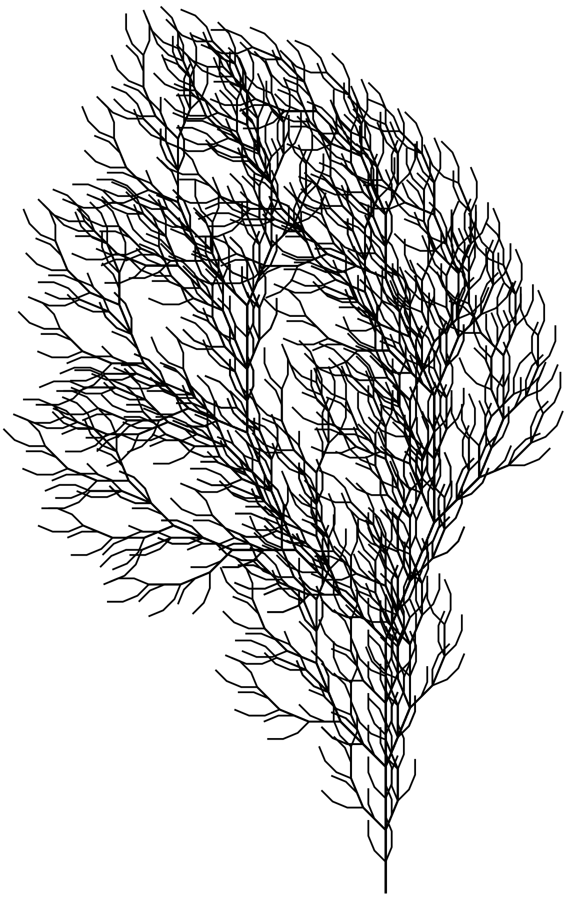
Pflanzen, DOL-Systeme und Computergrafik
Einführung und Bedienungshinweise für das Programm

Bernd Gärtner, Alexander Förster

18. Mai 1995



1 Einleitung

Dieses Skript beschreibt die grundlegende Idee hinter den sogenannten *DOL-Systemen*, die es ermöglichen, auf einfache Weise komplizierte Bilder zu zeichnen (z.B. Pflanzen wie auf dem Titelblatt). Gleichzeitig beschreiben wir hier die Funktionsweise des mitgelieferten Programms zum Erzeugen dieser Bilder. Das Programm läuft unter DOS mit VGA-Grafik.

Der erste Teil des Skripts ist im Rahmen des Berliner Tags der Mathematik am 13. Mai 1995 entstanden und wurde begleitend zur Präsentation 'Pflanzen und Computergrafik' von Bernd Gärtner ausgegeben. Wir beschreiben hier, wie man zu den Bildern kommen kann, die wir auf Folien oder direkt am Rechner gezeigt haben. Das Programm dazu (geschrieben in `C++` unter UNIX) wurde vom Autor Alexander Förster für DOS überarbeitet und befindet sich zusammen mit vielen Beispielbildern (sämtliche Beispiele aus dem Skript sind enthalten) auf der beiliegenden Diskette. Die Programmbenutzung wird im zweiten Teil des Skripts erklärt.

Das Skript ist in sehr kurzer Zeit entstanden, enthält sicher Fehler und Ungenauigkeiten und geht nicht detailliert auf alle 'features' des Programms ein. Trotzdem hoffen wir, daß die Software benutzbar ist und Schülern, Lehrern und anderen interessierten Menschen zusammen mit der 'theoretischen' Einführung einen gewissen Einblick gibt. Natürlich können wir auch für die Korrektheit des Programms nicht garantieren – zumindest funktioniert es aber für alle mitgelieferten Beispiele.

Das Programm darf beliebig verwendet und weitergegeben werden, solange auch diese Einführung kopiert und mitgegeben wird. Für Fragen und Anregungen stehen wir gerne zur Verfügung:

Bernd Gärtner
Freie Universität Berlin
Fachbereich Mathematik und Informatik
Institut für Informatik
Takustr.9
14195 Berlin

2 Eine Schildkröte, die malen kann

Eine Schildkröte sitzt auf einem Blatt Kästchenpapier (und zwar genau auf der Ecke eines Kästchens). Sie schaut in eine der vier Richtungen (links, rechts, oben, unten). Die Schildkröte kann drei Befehle befolgen:

- Befehl +: Drehe dich um 90^0 gegen den Uhrzeigersinn (das heißt, wenn du z.B. vorher nach rechts geschaut hast, dann schaue jetzt nach oben).
- Befehl −: Drehe dich um 90^0 im Uhrzeigersinn (das heißt, wenn du vorher nach rechts geschaut hast, dann schaue jetzt nach unten).
- Befehl **F**: Gehe ein Kästchen in Blickrichtung und markiere deinen Weg mit einem dicken schwarzen Strich.

Zeichnen wir die Schildkröte als Punkt und die Blickrichtung als Pfeil, so gibt uns das einen *Zustand* der Schildkröte, der sich durch die Befehle ändert. Das folgende Bild zeigt die drei Möglichkeiten:

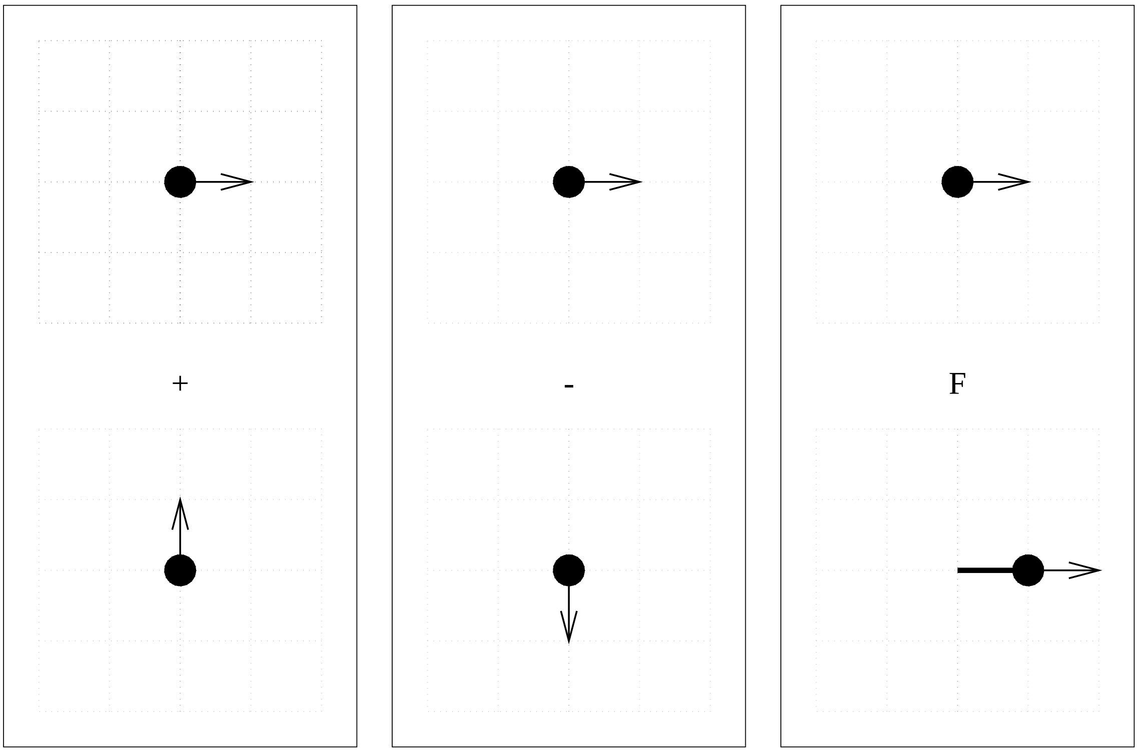


Abbildung 1: Die drei Schildkrötenbefehle

Im allgemeinen wird die Schildkröte nicht nur einen Befehl, sondern eine ganze Folge von Befehlen bekommen, die sie dann nacheinander ausführt. Auf diese Weise können wir die Schildkröte dazu bewegen, ein Bild zu malen.

