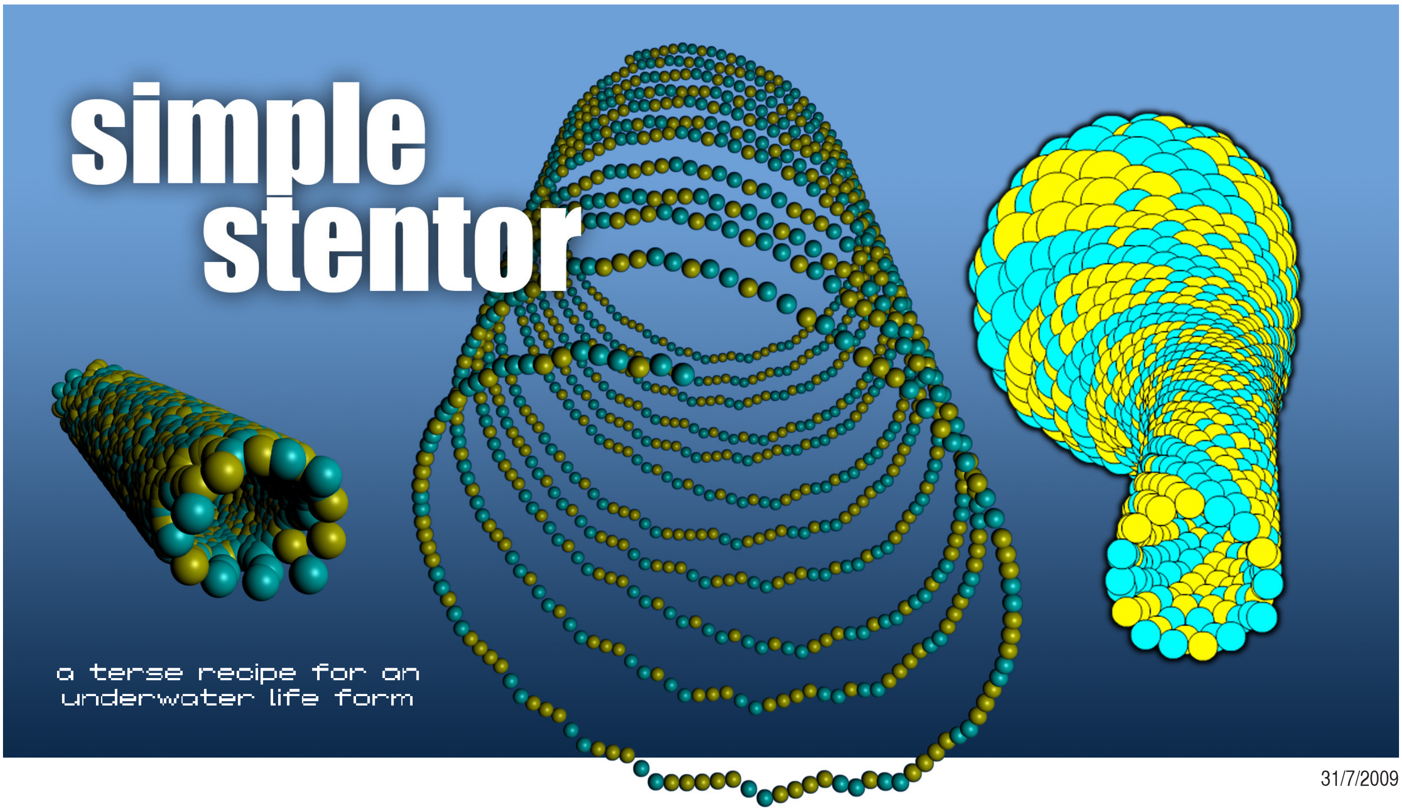


31/7/2009

Duncan Carr
scan@btinternet.com

Believing that life has an underlying simplicity, and inspired by the myriad forms that DNA can create (using just four nucleotide bases, A, G, C & T), I set out to build structures and create motion using just four building blocks. While this is in no way intended to mirror the complex chemistry of DNA, it makes it much easier to explain using this analogy. While DNA is a blueprint for proteins, I simply restrict myself to four different types of fundamental building blocks to simultaneously build the creature and to describe its motion.

In order to grow the underwater sea creature, I started off with a string of letters. I will refer to this as the creature's *initial DNA*: Any number of letters may be chosen, but, akin to DNA, the palette of options is restricted to *R*, *L*, *F* and *B*. The pattern I chose was:

FFFLFRLFRFFLL

Then it was necessary to create some *rules*, or *replacements*, for each of the four building blocks. These *rules* are used to update the string upon each iteration. These, too, are simple strings of DNA. In this instance I used:

rules["*R*"] = **L**
rules["*L*"] = **LBB**
rules["*F*"] = **B**
rules["*B*"] = **L**

Now I need to "grow" the *initial DNA* using the *rules*. To do this I simply refer to the *rule* for the appropriate letter. The value of the rule contains the replacement string. In the first examples, the letters I am replacing are highlighted for clarity.

