Mathematik und Geometrie mit Logo

Lars Widmer

11. Juni 2007

Inhaltsverzeichnis

1 Aufgaben

1.1 Geschwindigkeit berechnen

Lehrerhinweis 1.1 *Diese Aufgabe ist verhältnismässig aufwändig und führt trotzdem keine neuen Konzepte ein. Die gewonnenen Erkenntnisse werden später nicht direkt verwendet. Es handelt sich schlicht um eine Aufgabe zum Warmlaufen. Fühlen Sie sich frei diese Aufgabe nach Bedarf auch wegzulassen.*

Lehrerhinweis 1.2 *Momentan enthält das Dokument an verschiedenen Orten solche* ***Hinweise*** *für die Lehrperson. Für die Version für die Schüler können sie weggelassen werden.*

Es ist ein einfaches Programm zu schreiben, dass die Schildkröte eine längere Strecke zurücklegen lässt. Am einfachsten ist es, dazu eine Prozedur zu entwerfen, die zwei Parameter verwendet. Der erste Parameter w soll die Länge einer Strecke sein und mit dem zweiten Parameter n wird angegeben, wie oft die Strecke (in doppelter Ausführung) gezeichnet werden soll. Die Schildkröte soll also eine Strecke zeichnen, sich umdrehen und um einen Pixel versetzt eine neue Linie entlang der vorhergehenden zeichnen. Diesen Vorgang wiederholt sie (immer um einen Pixel verschoben) so oft wie mit n angegeben und zeichnet dabei einen grossen schwarzen Block. Abbildung 1 zeigt einen vergrösserten Ausschnitt eines solchen Blockes. Es darf keine Strecke zweimal gezeichnet werden.
Es ist nun die Aufgabe während der Ausführung die Sekunden abzuzählen und damit ungefähr auszurechnen, mit welcher **Geschwindigkeit** die Schildkröte unterwegs ist. Die Streckenlänge sei in Pixel gegeben. Mit der Annahme, dass 72 Pixel einer Länge von 2.54cm entsprechen, kann weiter die Geschwindigkeit in m/s und km/h umgerechnet werden. Beim Vergleichen der erhaltenen Resultate ist zu beachten, dass sie Werte abhängig vom verwendeten Computer und dessen Betriebszustand sind.

1.2 Polygon zeichnen

Es ist ein Programm zu schreiben, welches beliebige **Polygone** zu zeichnen vermag. Es sind nur Parameter für die Eckenzahl und die Seitenlänge zu verwenden. Der Winkel, um welchen an den Ecken jeweils gedreht werden muss, soll im Programm berechnet werden.
In Abbildung 2 ist ein mit dem Programm erstelltes Siebeneck zu sehen.

Lehrerhinweis 1.3 *Die gestellte Aufgabe erfordert geometrische Grundkenntnisse. Es handelt sich aber um nichts Schwieriges. Wenn die Berechnung*

Abbildung 1: Ausschnitt aus der Ausgabe des Programmes zur Geschwindigkeitsmessung

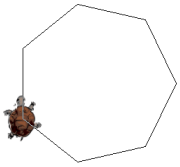

Abbildung 2: Mit dem Polygon-Programm gezeichnetes Siebeneck

der Winkel in einem Polygon noch nicht bekannt ist, kann sie vor der Aufgabe von der Lehrperson eingeführt werden. Ich empfehle folgende Erklärungen:

- *Man zeichne ein Polygon mit n Ecken.*
- *Von jeder Ecke aus, ziehe man eine Linie zum Mittelpunkt.*
- *Der Winkel zwischen zwei benachbarten von den hinzugefügten Linien, beträgt* $\alpha = \frac{360°}{n}$*. Dies kann einfach eingesehen werden, weil die n hinzugefügten Linien über* 360° *verteilt angeordnet sind. Alle verwendeten Winkel sind in Abbildung 3 eingezeichnet.*
- *Zwei benachbarte der hinzugefügten Linien bilden mit der zugehörigen Seite des Polygons ein Dreieck.*
- *Die Winkelsumme im Dreieck beträgt* 180°*.*
- *Es handelt sich um gleichschenklige Dreiecke. Damit sind die beiden äusseren Winkel* β *gleich gross. Und den Öffnungswinkel im Zentrum des Polygons kennen wir ja bereits.*

