Fibbonacci Folge

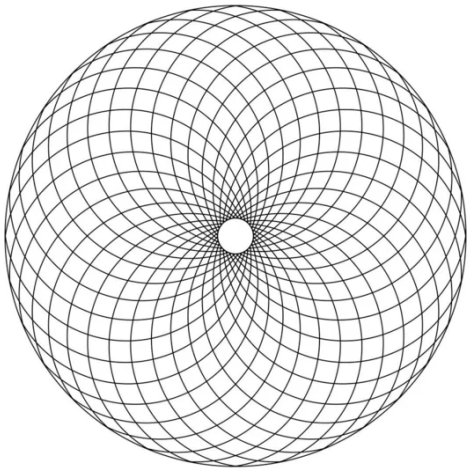
Die Fibonacci-Folge ist eine mathematische Folge von Zahlen, bei der jede Zahl die Summe der beiden vorhergehenden Zahlen ist. Die Folge beginnt normalerweise mit den Zahlen
0 und 1.

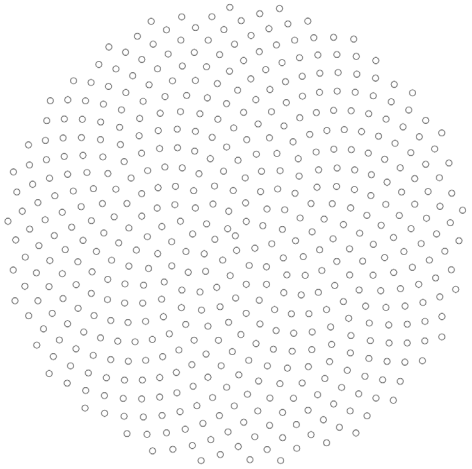
Die ersten Zahlen der Fibonacci-Folge sind:
0,1,1,2,3,5,8,13,21,34,...

0+1=1, 1+1=2, 1+2=3, 2+3=5, 3+5=8, 5+8=13...

Die Fibbonacci Folge kommt auch in der Natur vor, da sie eine grundlegende mathematische Struktur widerspiegelt, die in Wachstumsprozessen und geometrischen Anordnungen sichtbar wird. Einige Beispiele dafür sind die Anzahl der Spiralen in Tannenzapfen, Ananas und Sonnenblumen. Diese entsprechen oft aufeinanderfolgenden Fibonacci-Zahlen (z. B. 8 und 13 Spiralen).

Die Blätter an Pflanzenstängeln wachsen oft in einem Winkel (Phyllotaxis-Winkel), der dem goldenen Schnitt (ca. 137,5) entspricht, wodurch eine Fibonacci-Sequenz entsteht. Dies optimiert die Lichtaufnahme. Auch in Schneckenhäusern, Wirbelstürmen oder in Meereswellen kann man die Fibbonacci-Muster erkennen.



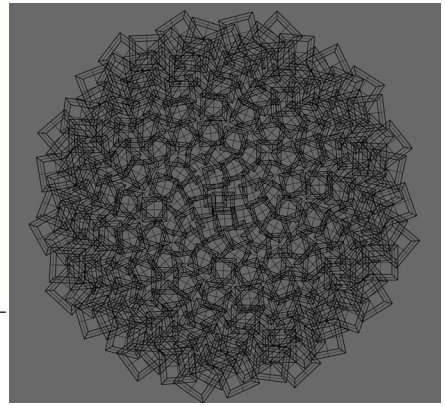


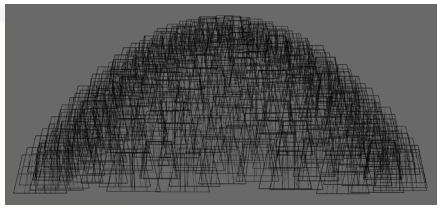
Ausgangs Code:

```
import rhinoscriptsyntax as rs
import math


s = 137.508
for n in range(0,500):
    t = math.sqrt(n)
    g = n * s
    z = rs.Polar([0,0,0],g,t)
    rs.AddCircle(z,0.3)
```

Quelle: https://www.designcoding.net/fermats-spiral-with-rhino-python/, aufgerufen am 22.01.2024





```
# -*- coding: utf-8 -*-
import rhinoscriptsyntax as rs
import math

# Alle Objekte in der Szene löschen
allobjs = rs.AllObjects()
if allobjs:
    rs.DeleteObjects(allobjs)

# Parameter
s = 137.508  # Phyllotaxis-Winkel (goldener Winkel in Grad)
num_points = 1000  # Anzahl der Punkte
base_cube_size = 2  # Größe der Würfel
z_distance = -0.05 # Z-Achse nach oben
scaling_factor = 0.1  # Faktor zur Beeinflussung der Größenänderung
pts_list=[]
for n in range(num_points):
    # Berechnung der Polar-Koordinaten
    angle = math.radians(n * s)  # Winkel in Bogenmaß
    distance = math.sqrt(n)  # Radialer Abstand

    # Kartesische Koordinaten berechnen
    x = distance * math.cos(angle)  # X-Koordinate
    y = distance * math.sin(angle)  # Y-Koordinate
    z = n * z_distance  # Z-Koordinate steigt proportional zu n


 # Würfelgröße basierend auf der Z-Koordinate anpassen
    # Je negativer z, desto größer wird der Würfel
    cube_size = base_cube_size + abs(z) * scaling_factor

    # Eckpunkte des Würfels definieren
    half_size = cube_size / 2
    third_size=cube_size/3
    corner1 = [x - half_size, y - half_size, z - half_size]
    corner2 = [x + half_size, y - half_size, z - half_size]
    corner3 = [x + half_size, y + half_size, z - half_size]
    corner4 = [x - half_size, y + half_size, z - half_size]
    corner5 = [x - third_size, y - third_size, z + half_size]
    corner6 = [x + third_size, y - third_size, z + half_size]
    corner7 = [x + third_size, y + third_size, z + half_size]
    corner8 = [x - third_size, y + third_size, z + half_size]

    # Würfel erstellen
    cube = rs.AddBox([corner1, corner2, corner3, corner4, corner5, corner6, corner7, corner8])
    rs.RotateObjects(cube,[x,y,z],angle,[0,0,1])
```