Übung 1.

(1) Im linken Teil des Bildes auf dieser Seite siehst du, wie die Schildkröte zunächst im Anfangszustand ist (oben); dann geben wir ihr eine Folge von Befehlen (in diesem Fall F+F+FF+F–F–), und unten siehst Du den Zustand der Schildkröte nach Ausführung dieser Befehle. Überprüfe ob die Schildkröte wirklich genau das angebene 'Bild' malt und zum Schluß auf dem richtigen Feld und mit der richtigen Blickrichtung landet!

(2) Im mittleren Teil des Bildes fehlt der Weg der Schildkröte und ihre Position und Blickrichtung zum Schluß. Führe die Befehle für die Schildkröte aus und zeichne das Ergebnis im unteren Teil ein!

(3) Im rechten Teil des Bildes ist der Weg und Endzustand der Schildkröte eingezeichnet – es fehlt aber die Befehlsfolge dazu! Versuche, eine Folge von Befehlen zu finden, die genau das Bild im unteren Teil erzeugt!

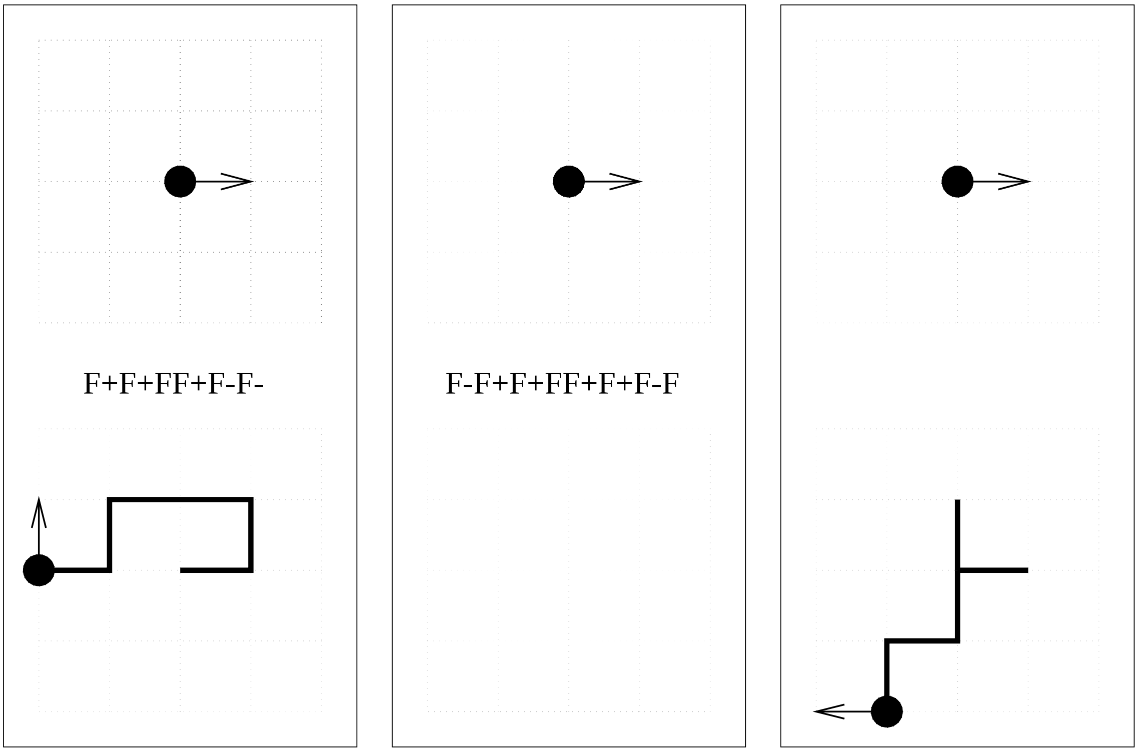


Abbildung 2: zu Übung 1

3 Die Schildkröte wird fleißig

Wir haben gesehen, daß die Schildkröte Befehlsfolgen verarbeiten kann, die aus den Zeichen $+, -$ und F bestehen, und für jedes F in der Folge malt sie einen Strich aufs Papier. Nun möchten wir ja eigentlich sehr komplizierte Bilder mit vielen Strichen malen, und wenn wir für jeden Strich ein F hinschreiben müßten, würde das eine sehr lange Befehlsfolge ergeben. Darauf haben wir keine Lust. Wenn wir aber faul sind, sollte zumindest die Schildkröte fleißig sein. Das ist sie auch, denn sie überlegt sich den folgenden Trick, um mehr Striche malen zu können:

> Ich habe eine relativ kurze Befehlsfolge, möchte aber eigentlich ein schönes großes Bild malen. Da könnte ich doch jedesmal, wenn ich den Befehl F bekomme und eigentlich einen Strich malen müßte, anstattdessen die ganze Befehlsfolge ausführen und damit jedesmal einen komplizierteren 'Strich' wie etwa den in Abbildung 2 auf der linken Seite malen.

Die Schildkröte wird also 'ungehorsam' und verändert eigenständig ihre Befehlsfolge, indem sie jedes F durch die gesamte Befehlsfolge ersetzt. Hier ist ein Beispiel:

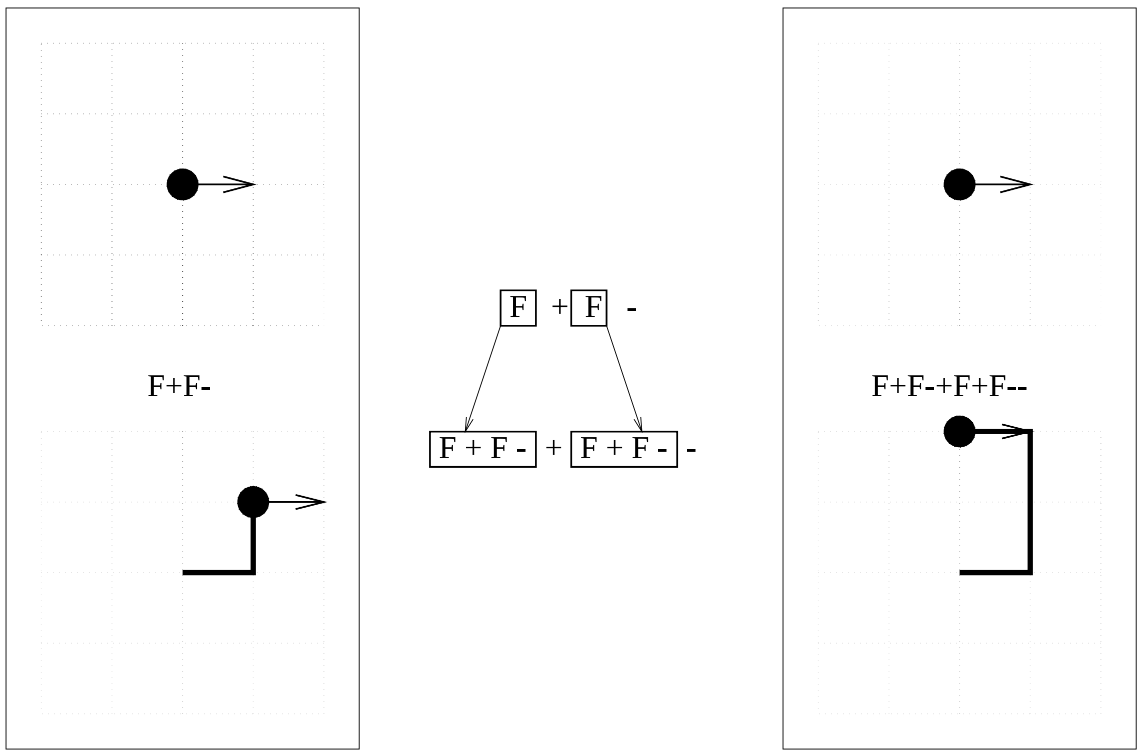


Abbildung 3: Der Ersetzungsschritt der fleißigen Schildkröte

Der linke Teil des Bildes zeigt eine Befehlsfolge und ihre Ausführung durch die Schildkröte. In der Mitte siehst du, wie aus der kurzen Befehlsfolge nach der Regel 'ersetze jedes F durch die ursprüngliche Befehlsfolge' eine neue, längere Befehlsfolge wird, die auch ein anderes (geringfügig komplizierteres) Bild erzeugt (rechter Teil).

Übung 2.

Jetzt sollst Du die fleißige Schildkröte spielen. Gegeben ist die Befehlsfolge F+F−−F+F. Vervollständige die untere Abbildung wie in Abbildung 3: schreibe die Befehlsfolge auf, die nach der Ersetzung entsteht und male jeweils die Bilder zur ursprünglichen und zur ersetzten Befehlsfolge!

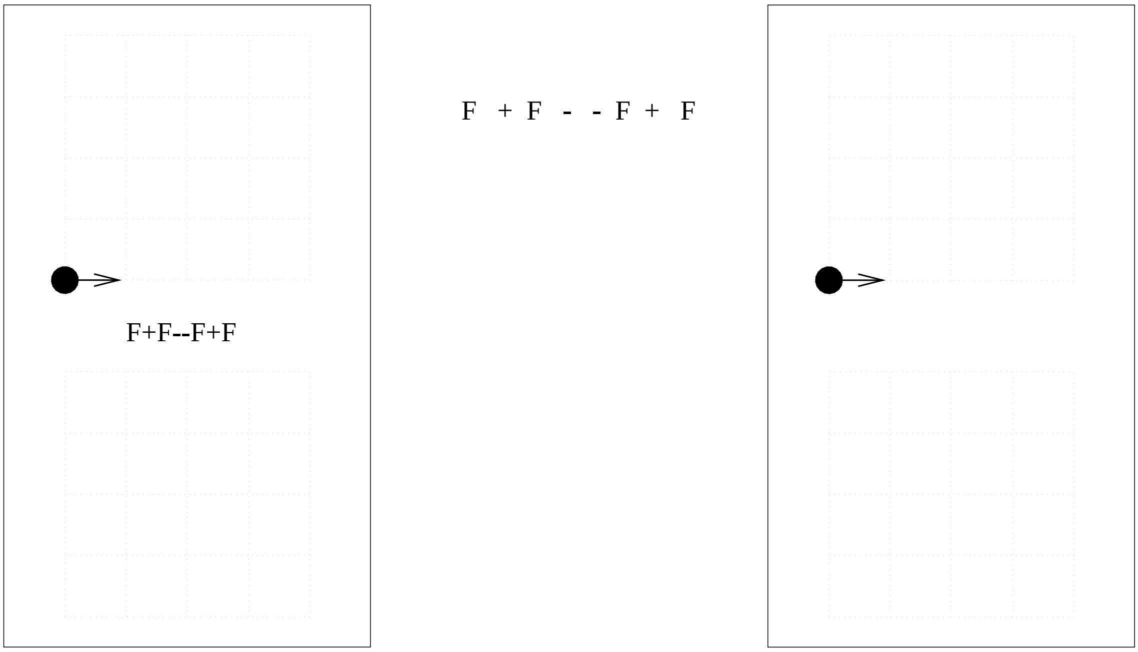


Abbildung 4: zu Übung 2

4 Die Schildkröte wird noch fleißiger

Das nach der Ersetzung erhaltene Bild ist der Schildkröte immer noch nicht schön genug, und du wirst zugeben, dass auch dein Bild aus Übung 2 noch nicht sehr interessant aussieht. Aber es spricht ja nichts dagegen, die Ersetzung *nochmal* zu machen, das heißt, in der längeren Befehlsfolge wieder alle F's durch die ursprüngliche Befehlsfolge zu ersetzen. Die folgende Abbildung zeigt auf der linken Seite die zwei Ersetzungsschritte und auf der rechten Seite das Bild, das dann von der zweimal ersetzten Befehlsfolge erzeugt wird.