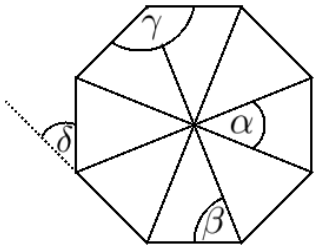

Abbildung 3: Sechseck mit allen Winkeln angeschrieben

- *Damit haben wir:*

$$\beta = \frac{180^\circ - \frac{360^\circ}{n}}{2} \tag{1}$$

- *Der Winkel γ zwischen zwei Seiten des Polygons setzt sich nun aus zweimal dem Winkel β zusammen, also $\gamma = 2 \cdot \beta$.*

- *Wenn die Schildkröte jeweils eine Seite des Polygons gezeichnet hat, muss sie sich aber nicht um γ drehen, sondern um $\delta = 180^\circ - \gamma$, weil die Schildkröte in die verkehrte Richtung schaut. Also nach dem Zeichnen einer Linie schaut die Schildkröte von der Linie weg, anstelle ihr entlang zu schauen, wie es bei der Winkelmessung sonst üblich ist.*

- *Leiten wir nun also die benötigte Formel her:*

$$\begin{aligned}
\delta &= 180^\circ - \gamma && (2)\\
&= 180^\circ - 2 \cdot \beta && (3)\\
&= 180^\circ - 2 \cdot \frac{180^\circ - \frac{360^\circ}{n}}{2} && (4)\\
&= 180^\circ - \left(180^\circ - \frac{360^\circ}{n}\right) && (5)\\
&= 180^\circ - 180^\circ + \frac{360^\circ}{n} && (6)\\
&= \frac{360^\circ}{n} && (7)
\end{aligned}$$

- *Das Resultat fällt erstaunlich einfach aus.*

1.3 Haus bauen

Lehrerhinweis 1.4 *Für diese Aufgabe ist die Kenntnis des Satzes von Phytagoras notwendig. Der Satz wir benötigt, um die Länge der Linien für das*

Schrägdach zu bestimmen. Gegeben ist nur die Breite und Höhe des gesamten Hauses.
Ebenfalls spielt der Thaleskreis in der Aufgabe mit. Das Schrägdach hat einen Öffnungswinkel von 90°*. Damit ist die Höhe des Daches genau die Hälfte der Breite des Hauses.*

In dieser Aufgabe soll ein Haus gezeichnet werden. Parameter sind die **Höhe** vom Gibel bis zum Boden, die **Breite** des Hauses und die Anzahl **Stockwerke** (ohne den Dachstock gezählt). Die Dachschräge sei immer 45°. Alle Stockwerke sind gleich hoch. Die zur Konstruktion fehlenden Werte sind im Programm zu berechnen. Die Abbildungen 4 und 5 zeigen Häuser wie sie mit dem gewünschten Programm gezeichnet wurden.

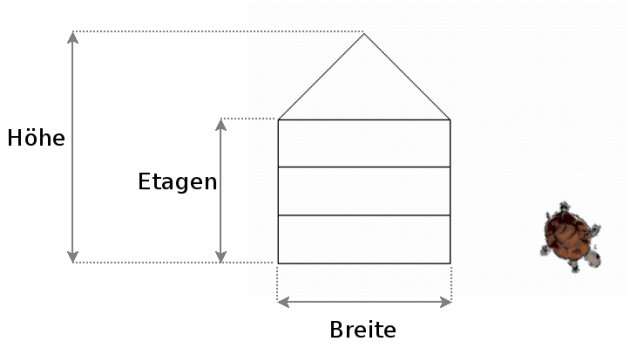

Abbildung 4: Ein simples Haus mit drei Stockwerken

Abbildung 5: Mit dem simplen Programm generiertes Hochhaus.

1.4 Haus verschönern

Lehrerhinweis 1.5 *Dies ist eine verspielte Aufgabe, die sich als freiwillige Aufgabe für die schnelleren Schüler eignet. Damit erhalten die langsameren Schüler die Möglichkeit aufzuholen. Die Aufgabe lässt sich beliebig ausbauen und kann damit sehr zeitintensiv werden. Mathematisch enthält die Aufgabe keine Erweiterungen zur vorhergehenden Aufgabe.*

Zum Schluss kann das Programm aus vorhergehender Aufgabe erweitert werden, um schönere Häuser zu zeichnen. Beispiele dazu sind auf den Abbildungen 6 und 7 zu finden. Viel Spass!

