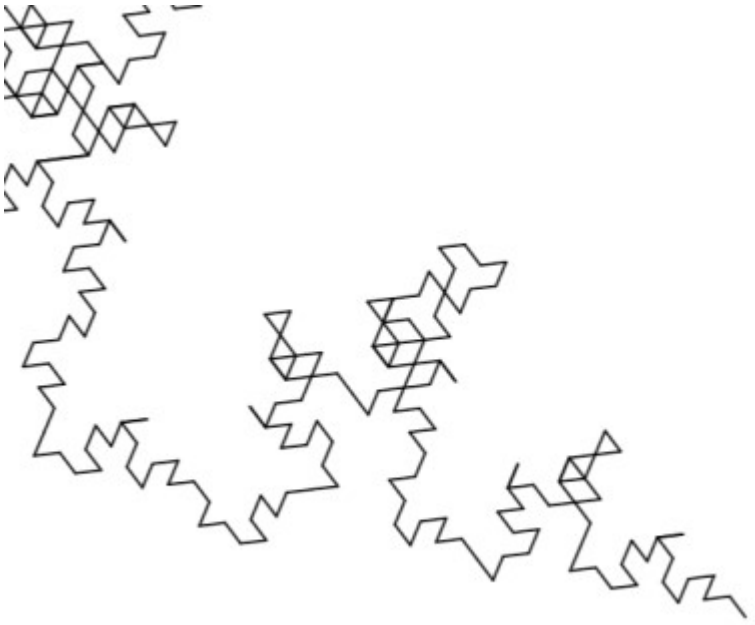
Writing useless strings has been a focus of this series, and L-Systems (Lindenmayer Systems) are no different. It is a set or a string of characters that is rewritten on each iteration, following a set of rules. Probably the most famous one is Fractal tree, Sierpinski triangle, Koch curve and many others.

It can be represented as a iterative (recursive) set of symbols, that follow set of rules.



This recursions follow strict rules in case of Fractal Tree or Koch curve, but useless functions follow sheer randomness.

To kick off this function, let's load the Turtle:

```
library(TurtleGraphics)
```

This adorable animal will do the tedious, useless, slow and tangled walk.

With common function:

```
# common function
turtlebump <- function(i, j) {
  if (i==0) {
    turtle_forward(10)
  } else {
    turtlebump(i-1, j)
    turtle_left(60)
    turtlebump(i-1, j)
    turtle_right(60)
    turtle_right(60)
    turtlebump(i-1, j)
    turtle_left(60)
  }
}
```

We will have the random part generated by a function:

```
random_turtle <- function(){

    f <- ""
    single_com <- function(){
      list_com <- c("turtle_left(","turtle_right(")
      angle <- sample(1:120, 1, TRUE)
      com <- sample(list_com,1,TRUE)
      return(paste0(com, angle, ")\n"))
    }

    comm1 <- "set.seed(2908)
    turtle_init(600, 500, 'clip')
    turtle_hide()
    i <- 8
    j <- 500
    turtle_do({"

    for (i in 1:10){
      sc <- single_com()
      i <- i + 1
      f <- paste(f, sc, collapse = NULL)
      #print(f)
      comm2 <<- f
    }

    comm3 <- "
    turtlebump(i,j)
    })"

    fin <- paste0(comm1, comm2, comm3)
    eval(parse(text=fin))

}
```

And to fire the function, just set the poor turtle for a tangled walk:

```
#run random function
random_turtle()
```

Thank you Turtle.

p.s.: No animals were harmed during function coding!