
# Курсова работа:

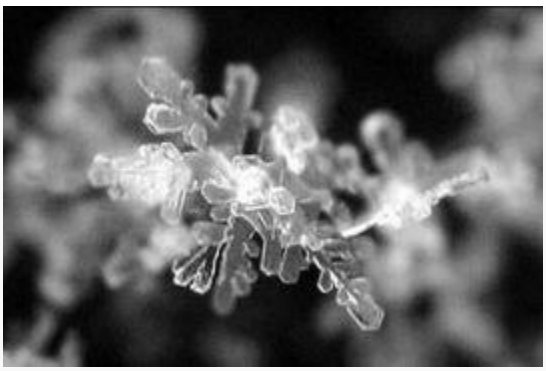
# „Снежинка на Кох“

**Изготвил: Павел Руменов Романов**

**Фн: 81762 Компютърни науки**

**гр. Плевен**

# Фрактал



Естествен природен фрактал - снежинка

Фракталът е обект с доста сложна форма, получен в резултат на прост итерационен цикъл. Итерационността и рекурсивността определят такива свойства на фракталите, както самоподобие - отделните части приличат на целия фрактал.

## Същност

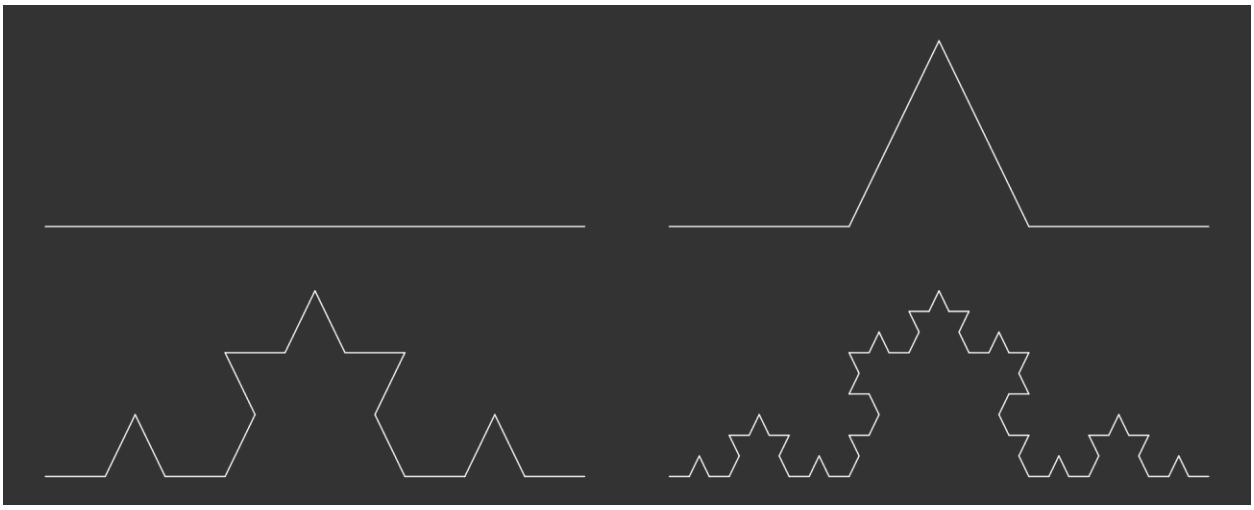
Фракталите са геометрични обекти с дробна размерност. Например, размерността на линия е 1, на площта – 2, на обема – 3. При фрактала това значение на размерността може да бъде между 1 и 2 или между 2 и 3. В математиката съществува специална сложна формула за изчисление размерността на фракталите.

Корените на теорията за фракталите могат да се проследят до опитите за измерване на периметъра (или площта, или обема) на фрактали в случаи, в които традиционният анализ е неприложим. Традиционните математически методи „се приближават“, с цел да опростят локалната картина. Съществуването на фракталите показва неприложимостта на този подход при появата не неограничено количество все по-дребни подробности.

Разклонената система от тръбички на трахеите, листата на дърветата, вените на ръцете, реките - това са фрактали. Както е казано по-горе, фракталът е геометрична фигура, определена част от която се повторя отново и отново, изменяйки се по размери. Фракталите са подобни сами на себе си, те са подобни сами на себе си на всички нива (т.е. във всеки мащаб). Съществуват много различни типове фрактали. По принцип, може да се каже, че всичко, което съществува в реалния свят е фрактал, било то облак или молекула кислород.