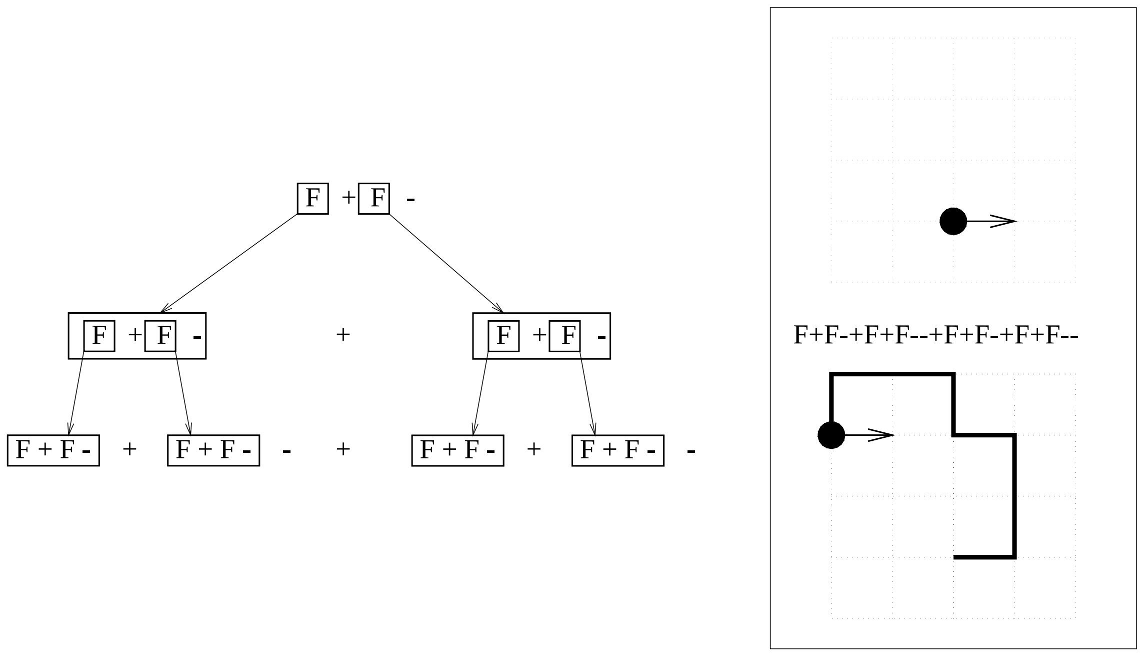


Abbildung 5: Der zweite Ersetzungsschritt der noch fleißigeren Schildkröte

Dem Fleiß der Schildkröte sind natürlich keine Grenzen gesetzt. Wenn sie immer noch nicht zufrieden ist mit dem bisher entstandenen Bild, so kann sie in der zweimal ersetzten Befehlsfolge *nochmal* alle F's durch die ursprüngliche Befehlsfolge ersetzen, und nochmal, und nochmal...

Wir wissen jetzt, wie wir eine Befehlsfolge aus der nächsten konstruieren können: bezeichnen wir mit B_0 die ursprüngliche, mit B_1 die einmal ersetzte, mit B_2 die zweimal ersetzte Befehlsfolge dann gilt in unserem Beispiel folgendes:

$$\begin{aligned} B_0 &= \mathrm{F{+}F{-}} && \text{(die ursprüngliche Befehlsfolge),} \\ B_1 &= B_0 + B_0 - && \text{(denn wir haben ja alle F's durch } B_0 \text{ ersetzt).} \end{aligned}$$

Was ist B_2, die zweimal ersetzte Folge? Nun, wir ersetzen in B_1 alle F's durch B_0. Das heißt, wir müssen in den beiden B_0's die in B_1 vorkommen, die F's ersetzen. Die B_0's werden dadurch aber zu B_1, das war ja gerade der erste Ersetzungsschritt! Also gilt:

$$\begin{aligned} B_2 &= B_1 + B_1 -, \text{ und ebenso für den } n\text{-ten Ersetzungsschritt} \\ B_n &= B_{n-1} + B_{n-1} - . \end{aligned}$$

Das heißt, wir haben jetzt zwei verschiedene Methoden, um die nächste Befehlsfolge zu berechnen: um B_n zu erhalten können wir

- in B_{n-1} alle F's durch B_0 ersetzen (die ursprüngliche Methode), oder
- in B_0 alle F's durch B_{n-1} ersetzen (das hatten wir uns ja gerade überlegt).

Wenn wir B_{n-1} schon ausgerechnet haben, ist die zweite Methode natürlich wesentlich praktischer, weil wir dann B_{n-1} nur zweimal aneinanderhängen und die $+$ und $-$ Zeichen an den richtigen Stellen einfügen müssen.

Nach diesen etwas abstrakten Überlegungen wird es wieder konkreter. Die Schildkröte überlegt sich, daß sie es mit dem Fleiß auch nicht übertreiben möchte – insbesondere hat sie gar keine Lust, die Ersetzungen in den Befehlsfolgen durchzuführen. Viel lieber möchte sie malen. Sie möchte also wissen: wenn ich das Bild für die ursprüngliche Befehlsfolge B_0 gemalt habe, wie bekomme ich dann das Bild für die einmal ersetzte Befehlsfolge B_1, ohne groß herumzurechnen? Na ja, sie weiß ja, daß B_1 aus zweimal B_0 und $+, -$ an den richtigen Stellen besteht, nämlich

$$B_1 = B_0 + B_0 - .$$

Um das Bild zu B_1 zu malen, überlegt sich die Schildkröte also einfach folgendes:

- Ich beginne im Startzustand und male das Bild für B_0.
- Dann drehe ich mich um 90^0 gegen den Uhrzeigersinn (Befehl $+$).
- Vom dann erreichten Zustand aus male ich *nochmal* das Bild für B_0 (das gibt dann das gleiche Bild wie beim ersten Mal, nur daß es woanders anfängt und um 90^0 gedreht ist).
- Schließlich drehe ich mich nur noch einmal um 90^0 *im* Uhrzeigersinn (Befehl $-$) und befinde mich im Endzustand nach Malen des Bildes zu B_1.

Als Beispiel sehen wir uns an, wie die Schildkröte in unserem Beispiel mit Befehlsfolge F+F− das Bild zur ersten Ersetzung B_1 (Abbildung 3) erhält:

