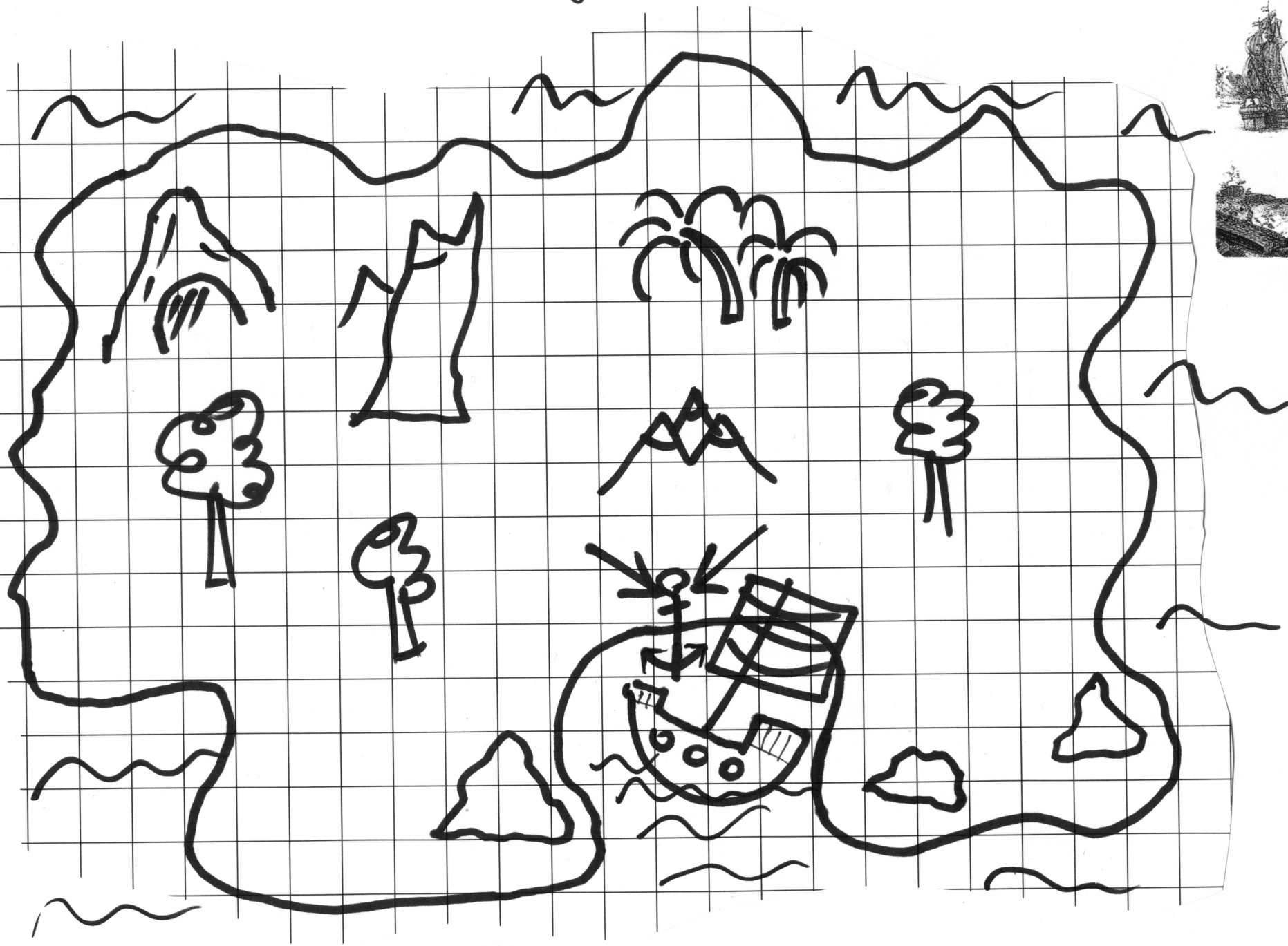
```
to Blume :n :r :f
setpc f
ht
fd r/3
lt 90
fd r/2
repeat n [pd arc 360 r pu fd 2*r*RADSIN(3.1415/n) rt 360/n]
end
```

Erklärung:
Die Kreise werden um die Eckpunkte eines regelmäßigen Polygons gezogen.
Die Seitenlänge dieses Polygons wird in Abhängigkeit der Eckzahl n und des Radius r nach dieser Formel berechnet:

$$a = 2\,r_u \cdot \sin\left(\frac{\pi}{n}\right)$$

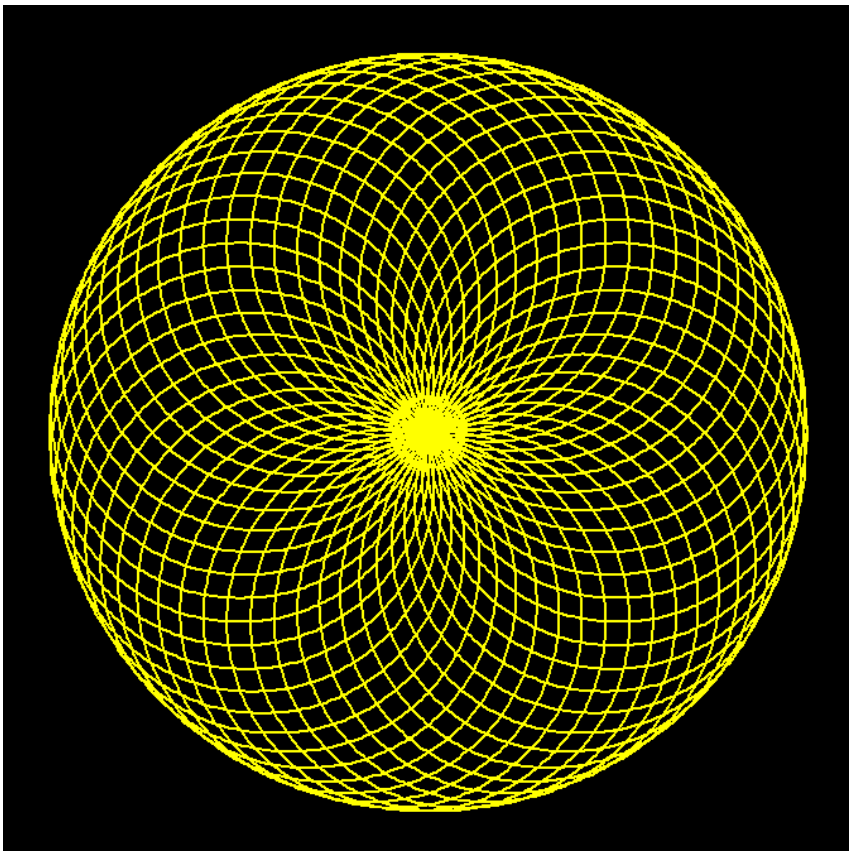
In LOGO ist es der Befehl „RADSIN“.
n definiert die Eckzahl des Polygons
r definiert den Radius der Kreise
f definiert die Farbe
Wenn das Programm „Blume.txt“ geladen ist, wird es im Beispiel der obigen Abbildung folgendermaßen gestartet:
Blume 50 150 6

Anne Bonny 1782



V 3
R 4
R 4
L 3
L 6
L 11
L 6
R 3
R 7
L 2

Programmieren mit LOGO



Programmier-Befehle

Befehl	Kurzform	Erklärung

Meine Programm-Codes

Name der Figur:

Code

Meine Programm-Codes

Name der Figur:

Code