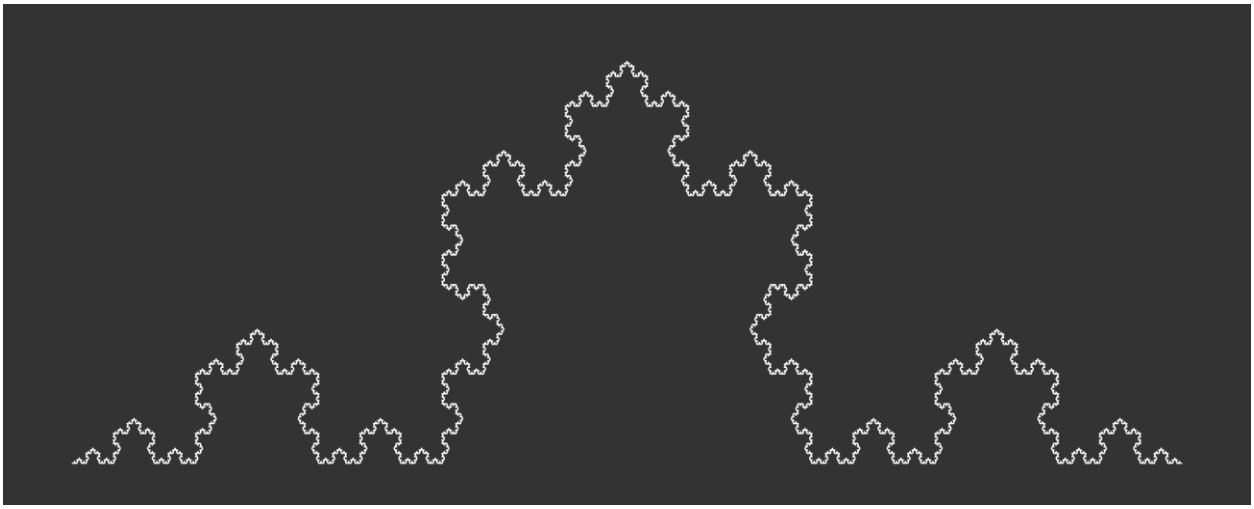
# Кривата на Кох

Кривата на Кох започва от права линия, която се разделя на три равни парчета. След това, две нови линии, дълги колкото останалите се добавят върху първоначалната линия. Така се получава триъгълник, на който махаме основата. Тези стъпки се повтарят рекурсивно за всяка една линия от кривата. Това виждаме ние след нула, една, две и три итерации:



# Как ще го програмираме

Започваме от крайната точка на първоначалната линия и правим 180 градусов завой. След това използваме факта, че ъглите на равностранен триъгълник са 60 градуса. Това означава, че първо завиваме с -60 градуса, след това със 120 градуса и накрая със -60 градуса отново за да достигнем до началната точка на линията. За първите 4 сегмента правим reduce = 1/3, а след това го правим с дължина 1, понеже няма нужда да смаляваме повече сегментите. След като сме започнали от крайната точна, накрая ще пренаредим координатите и линията за да сме сигурни че обекта е в правилният ред.



Това е резултатът от алгоритъма. Ако започнем рекурсивния процес от триъгълник, вместо от линия, ще получим Снежинката на Кох

Ще започнем със създаване на платно, на което да работим :

```
emptyCanvas <- function(xlim, ylim, bg="gray20") {
  par(mar=rep(1,4), bg=bg)
  plot(1,
       type="n",
       bty="n",
       xlab="", ylab="",
       xaxt="n", yaxt="n",
       xlim=xlim, ylim=ylim)
}
```

