Logo-Aufgaben mit Verbindung zur Mathematik

Student: Matthias Bolli
Dozent: Prof. Juraj Hromkovic
Datum: 13.06.2007

Logo-Kenntnisse

Für die Lösung der Aufgaben werden folgende Logo-Befehle benötigt:

Arithmetik:	`+, -, *, /, ()`
Grafik:	`left, right, forward, setxy, window, ycor`
Definitionen:	`to...end, make`
Kontrollstrukturen:	`repeat, if`
Mathematik:	`sqrt, sin, cos, arctan`
Zufallszahlen:	`random`
Ausgabe:	`print`

Die Lösungen wurden mit Berkeley Logo 5.5 getestet (http://www.cs.berkeley.edu/~bh/).

Mathematik-Kenntnisse

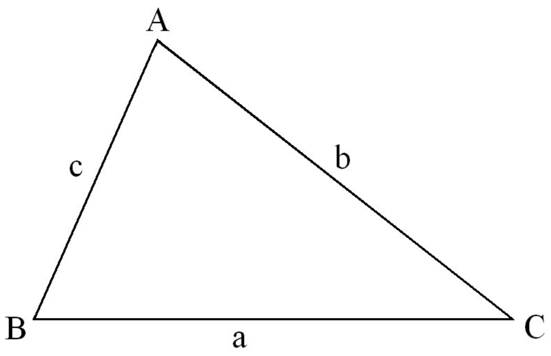
Die benötigten Mathematik-Kenntnisse sind unterschiedlich. Sie werden bei jeder Aufgabe separat angegeben.

Inhalt

Aufgaben

(A1) Dreieck aus drei gegebenen Seiten

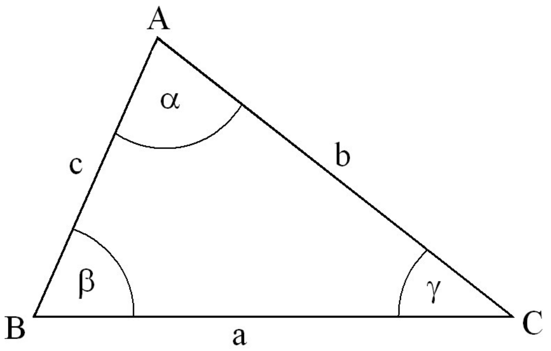
Erstellen Sie ein Programm, welches aus drei gegebenen Seitenlängen a, b und c ein Dreieck zeichnet. Dabei soll die Seite a waagrecht und die Ecke B in den Nullpunkt zu liegen kommen.



(a)

Themen Mathematik: Trigonometrie, Cosinussatz

Lösen Sie die Aufgabe, indem Sie die Winkel des Dreiecks berechnen und das Dreieck mit den klassischen Logo-Zeichenbefehlen `forward` und `left` bzw. `right` zeichnen.



Tipp: Berkeley Logo bietet keine Funktion `arccos` an. Sie kann wie folgt nachgebildet werden:

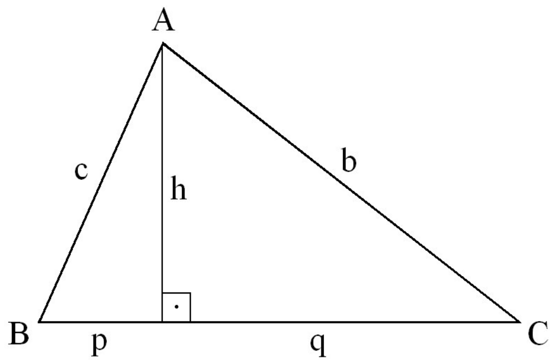
```
to arccos :x
output arctan((sqrt(1 - :x*:x)) / :x)
end
```

(b)

Themen Mathematik: Pythagoras, Gleichungssysteme

Lösen Sie die Aufgabe ohne Trigonometrie und ohne Verwendung von `forward` und `left` bzw. `right`. Berechnen Sie stattdessen die Koordinaten der Eckpunkte und verwenden Sie den Logo-Befehl `setxy`, um eine Linie zu einem bestimmten Punkt zu ziehen.

Tipp: Berechnen Sie die Höhe h auf die Seite a sowie die beiden Höhenabschnitte p und q.