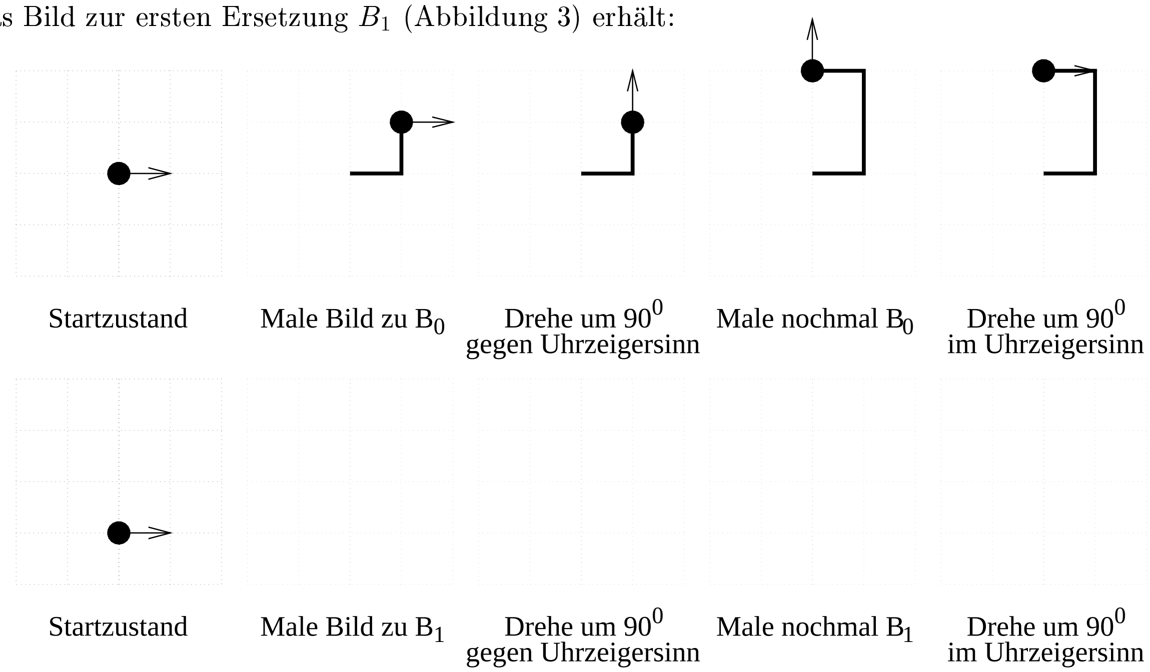


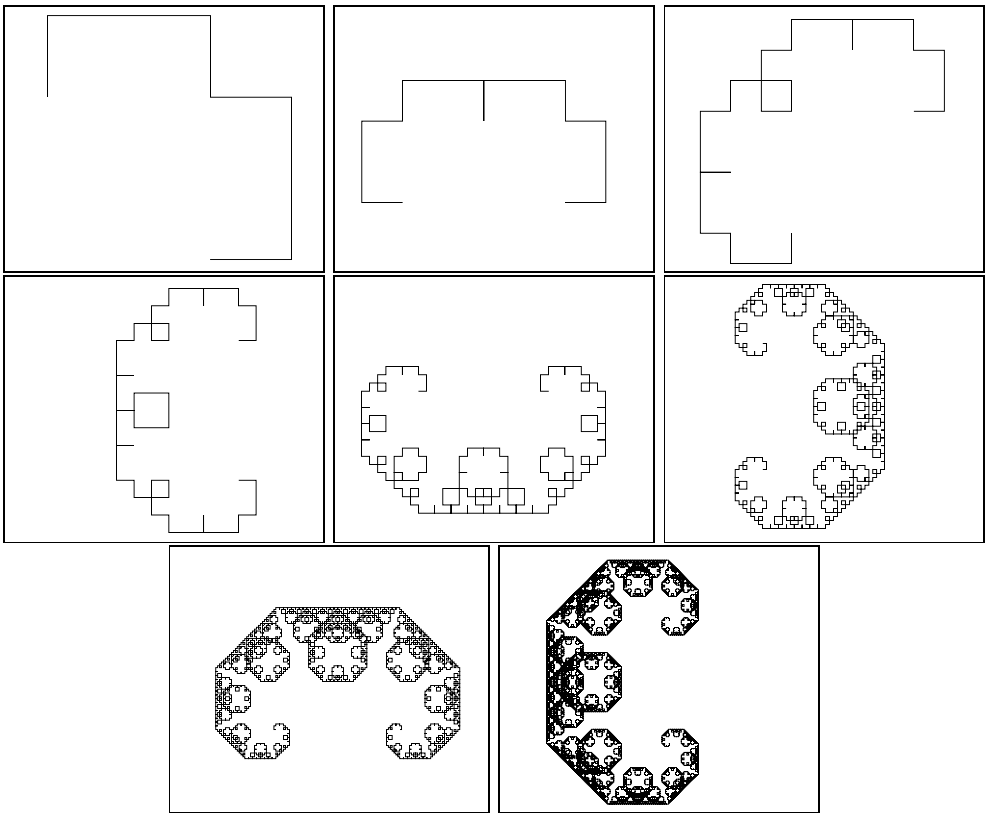
Abbildung 6: Konstruktion des nächsten Bildes aus dem vorhergehenden

Übung 3. Vervollständige den unteren Teil des Bildes, d.h. konstruiere analog das Bild zur zweiten Ersetzung B_2 aus dem Bild zur ersten Ersetzung B_1 und vergleiche dein Ergebnis mit Abbildung 5!

5 Die Schildkröte malt Fraktale

Bisher sind wir nie über die zweite Ersetzung hinausgekommen, und das komplizierteste Bild war noch das in Abbildung 5. Aber wir haben ja in Übung 3 dieses Bild schon ganz einfach aus dem vorhergehenden konstruieren können, und nach dem gleichen Schema könnten wir weitermachen. Das Problem ist, daß jedesmal, wenn wir ein neues Bild malen, sich die Anzahl der gemalten Striche verdoppelt! Zähle es ruhig mal nach: im Bild zur ursprünglichen Befehlsfolge haben wir zwei Striche (Abbildung 3 links); daraus werden vier Striche für die einmal ersetzte Befehlsfolge (Abbildung 3 rechts) und sogar acht Striche für die zweimal ersetzte Folge (Abbildung 5). Das ist auch klar, denn um das nächste Bild zu bekommen, malen wir ja das vorhergehende zweimal.

Um also etwa das zehnte Bild zu malen, muß die Schildkröte $2^{10} = 1024$ Striche ziehen – das wollen wir dann auch ruhig ihr überlassen, anstatt es von Hand zu versuchen. Eine andere Möglichkeit ist natürlich, den Computer als Schildkröte zu benutzen. Genau das habe ich gemacht, und die folgenden Abbildungen zeigen, wie sich das Bild bei mehr und mehr Ersetzungen entwickelt. Hättest du gedacht, daß es so aussieht? Gezeigt sind die uns schon bekannte zweite Ersetzung, gefolgt von den Ersetzungen $3, 4, 5, 6, 10, 12, 14$, allerdings jeweils so verkleinert, daß sie aufs Papier passen.



Je mehr Ersetzungen die Schildkröte durchführt, desto komplizierter wird das Bild. Wenn du genau hinschaust, siehst du, daß jedes Bild in verkleinerter Darstellung im nächsten Bild auftaucht, und zwar zweimal. Das muß nach unserer Konstruktion natürlich auch so sein, denn das Bild zur n-ten Ersetzung haben wir ja gerade erhalten, indem wir zweimal das Bild zur $n-1$-ten Ersetzung gemalt haben.

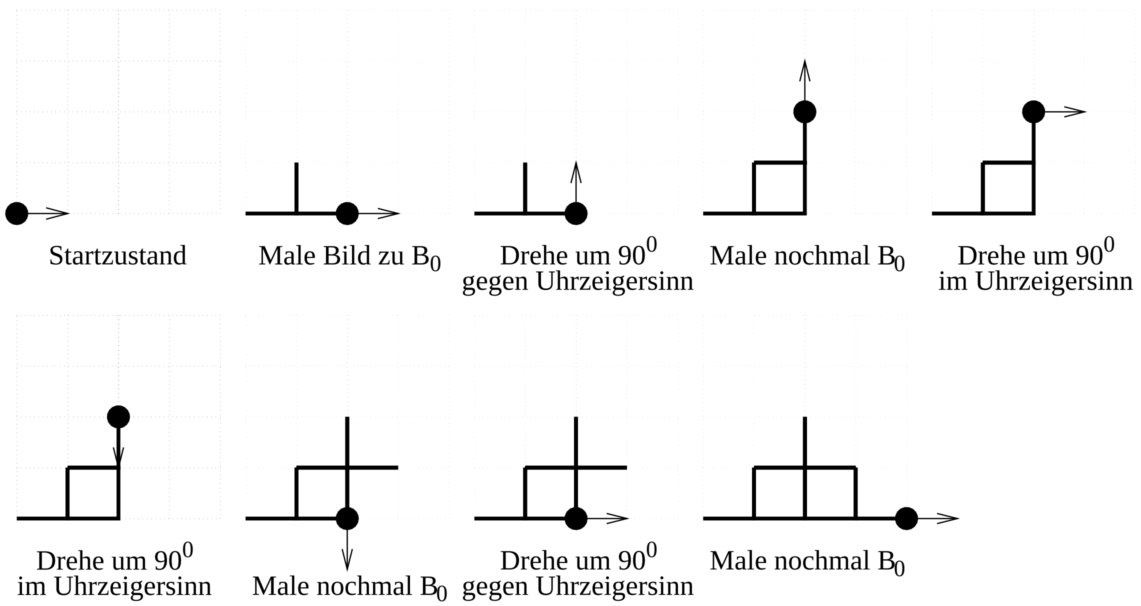
Obwohl die Bilder am Anfang wirklich komplizierter werden, sehen sich die Bilder zu Ersetzung 12 und 14 in der untersten Reihe erstaunlicherweise ziemlich ähnlich, und man kann sich theoretisch überlegen, daß sich da auch nicht mehr viel tut, wenn man noch mehr Ersetzungen macht. Das liegt an der Verkleinerung, die die noch bestehenden Unterschiede für das Auge unsichtbar macht. Wenn man die Bilder wieder sehr stark vergrößern würde, könnte man sehen, daß es sehr wohl noch große Unterschiede gibt.