Kommando 1.1 *Falls nicht schon bekannt lohnt es sich unter Umständen für diese Aufgabe folgende Kommandos anzuschauen:*

- ***pu**: "Pen Up", "Stift anheben" Ab hier zeichnet die Schildkröte nicht mehr, wenn sie sich bewegt.*
- ***pd**: "Pen Down", "Stift absetzen" Ab hier zeichnet die Schildkröte wieder.*
- ***pe**: "Pen Erase", "Tintenkiller verwenden" bestehende Striche, werden beim Überfahren gelöscht.*

Es ist nicht das Ziel, hier möglichst viele Befehle zu lernen. Der Einsatz ist freiwillig.

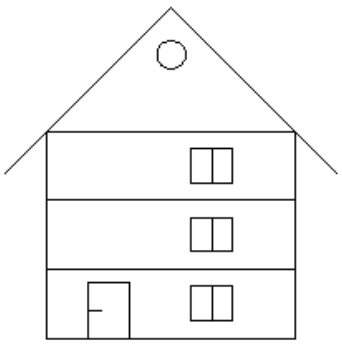

Abbildung 6: Mit einem ausgefeilteren Programm gezeichnetes Haus.

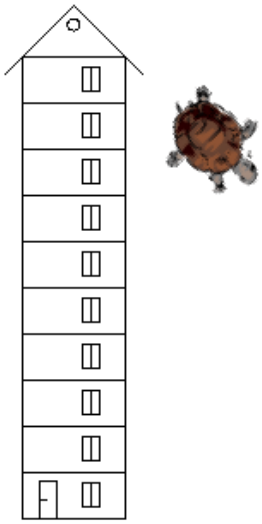

Abbildung 7: Anderes Haus, mit dem ausgefeilten Programm gezeichnet.

1.5 Pfeile zeichnen

Lehrerhinweis 1.6 *Wegen des neuen Winkels kann in dieser Aufgabe nicht mehr der Phytagoras angewendet werden. Stattdessen muss mit dem Cosinus aus der Trigonometrie gerechnet werden. Es reicht dabei die Formel für rechtwinklige Dreiecke zu kennen.*
Die Prozentrechnung wird auch gestreift, aber das stellt gewiss kein Problem dar.
Das Resultat dieser Aufgabe wird in der nächsten Aufgabe eingesetzt.

Es ist ein Logo-Programm zu erstellen, welches Pfeile zeichnet. Folgende Parameter werden übergeben:

- Länge des Pfeiles. Gemeint ist die Länge inklusive Pfeilspitze. Einen allgemeinen Pfeil sieht man auf Abbildung 8.

- Prozentuale Länge der Pfeilspitze im Vergleich zur gesamten Pfeillänge. Abbildung 9 zeigt einen Pfeil, bei dem die Länge der Pfeilspitze 40% der gesamten (Pfeil-)Länge beträgt.
  In Abbildung 10 ist ein Pfeil mit 100% Pfeilspitzenlänge zu sehen. Weil die Pfeilspitze 100% der Pfeillänge abdeckt, beginnen die Linien für die Schenkel auf der gleichen Höhe wie die Mittellinie (Schaft). Das heisst aber nicht, dass die Schenkel gleich lang sind, wie die Mittellinie! Weil sie auf der exakt gleichen Höhe starten müssen, müssen sie länger sein

als die Mittellinie. Die Pfeilspitze kann als gleichschenkliges Dreieck berechnet werden. Um die Länge der Schenkel zu berechnen muss eine Formel aus der Trigonometrie angewendet werden.

Folgendes ist zu beachten:

- Der Pfeil soll in die Richtung gezeichnet werden, in die die Schildkröte bei Programmaufruf schaut.
- Die Schildkröte muss am Schluss wieder genau am Ausgangspunkt stehen und in die ursprüngliche Richtung schauen.
- Die Pfeilspitze hat insgesamt einen **Öffnungswinkel** von 40°. Die Winkel zwischen den einzelnen Linien betragen also 20°.

Kommando 1.2 *Der Aufruf, um in XLogo den* ***Cosinus*** *eines Winkels von zum Beispiel* 45° *zu berechnen lautet: "****cos 45****".*
*Ein Cosinus-Ausdruck muss in Klammer geschrieben werden oder am Schluss eines Termes stehen. Anstelle von beispielsweise ":x / cos 20 * 17 / 47" muss also ":x * 17 / 47 / cos 20" oder ":x / (cos 20) * 17 / 47" geschrieben werden.*

Abbildung 8: Ein mit dem gesuchten Programm gezeichneter Pfeil.