(A2) Kreis mit gegebenem Radius

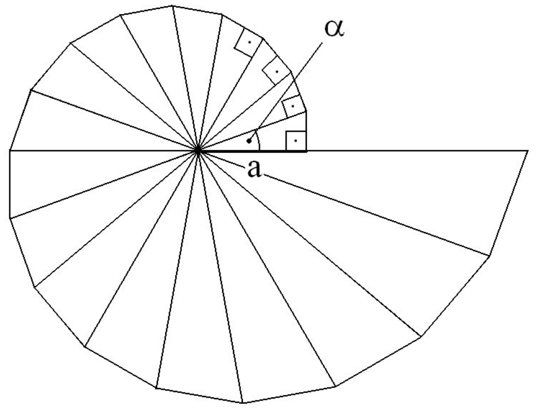
Themen Mathematik: Trigonometrie, regelmässige Vielecke

Erstellen Sie ein Programm, welches einen Kreis mit gegebenem Radius näherungsweise zeichnet. Der Kreis soll durch ein einbeschriebenes regelmässiges 100-Eck angenähert werden.

(A3) Schnecke

Themen Mathematik: Trigonometrie, rechtwinkliges Dreieck

Erstellen Sie ein Programm, welches aus einem Winkel α und einer Startlänge a folgende Schnecke zeichnet:



(A4) Bodenplatten

Themen Mathematik: ggT

Erstellen Sie ein Programm, welches einen Platz mit gegebener Länge und Breite mit möglichst grossen quadratischen Platten belegt. Das Programm soll ausserdem die Seitenlänge der Platten herausschreiben.

Tipp: Sie können dazu die folgende Logo-Prozedur verwenden, welche mit dem Euklidischen Algorithmus den grössten gemeinsamen Teiler (ggT) zweier Zahlen bestimmt:

```
to ggt :a :b
if :b > :a [ output ggt :b :a ]
if :b = 0 [ output :a ]
output ggt :b (:a - :b)
end
```

(A5) Zufallswanderer

Themen Mathematik: Wahrscheinlichkeitsrechnung, Trigonometrie, Integralrechnung

Erstellen Sie ein Programm, welches eine gegebene Anzahl Schritte ausführt. Jeder Schritt ist 50 Einheiten lang und geht in eine zufällige Richtung. Das Programm soll dabei zählen, wie viele Schritte in y-Richtung eine Hunderter-Grenze überschreiten. Am Schluss soll das Programm den Quotienten

$$\frac{\text{Anzahl Schritte insgesamt}}{\text{Anzahl Schritte, welche eine Hunderter-Grenze überschritten haben}}$$

berechnen und ausgeben.

Lassen Sie das Programm mit verschiedenen (auch grossen) Schrittzahlen laufen. Was fällt Ihnen auf? Können Sie Ihre bemerkenswerte Beobachtung erklären?

Lösungen

(L1) Dreieck aus drei gegebenen Seiten

(a)

Der Cosinussatz für den Winkel γ lautet:

$$c^2 = a^2 + b^2 - 2ab \cdot \cos\gamma$$

Nach γ auflösen ergibt:

$$\gamma = \arccos\left(\frac{a^2 + b^2 - c^2}{2ab}\right)$$

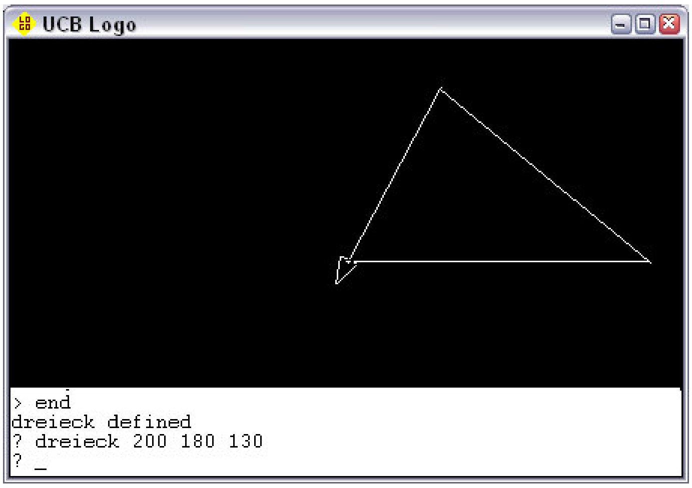
Analog:

$$\alpha = \arccos\left(\frac{b^2 + c^2 - a^2}{2bc}\right)$$

Damit haben wir schon alles, was wir brauchen.

```
to arccos :x
output arctan((sqrt(1 - :x*:x)) / :x)
end

to dreieck :a :b :c
make "gamma arccos((:a*:a + :b*:b - :c*:c)/(2*:a*:b))
make "alpha arccos((:b*:b + :c*:c - :a*:a)/(2*:b*:c))
right 90
forward :a
left (180 - :gamma)
forward :b
left (180 - :alpha)
forward :c
end
```