Selbst wenn man die Ersetzungen bis ‘unendlich’ durchführt, sieht das Bild in der Verkleinerung nicht viel anders aus als das zu Ersetzung 14. Wir haben dann ein *Fraktal.* Als Teile tauchen zwei Bilder zur Ersetzung ‘unendlich minus 1’ auf. Das ist aber genauso unendlich – das Fraktal besteht also aus zwei Kopien von sich selbst! So etwas nennt man Selbstähnlichkeit: egal wie stark man das Bild vergrößert, man sieht als Teil immer wieder das ganze Bild.

In Übung 2 (Abbildung 4) hast du ja schon für eine andere Befehlsfolge – nämlich F+F−−F+F – das Bild gemalt, und auch das Bild zur ersten Ersetzung konstruiert. Da wir ja nun wissen, wie man dieses Bild direkt aus dem ersten Bild bekommt, ohne die Ersetzungen wirklich durchzuführen, ist es vielleicht ganz gut, das hier nochmal zu tun, um zu sehen, ob wirklich das gleiche herauskommt. Zur Erinnerung: die ersetzte Befehlsfolge B_1 ergibt sich wie folgt:

$$B_1 = B_0 + B_0 - -B_0 + B_0,$$

und wie im anderen Beispiel auch können wir das Bild dazu bekommen, indem wir insgesamt viermal das Bild zu B_0 malen und dazwischen die richtigen Drehungen ausführen, wie sie in der Befehlsfolge gegeben sind:



Übung 4. Wie sieht das Bild zur dritten, vierten, fünften,. . . ,hundertsten Ersetzung aus? Versuche darauf zu kommen, indem du dir überlegst, wie sich ein Bild vom nächsten unterscheidet!

6 Die Schildkröte malt Pflanzen

Das Fraktal auf Seite 7 sieht vielleicht ganz nett aus, hat aber kaum Ähnlichkeit mit einer Pflanze wie etwa der auf dem Titelbild. Solche Pflanzen haben als Strichzeichnungen zwei Eigenschaften, die wir bisher nicht gut realisieren können:

- Pflanzen haben üblicherweise keine 90^0-Winkel
- Pflanzen haben viele Verzweigungen

Das Problem mit den Winkeln kann man ganz einfach lösen: die Schildkröte kann sich ja z.B. auch um 60^0 gegen den Uhrzeigersinn drehen, wenn sie den Befehl $+$ bekommt und um 60^0 *im* Uhrzeigersinn bei $-$. Jeder andere Winkel ist natürlich genauso möglich. Wir müssen nur am Anfang einmal einen Winkel festlegen. Nicht mehr ganz so klar ist, was dann der Befehl F bewirken soll; hat sich die Schildkröte z.B. zunächst um 60^0 gedreht wie in Abbildung 7 links, so kann sie sich in Blickrichtung zwar bewegen, aber nicht mehr auf den Kästchen. Wie weit soll sie denn dann überhaupt gehen? Antwort: genauso weit wie vorher, also die Entfernung, die einem Kästchen entspricht. Zwar landet die Schildkröte dann nicht mehr auf einer Ecke, aber das macht nichts. Eigentlich können wir die Kästchen dann sogar ganz vergessen. Wichtig ist nur, daß die Schildkröte immer beim Befehl F eine feste Entfernung in Blickrichtung zurücklegt und dabei ihren Strich malt. Abbildung 7 zeigt ein Beispiel dafür. Insgesamt hat sich die Schildkröte um $6 \times 60^0 = 360^0$ gedreht und schaut deshalb zum Schluß in die gleiche Richtung wie am Anfang.

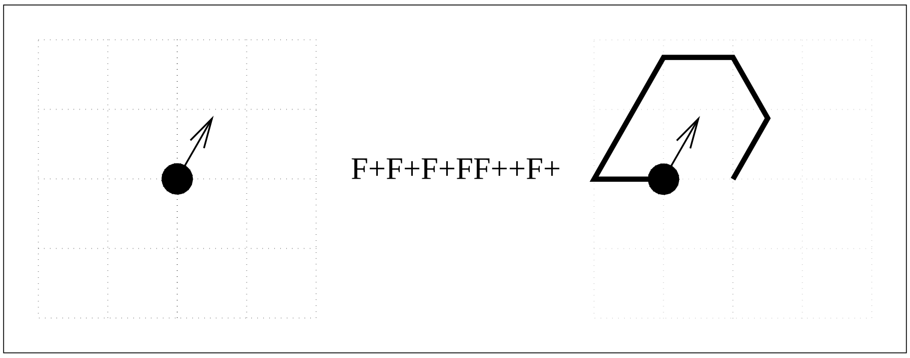


Abbildung 7: Ein Bild, erzeugt von der Folge F+F+F+FF++F+ mit Drehwinkel 60^0

Was die Verzweigungen betrifft, so *kann* die Schildkröte schon Bilder mit Verzweigungen malen (siehe z.B. Abbildung 2 rechts). Sie muß aber gewisse Striche zweimal malen, um nach Malen eines Zweiges zum Verzweigungspunkt zurückzukehren, an dem dann auch noch der andere Zweig hängt. Könnte sie direkt zum Verzweigungspunkt zurückspringen, würde das Arbeit sparen. Deshalb nehmen wir einfach ein, daß sie es kann! Dazu führen wir zwei neue Befehle ein:

- Befehl [: Merke dir den Zustand, in dem du jetzt gerade bist, das heißt merke dir deine Position und Blickrichtung.
- Befehl]: Springe direkt zurück zum letzten Zustand, den du dir gemerkt hast, d.h. nimm wieder die enstprechende Position und Blickrichtung ein.