1.6 Uhrzeit zeichnen

Lehrerhinweis 1.7 *Das gesuchte Programm baut auf dem Unterprogramm um Pfeile zu zeichnen auf. Der Auftrag ist verhältnismässig einfach.*

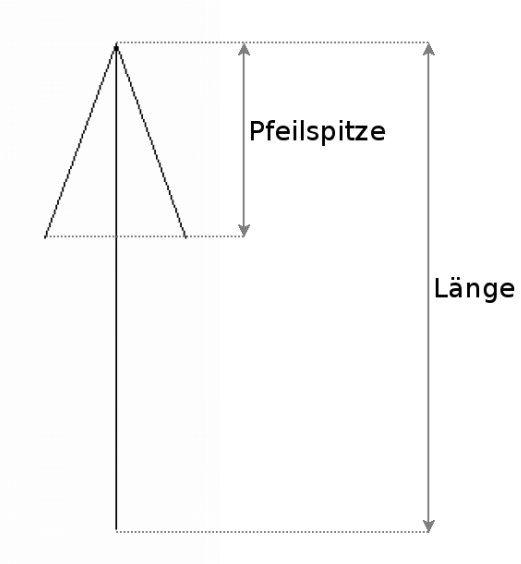

Abbildung 9: Ein Pfeil mit einer Länge der Pfeilspitze von 40% der Pfeillänge.

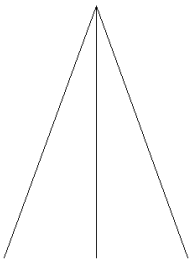

Abbildung 10: Ein Pfeil mit 100% Pfeilspitzenlänge.

Nun soll das Unterprogramm um Pfeile zu zeichnen verwendet werden, um eine Uhrzeit auf dem Bildschirm darzustellen:

- Die beiden Parameter enthalten zwei Zahlen für Stunden und Minuten.
- Als Ausgabe sollen die Stellungen des Minutenzeigers und des Stundenzeigers einer analogen Uhr zu sehen sein.
- Der Stundenzeiger soll eine Länge von 150 Pixeln haben und die Pfeilspitze soll halb solang sein, wie der gesamte Pfeil.
- Der Minutenzeiger ist 200 Pixel lang und die Pfeilspitze macht 10% des Pfeiles aus.
- Auch der Stundenzeiger geht jede Minute ein wenig vorwärts!

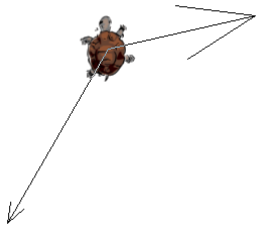

Abbildung 11: Ausgabe des Uhr-Programmes für 14 Uhr und 35 Minuten.

1.7 Wurzel ausprobieren

Lehrerhinweis 1.8 *Die Aufgabe hat drei Ziele: Sie führt in das Arbeiten mit XLogo und Fliesskommazahlen ein. Sie stellt eine Motivation für die nachfolgende Aufgabe dar. Und sie verlangt ein rekursives Programm als Lösung. Die Rekursion muss also für die verbleibenden Aufgaben bekannt sein! Ob die nachfolgenden Aufgaben noch Sinn machen mit der Klasse zu bearbeiten, hängt vom Niveau der Schüler ab. Es liegt im Ermessen der Lehrperson, die nachfolgenden Aufgaben wegzulassen.*

Es soll ein Programm geschrieben werden, dass eine Quadratwurzel auf **eine Stelle nach dem Komma** annähert. Der ganze Vorgang soll mit einem Liniendiagramm wie in Abbildung 12 illustriert werden. Die Idee ist, dass man bei 0 anfängt und solange 0.1 hinzuzählt, bis das Quadrat der Summe grösser oder gleich der gegebenen Zahl ist. Ist dieser Punkt erreicht, soll die erhaltene Zahl mittels dem print-Befehl ausgegeben werden. Beispielsweise für die Wurzel von 4 probiert das Programm folgende Zahlen: 0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9, 2.0. Jede dieser Zahlen wird mit sich selber multipliziert (quadriert) und mit 4 verglichen.
Zur Lösung der Aufgabe muss ein **Hilfsunterprogramm** geschrieben werden. Die Idee ist, dass sich dieses Hilfsunterprogramm selber solange aufruft, bis die gewünschte Toleranz erreicht ist.
Das Hilfsprogramm verwendet zwei Parameter:

- Die gegebene Zahl n, deren Quadratwurzel gesucht ist.
- Den aktuellen Versuch m um die Wurzel zu berechnen.
  Dies ist die Zahl zu der solange 0.1 hinzugezählt wird, bis deren Quadrat grösser oder gleich dem zweiten Parameter ist.