(b)

Wir berechnen die Höhe h sowie die Höhenabschnitte p und q. Daraus ergeben sich nachher direkt die Koordinaten der Eckpunkte: A(p,h), B(0,0) und C(a,0).

Für die Unbekannten h, p und q können wir folgendes Gleichungssystem aufstellen:

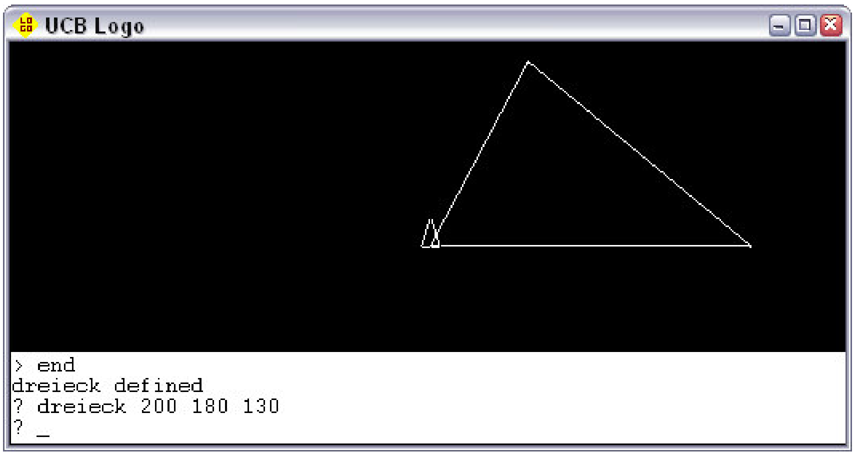
$$p^2 + h^2 = c^2$$
$$q^2 + h^2 = b^2$$
$$p + q = a$$

Wir erhalten nach einiger Rechnung:

$$p = \frac{a^2 - b^2 + c^2}{2a}, \quad q = \frac{a^2 + b^2 - c^2}{2a} \quad \text{und} \quad h = \sqrt{c^2 - p^2}$$

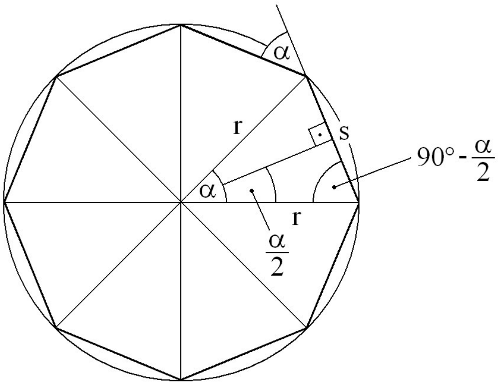
Beachten Sie, dass obige Formel für h zugunsten einer einfacheren Darstellung immer noch die Unbekannte p enthält. Für die Umsetzung in einem Programm stellt dies kein Problem dar. Wir müssen einfach zuerst p berechnen und in einer Variablen speichern. Die Grösse q wird für die Zeichnung gar nicht gebraucht.

```
to dreieck :a :b :c
make "p (:a*:a - :b*:b + :c*:c)/(2*:a)
make "h sqrt (:c*:c - :p*:p)
setxy :a 0
setxy :p :h
setxy 0 0
end
```



(L2) Kreis mit gegebenem Radius

Für die Herleitung der Formel betrachten wir ein regelmässiges n-Eck statt eines 100-Ecks. Für die Skizze nehmen wir n = 8:



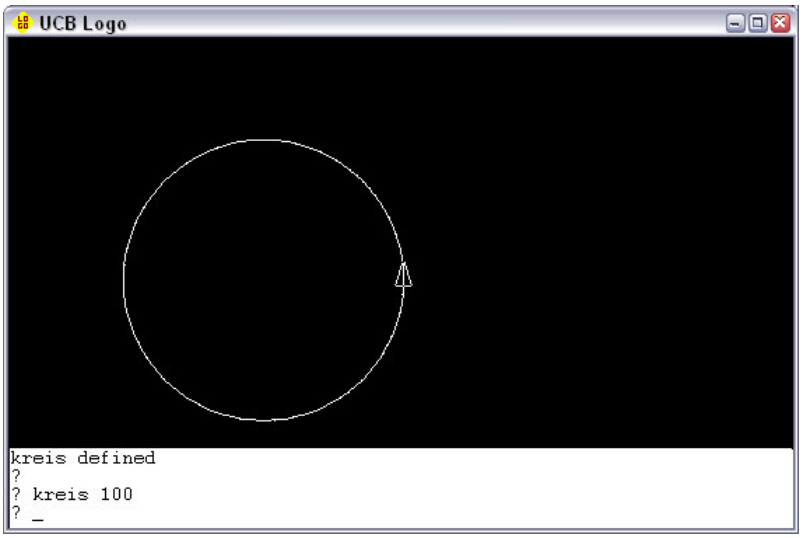
Der Winkel α beträgt offenbar 360°/n. Für die Seite s finden wir:

$$\sin\frac{\alpha}{2} = \frac{s}{2r}\text{, also}$$

$$s = 2r\sin\frac{\alpha}{2}$$

Das regelmässige n-Eck können wir zeichnen, indem wir zuerst um α/2 nach links drehen und dann die n Seiten zeichnen, jede Seite gefolgt von einer Linksdrehung um α.

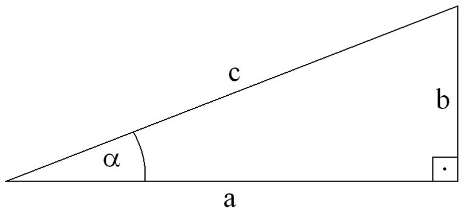
```
to kreis :r
make "n 100
make "alpha 360 / :n
make "s 2 * :r * sin (:alpha / 2)
left (:alpha / 2)
repeat :n [
  forward :s
  left :alpha
]
end
```



(L3) Schnecke

Die Schnecke besteht aus einer Anzahl ähnlicher rechtwinkliger Dreiecke. Alle Dreiecke haben das Zentrum als gemeinsamen Eckpunkt. Die dem Zentrum anliegende Kathete des ersten Dreiecks hat die Länge a. Die Hypotenuse des ersten Dreiecks wird zur (dem Zentrum anliegenden) Kathete des zweiten Dreiecks, die Hypotenuse des zweiten wird zur (dem Zentrum anliegenden) Katheten des dritten usw.

Wir müssen also im wesentlichen die Hypotenuse c des Dreiecks aus dem Winkel α und der Ankathete a berechnen.



Es gilt:

$$\cos\alpha = \frac{a}{c}$$

also:

$$c = \frac{a}{\cos\alpha}$$

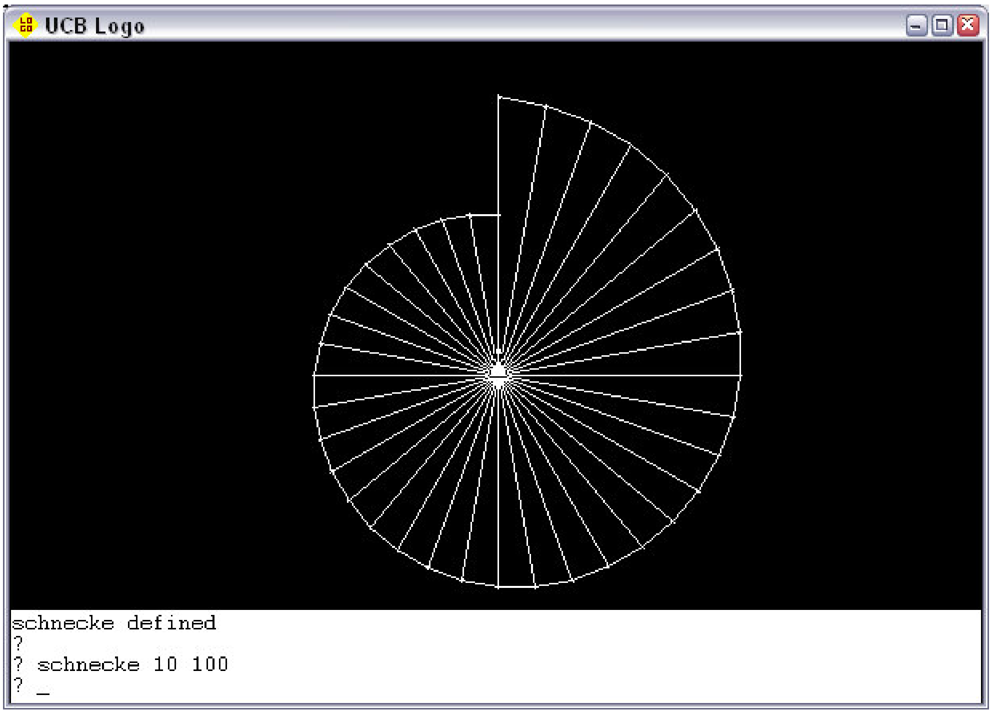
Die Seite b ergibt sich zu:

$b = a \cdot \tan\alpha$

Die Anzahl der Dreiecke für eine Umdrehung beträgt $n = 360°/\alpha$. Das geht natürlich nur auf, wenn α ein Teiler von 360° ist.

Das Programm besteht also aus einer Schleife, die n mal durchlaufen wird. In jedem Durchgang wird ein Dreieck gezeichnet. Dabei muss sorgfältig beachtet werde, dass am Ende eines Durchlaufs die Bedingungen für den nächsten Durchlauf richtig gesetzt werden: Der Zeichenstift muss in der Mitte stehen, die Richtung muss stimmen und die Grösse des nächsten Dreiecks muss berechnet und in einer Variablen abgelegt sein.

```
to schnecke :alpha :a
make "n 360 / :alpha
make "beta 90 - :alpha
repeat :n [
  make "b :a * sin :alpha / cos :alpha
  make "c :a / cos :alpha
  forward :a
  left 90
  forward :b
  left 180 - :beta
  forward :c
  left 180
  ; die Hypotenuse c wird zur Kathete a des nächsten Dreiecks
  make "a :c
]
end
```

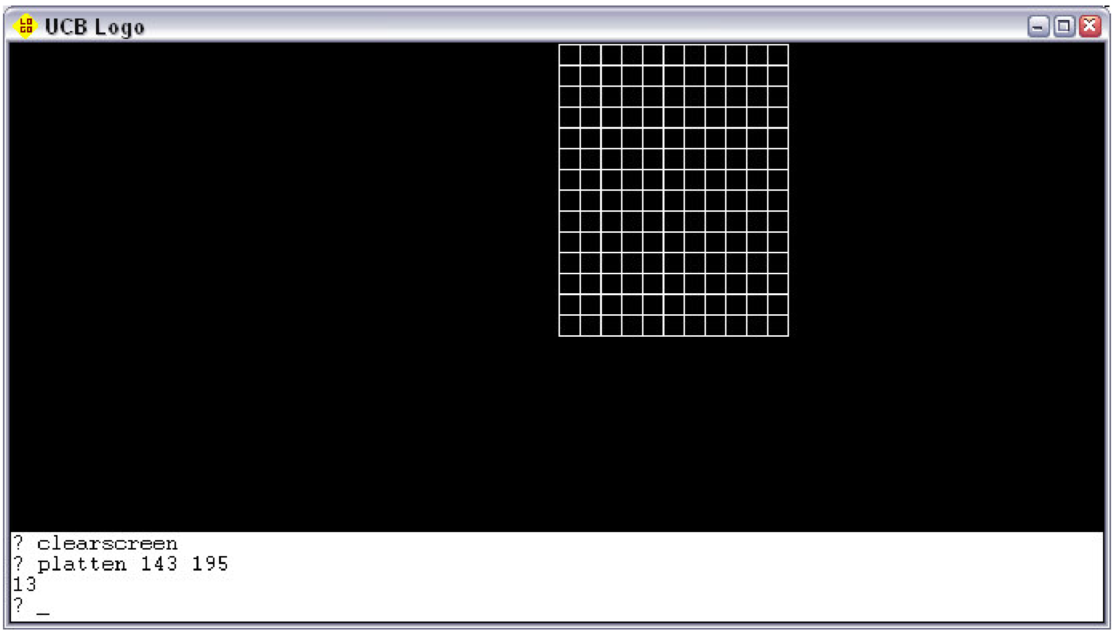


(L4) Bodenplatten

Die optimale Plattengrösse entspricht dem ggT von Länge und Breite des Platzes, für dessen Berechnung eine Prozedur vorgegeben ist. Es geht also nur noch darum, die Anzahl Platten in der Länge und in der Breite zu berechnen und das entsprechende Raster zu zeichnen.

```
to ggt :a :b
if :b > :a [ output ggt :b :a ]
if :b = 0 [ output :a ]