Die folgende Abbildung gibt ein Beispiel. Der Winkel ist 30^0. Zuerst wird ein senkrechter Strich gezeichnet. Die Schildkröte ist dann am Verzweigungspunkt. Dort merkt sie sich den Zustand, dreht um 30^0, zeichnet den linken Strich und kehrt zum Verzweigungspunkt zurück, wo sie sich wieder den Zustand merkt. Dann dreht sie um 30^0 in die andere Richtung, zeichnet den rechten Strich und kehrt zur Verzweigung zurück. Schließlich malt sie noch den zweiten senkrechten Strich und bleibt danach im Endzustand stehen.

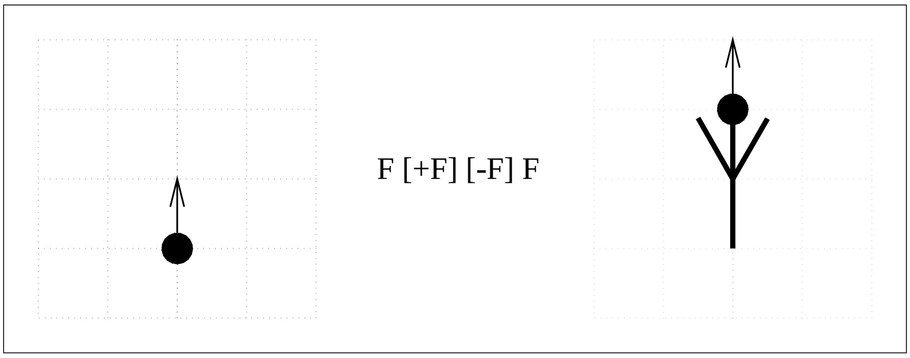
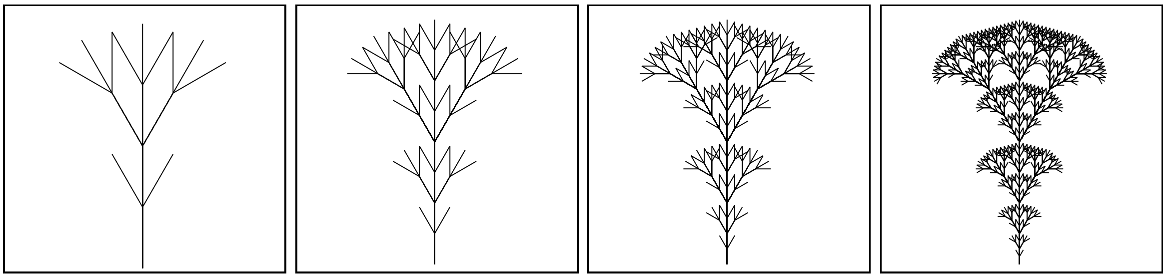
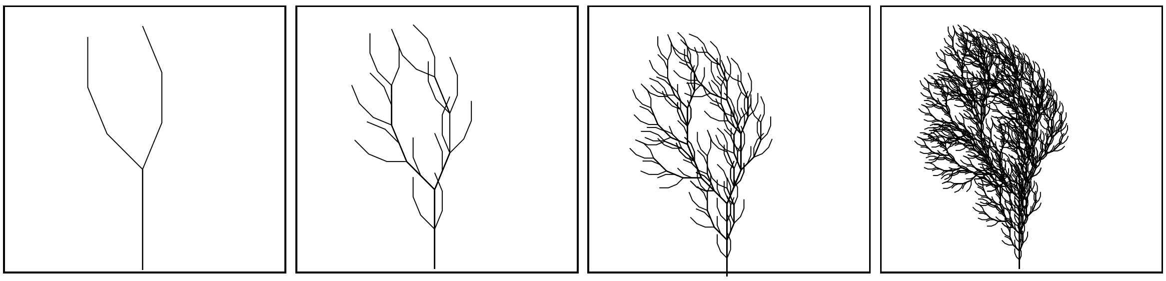


Abbildung 8: Ein Bild, erzeugt von der Folge F[+F][−F]F mit Drehwinkel 30^0

Das sieht doch schon ein wenig wie eine kleine Pflanze aus. Zum Glück funktioniert alles, was wir bisher über das Ersetzen der F's und das Konstruieren eines Bildes aus dem vorhergehenden gesagt haben, genauso, wenn noch die Befehle [und] dazukommen. Deshalb können wir jetzt die Pflanze 'wachsen' lassen: wir lassen die Schildkröte – wie gehabt – Bild für Bild konstruieren:



Schließlich fehlt noch die Pflanze auf der Titelseite. Diese wird mit Winkel 22.5^0 und der Befehlsfolge FF−[−F+F+F]+[+F−F−F] erzeugt:



7 DOL-Programm Bedienungsanleitung

Eine Befehlsfolge zusammen mit Drehwinkel und Anzahl der gewünschten Ersetzungsschritte heißt DOL-System (oder einfach L-System). Das Programm liest eine Datei ein, in der ein solches L-System angegeben ist (wie? siehe unten) und zeichnet den Weg der Schildkröte.

Installation.

Das Programm ist in C++ geschrieben und liegt im Quellcode sowie als ausführbare Datei für MS-DOS vor. Das Programm ist mit Borland C++ 3.1 kompiliert, läßt sich aber auch mit Borland C++ 2.0 kompilieren. Zum Ausführen des Programms benötigt man die Dateien `dol.exe`, `egavga.bgi` sowie eine `*.dol` Datei, die die Definition eines L-Systems enthält. Wenn man das Programm selbst kompilieren möchte, so benötigt man noch die Datei `dol.cpp` (Wie man ein Programm kompiliert wird hier nicht erklärt). Zum Installieren kopiert man alle Dateien der Diskette in ein Verzeichnis der Festplatte. Um zu sehen, ob das Programm funktioniert, ruft man z.B.

```
dol dragon.dol
```

auf. Am besten ist es, wenn man das Programm in den Norton Commander (nc) integriert, denn dann kann man die Grammatik eines Systems schnell ändern und sich das Ergebnis sofort ansehen. Zur Integration des Programms in den nc sucht man im Menu des nc den Eintrag Erweiterungsdatei bearbeiten, trägt dort (bei der Version 3.0)

```
dol: dol !.!
```

ein und speichert die Datei. Jetzt kann man durch auswählen einer `*.dol` Datei und drücken der Return-Taste die Schildkröte zeichnen sehen.

Bedienungshinweise

Das Programm hat keine Bedienungsoberfläche. Jede Grammatik wird in einer eigenen Datei gespeichert und beim Aufruf des Programms als Parameter übergeben. Falls die Datei fehlerhaft ist, wird der Grund des Fehlers angegeben, jedoch keine Zeilennummer, in der der Fehler aufgetreten ist. Man muß also etwas suchen, um den Fehler zu finden. Falls kein Fehler aufgetreten ist muß man etwas warten, bis nach einiger Zeit die Schildkröte zu zeichnen beginnt. Wenn das Bild fertig ist, kann durch Drücken irgendeiner Taste das Programm verlassen werden.