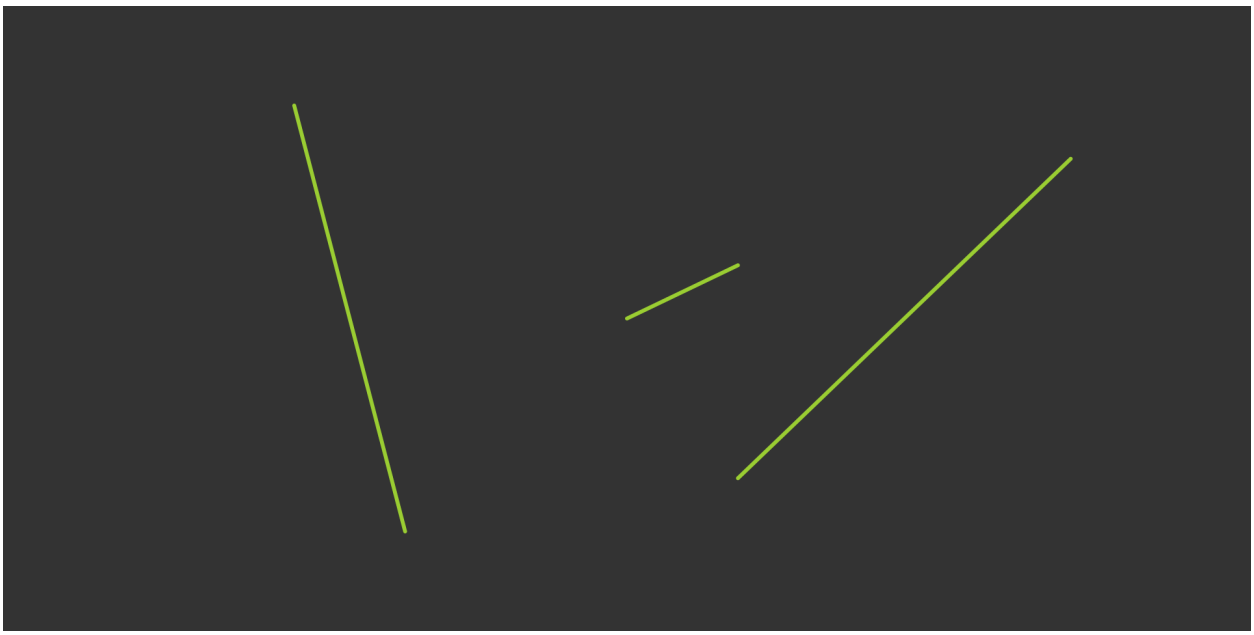
```
Пример: emptyCanvas(xlim=c(0,1), ylim=c(0,1))
```

Ще направим ф-я за създаване на линии. Всяка линия ще бъде вектор от 4 координати – x и y на началната и на крайната точка . За да начертаем няколко линии едновременно ще направим и ф-я drawObject(), която ще приема матрица и всеки ред на матрицата ще бъде различни линия:

```
drawLine <- function(line, col="white", lwd=1) {
  segments(x0=line[1],
           y0=line[2],
           x1=line[3],
           y1=line[4],
           col=col,
           lwd=lwd)
}
```

```
drawObject <- function(object, col="white", lwd=1) {
  invisible(apply(object, 1, drawLine, col=col, lwd=lwd))
}
```

```
Пример line1 = c(0,0,1,1)
line2 = c(-3,4,-2,-4)
line3 = c(1,-3,4,3)
mat = matrix(c(line1,line2,line3), byrow=T, nrow=3)
mat
```