output ggt :b (:a - :b)
end

to platten :a :b
make "s ggt :a :b
make "na :a / :s
make "nb :b / :s
forward :b
right 180
forward :b
left 90
repeat :na [
  forward :s  left 90  forward :b  right 180  forward :b  left 90
]
right 180
forward :a
right 90
repeat :nb [
  forward :s  right 90  forward :a  right 180  forward :a  right 90
]
print :s
end
```



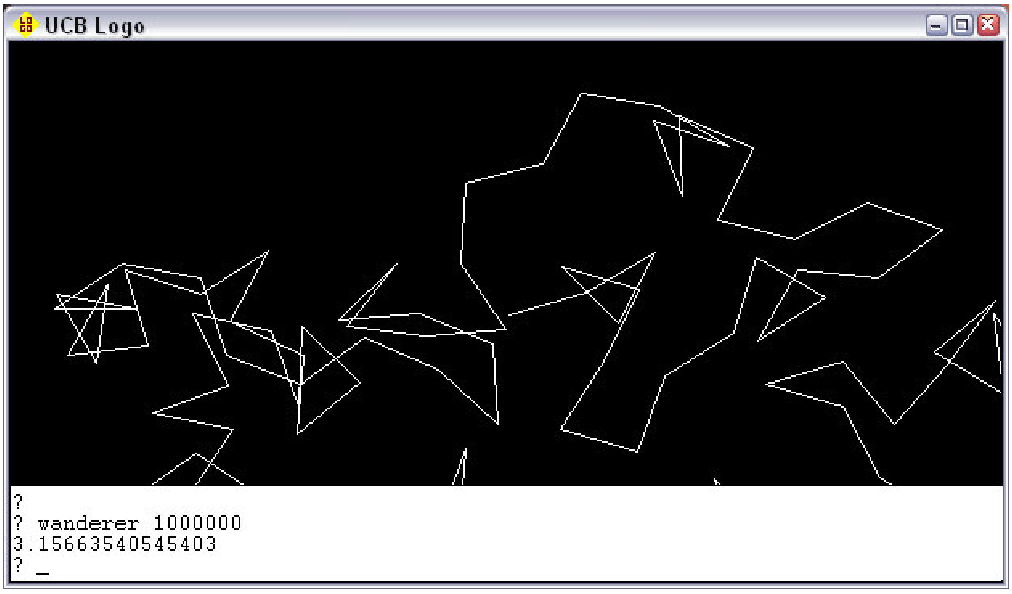
(L5) Zufallswanderer

Zuerst zur Programmierung: Die Marschrichtung soll zufällig sein. Eine ganzzahlige Zufallszahl im Bereich 0 bis 359 bekommt man mit

```
random 360
```

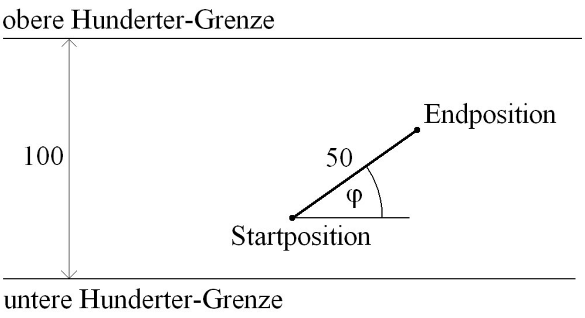
Um zu entscheiden, ob in y-Richtung eine Hunderter-Grenze überschritten wurde, bestimmt man das aktuelle Hunderter-Band vor und nach der Bewegung, indem man jeweils die aktuelle y-Koordinate durch 100 dividiert und auf Ganze abrundet. Ungleichheit der Hunderter-Bänder vor und nach der Bewegung bedeutet eine Grenzüberschreitung.

```
to wanderer :n
make "zaehler 0
make "ybandvorher int(ycor / 100)
; unendlich grosse Zeichenfläche einstellen:
window
repeat :n [
  make "winkel random 360
  left :winkel
  forward 50
  make "ybandnachher int (ycor / 100)
  if :ybandvorher <> :ybandnachher [ make "zaehler zaehler + 1 ]
  make "ybandvorher ybandnachher
]
print (:n / :zaehler)
end
```