Schritt für Schritt sieht das Vorgehen etwa so aus:

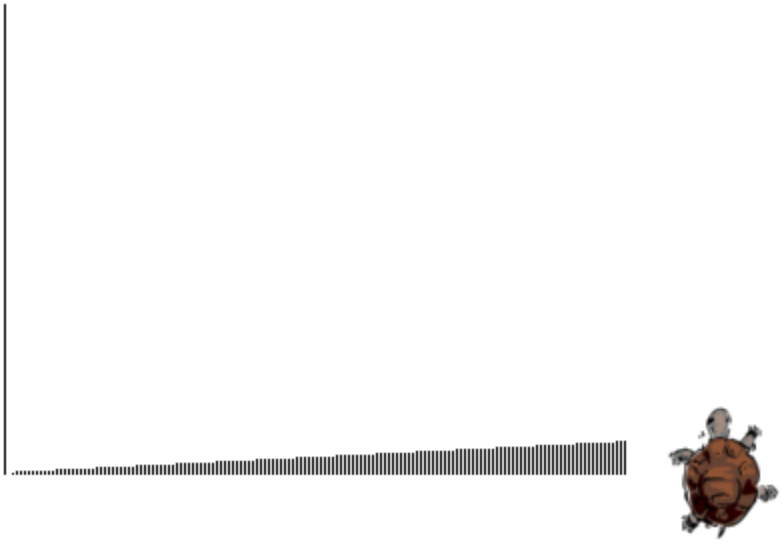

Abbildung 12: Schrittweises Annähern einer Wurzel.

1. Das Hauptprogramm zeichnet als erstes eine Linie der gegebenen Länge n.

2. Das Hauptprogramm ruft dann das Hilfsprogramm mit der gegebenen Zahl n und 0 für den Startwert m auf.

3. Das Hilfsprogramm zeichnet die geforderte Linie der Länge m für das Diagramm.

4. Dann prüft das Hilfsprogramm, ob $m \cdot m > n$ oder $m \cdot m = n$ ist.

5. Wenn ja, haben wir das Ziel erreicht und die Zahl wird mittels print ausgegeben.

6. Wenn nein, startet sich das Hilfsprogramm selber wieder. Die gegebene Zahl n wird dabei unverändert übergeben. Für die Wurzel m wird $m + 0.1$ übergeben. So wird m von Aufruf zu Aufruf hochgezählt, bis wir das Ziel erreicht haben.

Für jeden Annäherungsschritt wird eine Linie in der Länge der aktuellen Summe gezeichnet. So ist zu sehen, wie sich der Computer langsam an das Resultat hintastet. Die letzte Linie entspricht dem Resultat, welches mittels print ausgegeben wird. Zwischen den Linien soll wie in Abbildung 12 jeweils ein Spalt von einem Pixel Breite bleiben.

Kommando 1.3 *Neben den üblichen Befehlen, werden die folgenden zwei Kommandos in der Musterlösung verwendet:*

- ***print*** *"Ausgabe"*
  Unter der Zeichnungsfläche in XLogo werden alle gemachten Befehle aufgelistet. Mittels dem print-Befehl kann aus einem Logo-Programm heraus etwas in diese Liste geschrieben werden. Wir verwenden print um unser Resultat für die Quadratwurzel auszugeben. Das Kommando print nimmt Zahlen wie auch Texte als Parameter. Texte müssen in XLogo mit einem doppelten Hochkomma angefangen werden.

- ***bk****: "Backward", Rückwärts gehen*
  Das Kommando bk funktioniert genau gleich wie fd, lediglich geht die Schildkröte rückwärts anstelle von vorwärts. Der Befehl kann verwendet werden, wenn es praktisch ist, rückwärts zu gehen, ohne sich umzudrehen.

1.8 Babylonisches Wurzelziehen - Heronverfahren

Lehrerhinweis 1.9 *Das hier gefragte Programm kann mit nur einer kleinen Änderung von der vorhergehenden Aufgabe übernommen werden. Didaktisch macht es Sinn, das Heronverfahren im Unterricht zu erklären.*

Wenn mit dem vorhergehenden Programm die Wurzel aus 10′000 berechnet werden soll, benötigt die Schildkröte soviele Schritte, dass sie rechts aus dem Bildschirm fährt. Fragen wir uns nun, wie wir das Verfahren verbessern können, damit wir weniger Schritte benötigen.
Ein sehr gutes Verfahren ist das babylonische Wurzelziehen, auch Heronverfahren genannt. Das Verfahren hat folgende Formel zur Grundlage:

$$x_{n+1} = \frac{x_n + \frac{a}{x_n}}{2} \tag{8}$$

- x_n stellt die als letztes berechnete Näherung dar.
- x_{n+1} ist die nächste Näherung.
- a ist die gegebene Zahl, deren Quadratwurzel gesucht ist.