Die Syntax der `*.dol` Dateien

Ein Beispiel sagt mehr als 1000 Worte, deshalb:

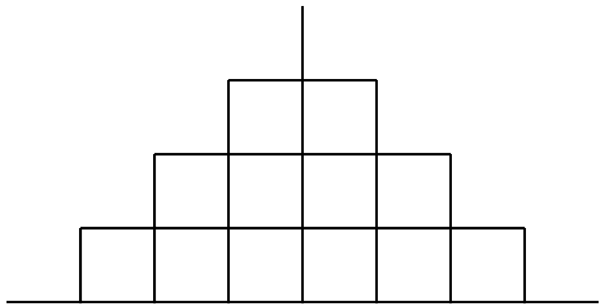
```
symbols {
   F; +; -;
}
axiom {
   F;
```

```
}
productions {
   F => F+F--F+F;
}
angle {
   90;
}
iterations {
   2;
}
```

Dieses Beispiel erzeugt die Zeichenkette

`F+F--F+F+F+F--F+F--F+F--F+F+F+F--F+F`

und das Bild auf Seite 9 unten. Mit 3 Iterationen ergibt sich das folgende Bild:



Ein weiteres, komplizierteres Beispiel zeigt, daß auch andere Zeichen erlaubt sind, mehr als eine Ersetzungsregel angegeben werden kann etc. Unten wird genau erklärt, wie die Datei aussehen muß. Diese Datei erzeugt die sogenannte Drachenkurve.

```
symbols {
  F; f; +; -; |; [ ; ] ; X; Y;
  }
axiom {
  -X;
}
productions {
 X => X+YF+;
 Y => -FX-Y;
 }
angle {
 90;
 }
iterations {
 16;
 }
```

Die Schlüsselwörter `symbols`, `axiom`, `productions`, `angle` und `iterations` müssen in jeder `*.dol`-Datei vorkommen. Nach den Schlüsselwörtern folgt immer eine geschweifte öffnende Klammer.

- `symbols`
  Hier werden alle Symbole angegeben, die unter `axiom` sowie `productions` vorkommen. Jedes Symbol wird von einem Semikolon gefolgt. (Wir haben im ersten Teil nur die Symbole F, $+$, $-$, [und] betrachtet, das Programm versteht aber noch andere Zeichen.) Zum Schluß gibt es noch eine geschweifte schließende Klammer.

- `axiom`
  Hier kommt die Zeichenkette hin, mit der wir anfangen wollen. Das war im Skript immer eine relativ kurze Befehlsfolge. Die kürzeste Befehlsfolge die überhaupt ein Bild erzeugt, besteht aus dem einzigen Zeichen F. Oft wird man in den Beispielen F als anfängliche Befehlsfolge finden. Zum Schluß gibt es wieder eine Semikolon und eine Klammer.

- `productions`
  Hier wird angegeben, wie die Symbole ersetzt werden sollen. Links steht das zu ersetzende Symbol, in der Mitte `=>` und rechts die Ersetzung, welche durch ein Semikolon beendet wird. Falls ein Symbol hier nirgends auf der linken Seite erscheint, so wird es durch sich selbst ersetzt. Jedes Symbol darf auf der linken Seite nur einmal vorkommen. Wenn alle Ersetzungen angegeben sind, so gibt es – wie immer – eine Klammer. Im Skript war das einzige Symbol, das wir ersetzt haben, das F. Auf der rechten Seite der Ersetzung steht dann die Befehlsfolge, durch die wir F jeweils ersetzen möchten. Anstatt – wie im Skript – mit einer Befehlsfolge B_0 anzufangen und jedes F darin wieder durch die Befehlsfolge B_0 zu ersetzen, können wir auch mit F selbst anfangen. Dann kommen wir eben erst nach der ersten Ersetzung zu B_0, nach der zweiten zu B_1 usw.

  Das Programm erlaubt es uns sogar, die anderen Symbole wie $+$ und $-$ zu ersetzen; das wird aber selten gemacht.

- `angle`
  Hier wird der Winkel angegeben, um den sich die Schildkröte dreht, wenn sie auf ein `+` oder `-` trifft. Auch hier gibt es zum Schluß ein Semikolon und eine geschweifte schließende Klammer.

- `iterations`
  Hier wird angegeben, wie viele Ersetzungsschritte durchgeführt werden sollen, bis die entstandenen Befehlsfolge von der Schildkröte gezeichnet wird. Zum Testen sollte man hier mit kleinen Zahlen beginnen, da man sonst sehr lange auf die Ausgabe warten muß. Und noch einmal ein Semikolon und eine Klammer.

Welche Zeichen die Schildkröte interpretiert

- `F`
  Die Schildkröte bewegt sich einen Schritt in die Richtung, in die sie gerade schaut und zeichnet dabei eine Linie.

- `f`
  Auch hier bewegt sich die Schildkröte einen Schritt in die Richtung, in die sie gerade schaut,

jedoch zeichnet sie dabei nicht (das hatten wir noch nicht betrachtet, ist aber manchmal ganz praktisch).

- `+`
  Die Schildkröte dreht sich gegen den Uhrzeigersinn um den in `angle` angegebenen Winkel.

- `-`
  Die Schildkröte dreht sich im Uhrzeigersinn um den in `angle` angegebenen Winkel.

- `|`
  Die Schildkröte dreht sich um 180 Grad (dieser Befehl ist überflüssig für den Drehwinkel 90^0, den wir im Skript die meiste Zeit betrachtet haben, weil er in diesem Fall durch $++$ oder $--$ simuliert werden kann. Für andere 'krumme' Drehwinkel aber benötigt man ihn).

- `[`
  Die Schildkröte merkt sich die Position, an der sie sich gerade befindet und ihre Blickrichtung. Sie legt diese Information auf einem Stapel ab.

- `]`
  Die Schildkröte springt zu der Position (und Blickrichtung), die sie sich zuletzt gemerkt hat, d.h. sie nimmt diese Information vom Stapel herunter.

Alle anderen Symbole (wie z.B. X und Y) werden von der Schildkröte nicht interpretiert. Sie werden nur benutzt um Symbolketten zu erzeugen, die kompliziertere Objekte darstellen, wie zum Beispiel die Drachenkurve im obigen Beispiel.

Weiterführende Literatur

The Algorithmic Beauty of Plants Prusinkiewicz; Lindenmayer, Springer 1990. Wie der Titel verrät ist es ein englischsprachiges Buch. Es wird auch auf dreidimensionale Welten, zufällige Bewegungen und Farben eingegangen.

The Science of fractal images Peidgen; Saupe, Springer 1988. Diese Buch gibt es auch auf deutsch. Es beschäftigt sich nicht nur mit DOL-Systemen, sondern auch mit anderen Fraktalen,wie z.B. Mandelbrotmengen. Im Anhang gibt es Hinweise zum schreiben von eigenen Programmen, die DOL-Systeme berechnen.

Die fraktale Geometrie der Natur Mandelbrot, Birkhäuser 1987. DOL-Systeme werden hier auf eine etwas andere Art eingeführt. Wie das vorherige Buch wird auch hier auf andere Themen eingegangen, z.B.: „Wie lang ist die Küste Großbritaniens?“