Offenbar nähert sich der vom Programm berechnete Quotient mit zunehmender Anzahl Schritte der Zahl π an. Warum wohl?

Die Startposition liegt immer in einem bestimmten Hunderter-Band. In welchem, ist nicht relevant. Es geht nur darum, ob mit dem nächsten Schritt eine Grenze überschritten wird. Ebenfalls nicht relevant ist die x-Koordinate des Wanderers. Die Situation kann also vereinfacht so dargestellt werden:



Unter welchen Bedingungen wird nun mit einem Schritt eine Hunderter-Grenze überschritten? Wir unterscheiden zwei Fälle:

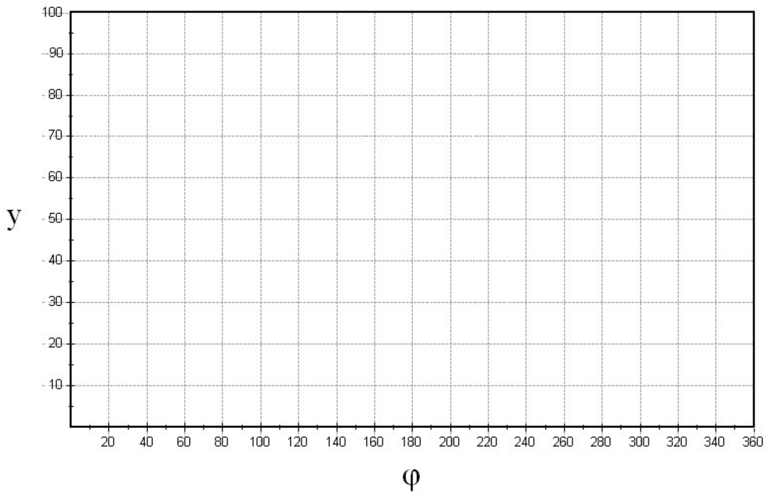
Eine Überschreitung oben kommt nur für Winkel von 0° bis 180° in Frage. Die Grenze wird überschritten, wenn

$y + 50 \sin \varphi > 100$

Eine Überschreitung unten ist nur bei Winkeln von 180° bis 360° möglich. Die Grenze wird überschritten, wenn

$y + 50 \sin \varphi < 0$

Ein Schritt kann als Punkt in folgendem Rechteck betrachtet werden:



In diesem Rechteck können wir nun die Fälle einzeichnen, welche zu einer Grenzüberschreitung führen.