Es ist nun das Programm aus der vorhergehenden Aufgabe abzuändern. Anstelle des Addierens von 0.1 soll die Formel von Heron eingesetzt werden. Weil nicht mehr sicher ist, ob wir uns von unten oder von oben an die Wurzel annähern, muss der Test angepasst werden. Als eine einfache Variante schaut man ob das Quadrat der berechneten Zahl +/-0.1 der gegebenen Zahl entspricht.
Fährt die Schildkröte nun immer noch rechts aus dem Bild?

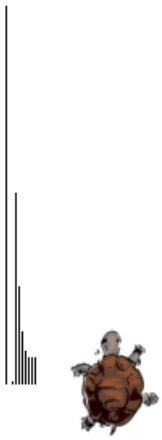

Abbildung 13: Annähern einer Wurzel mit dem Heronverfahren.

1.9 Kreuz aus Kreuzchen

Lehrerhinweis 1.10 *Diese (letzte) Aufgabe soll wieder als Puffer für die schnellen Schüler dienen. Mathematisch kommt hier nichts Neues hinzu. Es erfordert jedoch einiges an Überlegungen, um zu erkennen, dass die Distanz der Kreuze exponentiell ansteigt. Wiederum stellt die Aufgabe eine Anwendung der rekursiven Programmierung dar. Fraktale werden oft als faszinierend empfunden.*

Es soll ein **rekursives** Programm geschrieben werden, das aus vielen kleinen Kreuzchen ein Kreuz zeichnet und daraus wieder ein Kreuz usw. Die Verschachtelungstiefe wird dem Programm beim Aufruf mitgegeben. In Abbildung 14 ist der verwendete Grundbaustein zu sehen. Es handelt sich um

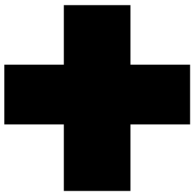

Abbildung 14: Ein Kreuz aus nur 5 Pixeln.

ein Kreuz bestehend aus 5 Pixeln. Ohne Vergrösserung soll dieses Kreuz nur 3 Pixel breit und 3 Pixel hoch sein.
Jetzt wollen wir ein Kreuz zeichnen, welches anstelle von 5 Pixeln aus 5 kleineren Kreuzen besteht. Ein Kreuz aus fünf Kreuzen sehen wir in Abbildung 15. Nun wollen wir das Spiel fortführen und ein Kreuz aus fünf Kreuzen, die

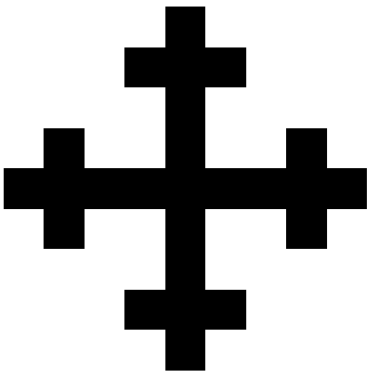

Abbildung 15: Ein Kreuz aus fünf Kreuzen.

wiederum aus fünf Kreuzen bestehen zeichnen. Ein entsprechendes Bild ist in Abbildung 16 zu sehen. Schlussendlich soll eine Zeichnung wie in Abbildung

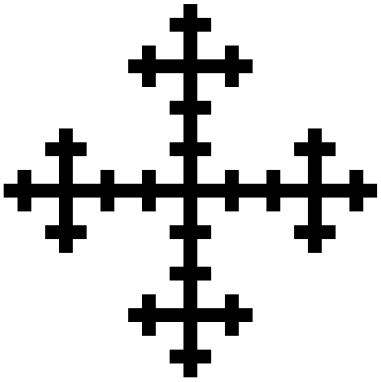

Abbildung 16: Der Anfang eines Fraktals.

17 entstehen. Die Aufgabe ist nun klar: Es ist ein Programm zu schreiben, das solche **Fraktale** zeichnet. Als Parameter wird dem Programm mitgegeben wieviele Verschachtelungen gewünscht sind. Mit 0 als Parameter soll eine Grafik wie in Abbildung 14 resultieren. Wenn 1 mitgegeben wird, sollten wir Abbildung 15 erhalten und mit 2 wird Abbildung 16 gezeichnet.