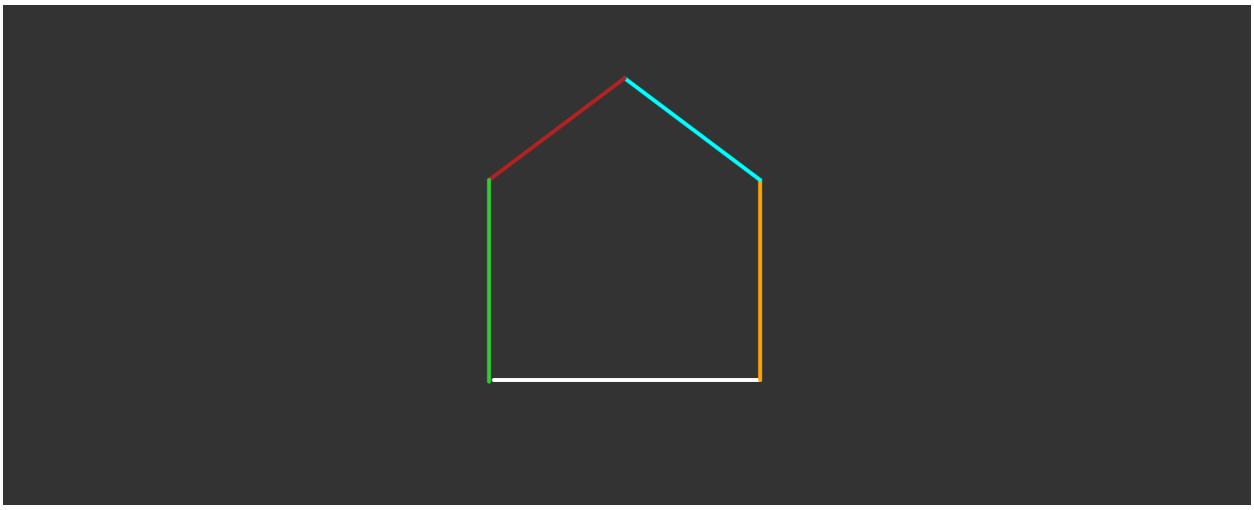


Сега ще дефинираме ф-я newline(), която ще генерира нови линии спрямо вече съществуващи. Ако имаме дадена линия а, то newline(line, angle, reduce) ще създаде нова линия б, която започва от крайната точка на а. Последните два аргумента на ф-ята контролират ъгъла между двете линии и колко по-къса е линия б спрямо а.

```
newLine <- function(line, angle, reduce=1) {

  x0 <- line[1]
  y0 <- line[2]
  x1 <- line[3]
  y1 <- line[4]

  dx <- unname(x1-x0)                       # change in x direction
  dy <- unname(y1-y0)                       # change in y direction
  l <- sqrt(dx^2 + dy^2)                    # length of the line

  theta <- atan(dy/dx) * 180 / pi           # angle between line and origin
  rad <- (angle+theta) * pi / 180           # (theta + new angle) in radians

  coeff <- sign(theta)*sign(dy)             # coefficient of direction
  if(coeff == 0) coeff <- -1

  x2 <- x0 + coeff*l*cos(rad)*reduce + dx   # new x location
  y2 <- y0 + coeff*l*sin(rad)*reduce + dy   # new y location
  return(c(x1,y1,x2,y2))

}
```

```
Пример a = c(-1.2,1,1.2,1)
b = newLine(a, angle=-90, reduce=1)
c = newLine(b, angle=45, reduce=.72)
d = newLine(c, angle=90, reduce=1)
e = newLine(d, angle=45, reduce=1.4)
```

```
# draw lines
emptyCanvas(xlim=c(-5,5), ylim=c(0,5))
drawLine(a, lwd=3, col="white")
drawLine(b, lwd=3, col="orange")
drawLine(c, lwd=3, col="cyan")
drawLine(d, lwd=3, col="firebrick")
drawLine(e, lwd=3 , col="limegreen")
```



Последната ф-я ще може да прилага фрактални функции върху някакъв набор от линии, които са в object

```
iterate <- function(object, ifun, ...) {
  linesList <- vector("list",0)
  for(i in 1:nrow(object)) {
    old_line <- matrix(object[i,], nrow=1)
    new_line <- ifun(old_line, ...)
    linesList[[length(linesList)+1]] <- new_line
  }
  new_object <- do.call(rbind, linesList)
  return(new_object)
}
```

Последно дефинираме ф-ята, която прилага алгоритъма, който ще се изпълнява рекурсивно върху всяка линия

```
koch <- function(line0) {

  # new triangle (starting at right)
  line1 <- newLine(line0, angle=180, reduce=1/3)
  line2 <- newLine(line1, angle=-60, reduce=1)
  line3 <- newLine(line2, angle=120, reduce=1)
  line4 <- newLine(line3, angle=-60, reduce=1)

  # reorder lines (to start at left)
  line1 <- line1[c(3,4,1,2)]
  line2 <- line2[c(3,4,1,2)]
  line3 <- line3[c(3,4,1,2)]
  line4 <- line4[c(3,4,1,2)]

  # store in matrix and return
  mat <- matrix(c(line4,line3,line2,line1), byrow=T, ncol=4)
  return(mat)

}

A <- c(0,1e-9)
B <- c(3,5)
C <- c(6,0)
fractal <- matrix(c(A,B,B,C,C,A), nrow=3, byrow=T)
for(i in 1:6) fractal <- iterate(fractal, ifun=koch)
emptyCanvas(xlim=c(-2,8), ylim=c(-2,5))
drawObject(fractal)
```

ето и резултатът