Eine Grenzüberschreitung oben geschieht, wenn

$y + 50 \sin \varphi > 100$

Die Trennlinie zwischen Grenzüberschreitung ja bzw. nein ist

$y + 50 \sin \varphi = 100$

bzw. die Funktion

$y = 100 - 50 \sin \varphi$

Eine Grenzüberschreitung unten geschieht, wenn

$y + 50 \sin \varphi < 0$

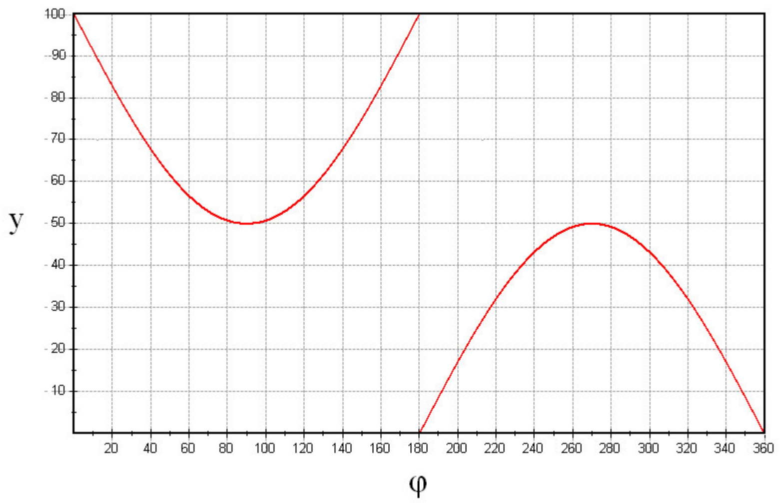
Hier ist die Trennlinie

$y + 50 \sin \varphi = 0$

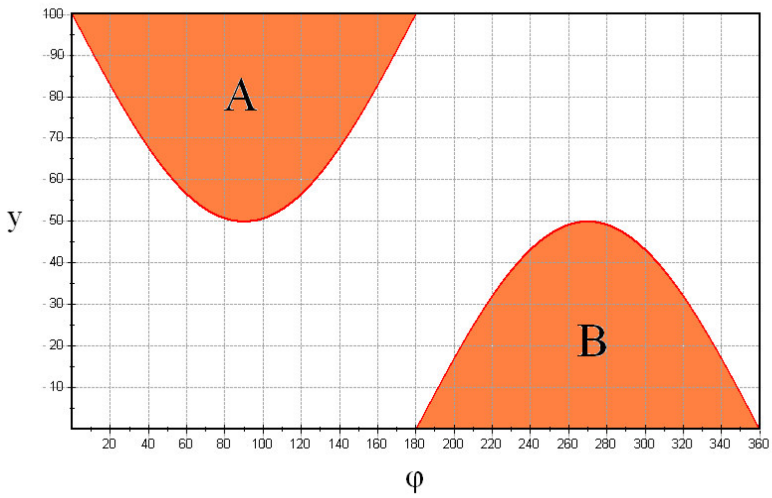
bzw. die Funktion

$y = -50 \sin \varphi$

Wir können diese Trennlinien als Funktionsgraphen in das Rechteck einzeichnen:



Offenbar entsprechen die eingefärbten Flächenteile A und B den Fällen, welche zu einer Grenzüberschreitung führen:



Die Bewegungsrichtung ist gleichverteilt zufällig im Intervall [0,360[. Die Startposition eines Schrittes (d.h. deren y-Koordinate innerhalb des aktuellen Hunderter-Bandes) kann ebenfalls als gleichverteilt zufällig angesehen werden, da sie durch zufällige Schritte entstanden ist (ohne weitere Begründung).

Daher entspricht die Wahrscheinlichkeit, dass ein Schritt (d.h. ein (φ,y)-Paar) in ein bestimmtes Gebiet fällt, dem Verhältnis der Fläche dieses Gebiets zu der Rechtecksfläche R.

Die Wahrscheinlichkeit P einer Grenzüberschreitung entspricht also dem Verhältnis der Summe der beiden eingefärbten Flächen zu der ganzen Rechtecksfläche R:

$$P = \frac{A+B}{R}$$

Es ist leicht zu sehen, dass

$R = 100 \cdot 360 = 36000$

und

$A = B.$

Für B erhalten wir:

$$B = \int_{180}^{360} -50 \sin\varphi \, d\varphi = -50 \int_{180}^{360} \sin\varphi \, d\varphi = -50 \left[-\frac{180}{\pi} \cos\varphi \right]_{180}^{360} = \frac{9000}{\pi} (\cos 360 - \cos 180) = \frac{18000}{\pi}$$

Zusammen:

$$P = \frac{A + B}{R} = \frac{2B}{R} = \frac{2 \cdot 18000}{36000\pi} = \frac{1}{\pi}$$

Der vom Programm berechnete Quotient

$$Q = \frac{\text{Anzahl Schritte insgesamt}}{\text{Anzahl Schritte, welche eine Hunderter-Grenze überschritten haben}}$$

strebt gemäss Definition der Wahrscheinlichkeit mit zunehmender Anzahl Schritte gegen den Reziprokwert von P:

$$Q \to \frac{1}{P} = \pi \qquad \text{(Anzahl Schritte } \to \infty \text{)}$$

Anmerkung: Der Zufallswanderer ist eine Variante von "Buffon's Needle Problem".