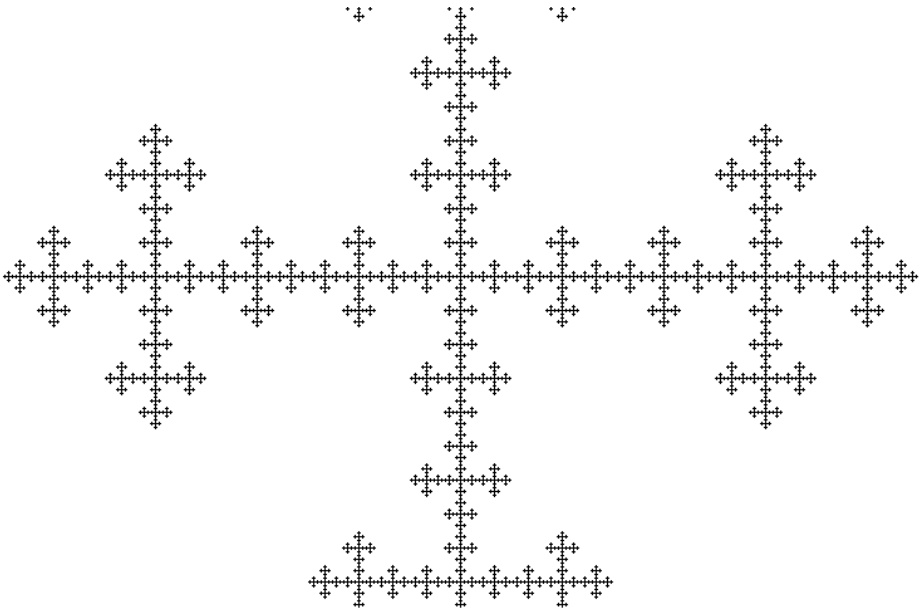

Abbildung 17: Ein Fraktal mit Kreuzen.

Benötigt wird ein Programm, dem die verbleibende Verschachtelungstiefe n als Parameter mitgegeben wird. Das Programm zeichnet auf **jeden** Fall ein Kreuz. Wenn die Verschachtelungstiefe 0 erreicht, zeichnet das Programm ein Kreuz aus fünf Pixeln wie in Abbildung 14. Wenn die Verschachtelungstiefe n noch nicht auf 0 ist, verwendet das Hilfsprogramm anstelle von Pixeln Kreuze der Verschachtelungstiefe $n - 1$.
Es empfiehlt sich Kreuze immer von der Mitte aus zu zeichnen und nach dem Zeichnen eines Kreuzes immer wieder in der genau gleichen Stellung wie beim Start anzukommen.
Eine weitere Stolperfalle ist die Seitenlänge der Kreuze in Abhängigkeit von n. Bei $n = 0$ misst ein Kreuz 3 Pixel. Für $n = 1$ sind es 9 Pixel. Man tut gut daran sich zuerst eine Formel für die Kreuzgrösse in Abhängigkeit von n zu überlegen. Damit ist dann auch klar wie weit auseinander die Kreuze liegen müssen um selber ein Kreuz zu formen.

Kommando 1.4 ***power b e***
Die Funktion power nimmt zwei Parameter: Basis b und Exponent e und berechnet die Potenz aus den beiden.

2 Lösungen

2.1 Geschwindigkeit berechnen

Es kann z.B. folgendes Programm verwendet werden:

```
to speed :w :n
 cs
 repeat :n [fd :w rt 90 fd 1 rt 90 fd :w lt 90 fd 1 lt 90]
end
```

2.2 Polygon zeichnen

Die Berechung des benötigten Winkels ist einfacher, als man zuerst vermuten würde:

```
to neck :n :l
 repeat :n [fd :l rt (360/:n)]
end
```

2.3 Haus bauen

Hier die geforderte einfache Variante:

```
to haus :h :b :s
 rt 90
 repeat :s [
  repeat 2 [
   fd :b
   lt 90
   fd ((:h - :b / 2) / :s)
   lt 90
  ]
  lt 90
  fd ((:h - :b / 2) / :s)
  rt 90
 ]
 lt 45
 fd (:b / (sqrt 2))
 rt 90
 fd (:b / (sqrt 2))
 pu
 fd :b
end
```

2.4 Haus verschönern

Dies ist der Code für die Luxushäuser:

```
to haus2 :h :b :s
 rt 90
 fd (:b /3)
 lt 90
 fd ((:h - :b / 2) / :s / 5 * 4)
 lt 90
 fd (:b / 6)
 lt 90
 fd ((:h - :b / 2) / :s / 5 * 4 / 2)
 lt 90
 fd (:b / 20 + 1)
 bk (:b / 20 + 1)
 rt 90
 fd ((:h - :b / 2) / :s / 5 * 4 / 2)
 rt 90
 fd (:b / 6)
 rt 180
 repeat :s [
  pu
  fd (:b / 4 * 3)
  lt 90
  fd ((:h - :b / 2) / :s / 4)
  pd
  fd ((:h - :b / 2) / :s / 2)
  lt 90
  fd (:b / 6)
  lt 90
  fd ((:h - :b / 2) / :s / 2)
  lt 90
  fd (:b / 12)
  lt 90
  fd ((:h - :b / 2) / :s / 2)
  bk ((:h - :b / 2) / :s / 2)
  rt 90
  fd (:b / 12)
  rt 90
  pu
  fd ((:h - :b / 2) / :s / 4)
  rt 90
  fd (:b / 4 * 3)
  rt 180
  pd
  repeat 2 [
   fd :b
   lt 90
   fd ((:h - :b / 2) / :s)
```

```
    lt 90
  ]
  lt 90
  fd ((:h - :b / 2) / :s)
  rt 90
]
lt 45
bk (:b / (sqrt 2) / 3)
fd (:b / (sqrt 2) * 4 / 3)
rt 135
pu
fd (:b/4)
rt 90
pd
repeat 36 [rt 10 fd (:b/100)]
pu
rt 90
fd (:b/4)
pd
rt 135
fd (:b / (sqrt 2) * 4 / 3)
pu
fd :b
end
```

2.5 Pfeile zeichnen

```
to pfeil :a :b
  fd :a
  rt 160
  fd :a * :b / 100 / cos 20
  rt 180
  fd :a * :b / 100 / cos 20
  lt 140
  fd :a * :b / 100 / cos 20
  rt 180
  fd :a * :b / 100 / cos 20
  rt 160
  fd :a
  rt 180
end
```

2.6 Uhrzeit zeichnen

```
to uhr :s :m
  rt :s * 360 / 12 + :m * 360 / 60 / 12
  pfeil 150 50
```

```
4   lt :s * 360 / 12 + :m * 360 / 60 / 12
5   rt :m * 360 / 60
6   pfeil 200 10
7   lt :m * 360 / 60
8 end
```

2.7 Wurzel ausprobieren

```
1 to wurzel :n
2   cs
3   pu
4   lt 90
5   fd 200
6   lt 90
7   fd 100
8   lt 180
9   pd
10  fd :n
11  bk :n
12  pu
13  rt 90
14  fd 2
15  lt 90
16  pd
17  wh 0 :n
18 end
19
20 to wh :m :n
21  pu
22  rt 90
23  fd 2
24  lt 90
25  pd
26  fd :m
27  bk :m
28  if (:m * :m > :n | :m * :m = :n)
29    [pu rt 90 fd 50 lt 90 print :m]
30    [wh (:m+0.1) :n]
31 end
```

2.8 Babylonisches Wurzelziehen - Heronverfahren

```
1 to wurzel2 :n
2   cs
3   pu
4   lt 90
5   fd 200
6   lt 90
7   fd 100
```

```
8    lt 180
9    pd
10   fd :n
11   bk :n
12   pu
13   rt 90
14   fd 2
15   lt 90
16   pd
17   wh2 1 :n
18 end
19
20 to wh2 :m :n
21   pu
22   rt 90
23   fd 2
24   lt 90
25   pd
26   fd :m
27   bk :m
28   if ((:m*:m > :n-0.1) & (:m*:m < :n+0.1))
29     [pu rt 90 fd 50 lt 90 print :m]
30     [wh2 (:m+:n/:m)/2 :n]
31 end
```

2.9 Kreuz aus Kreuzchen

```
1 to kreuz :n
2   if :n > 0 [
3     kreuz (:n-1)
4     pu
5     fd (power 3 :n)
6     kreuz (:n-1)
7     rt 180
8     pu
9     fd 2*(power 3 :n)
10    kreuz (:n-1)
11    rt 180
12    pu
13    fd (power 3 :n)
14    rt 90
15    pu
16    fd (power 3 :n)
17    kreuz (:n-1)
18    rt 180
19    pu
20    fd 2*(power 3 :n)
21    kreuz (:n-1)
22    rt 180
```

```
23      pu
24      fd (power 3 :n)
25      rt 270
26    ]
27    [
28      pd
29      fd 1
30      bk 2
31      fd 1
32      rt 90
33      fd 1
34      bk 2
35      fd 1
36      lt 90
37    ]
38  end
```