I take the first letter of the *initial DNA*, which is an 'F':

FFFLFRLFRFFLL

The replacement rule for 'F' is 'B'. After replacement I get:

BFFLFRLFRFFLL

Now I take the second letter, which is also an 'F', and perform the same replacement of 'B', as dictated by the rules. I now have:

BBFLFRLFRFFLL

Now I take the third letter, also 'F', and do the same again:

BBBLFRLFRFFLL

The fourth letter is an 'L' - this has a replacement rule of 'LBB':

BBBLBBFRLFRFFLL

I carry on replacing each of the letters in turn with their replacements until I have dealt with each letter. By the time I have replaced all thirteen characters of our *initial DNA* with the characters from the rules, I end up with (coloured, for clarity, from this point on):

BBBLBBBLLBBBLBBLBBLBB

I now have a string of twenty one characters, "grown" from the *inital DNA* length of thirteen characters. In order to remain unambiguous I have shown how the DNA grows over the full 7 iterations.

I now substitute this new *resulting DNA* string for the *initial DNA* and start over, replacing each letter in turn with the existing replacement rules. Once I get to the last character of this string, I start over yet again. After 2 iterations I have a string of thirty three letters:

LLLLBBLLLLBBLBBLLLLBBLLLBBLLLBBLL

After 3 iterations, a *resulting DNA* of 75 characters:

LBBLBBLBBLBBLLLBBLBBLBBLBBLLLBBLLLBBLBBLBBLBBLLLBBLBB
LBBLLLBBLBBLBBLLLBBLBB

After 4 iterations, a *resulting DNA* of 141 characters:

LBBLLLBBLLLBBLLLBBLLLBBLBBLBBLLLBBLLLBBLLLBBLLLBBLBBL
BBLLLBBLBBLBBLLLBBLLLBBLLLBBLLLBBLBBLBBLLLBBLLLBBLLLB
BLBBLBBLLLBBLLLBBLLLBBLBBLBBLLLBBLL

After 5 iterations, a *resulting DNA* of 291 characters:

LBBLLLBBLBBLBBLLLBBLBBLBBLLLBBLBBLBBLLLBBLBBLBBLLLBBL
LLBBLLLBBLBBLBBLLLBBLBBLBBLLLBBLBBLBBLLLBBLBBLBBLLLBB
LLLBBLLLBBLBBLBBLLLBBLLLBBLLLBBLBBLBBLLLBBLBBLBBLLLBB
LBBLBBLLLBBLBBLBBLLLBBLLLBBLLLBBLBBLBBLLLBBLBBLBBLLLB
BLBBLBBLLLBBLLLBBLLLBBLBBLBBLLLBBLBBLBBLLLBBLBBLBBLLL
BBLLLBBLLLBBLBBLBBLLLBBLBB

After 6 iterations, a *resulting DNA* of 573 characters:

LBBLLLBBLBBLBBLLLBBLLLBBLLLBBLBBLBBLLLBBLLLBBLLLBBLBB
LBBLLLBBLLLBBLLLBBLBBLBBLLLBBLLLBBLLLBBLBBLBBLLLBBLBB
LBBLLLBBLBBLBBLLLBBLLLBBLLLBBLBBLBBLLLBBLLLBBLLLBBLBB
LBBLLLBBLLLBBLLLBBLBBLBBLLLBBLLLBBLLLBBLBBLBBLLLBBLBB
LBBLLLBBLBBLBBLLLBBLLLBBLLLBBLBBLBBLLLBBLBBLBBLLLBBLB
BLBBLLLBBLLLBBLLLBBLBBLBBLLLBBLLLBBLLLBBLBBLBBLLLBBLL
LBBLLLBBLBBLBBLLLBBLLLBBLLLBBLBBLBBLLLBBLBBLBBLLLBBLB
BLBBLLLBBLLLBBLLLBBLBBLBBLLLBBLLLBBLLLBBLBBLBBLLLBBLL
LBBLLLBBLBBLBBLLLBBLBBLBBLLLBBLBBLBBLLLBBLLLBBLLLBBLB
BLBBLLLBBLLLBBLLLBBLBBLBBLLLBBLLLBBLLLBBLBBLBBLLLBLBB
LBBLLLBBLBBLBBLLLBBLLLBBLLLBBLBBLBBLLLBBLL

After 7 iterations, a final *output DNA* of 1,155 characters:

LBBLLLBBLBBLBBLLLBBLLLBBLLLBBLBBLBBLLLBBLBBLBBLLLBBLB
BLBBLLLBBLLLBBLLLBBLBBLBBLLLBBLBBLBBLLLBBLBBLBBLLLBBL
LLBBLLLBBLBBLBBLLLBBLBBLBBLLLBBLBBLBBLLLBBLLLBBLLLBBL
BBLBBLLLBBLBBLBBLLLBBLBBLBBLLLBBLLLBBLLLBBLBBLBBLLLBB
LLLBBLLLBBLBBLBBLLLBBLLLBBLLLBBLBBLBBLLLBBLBBLBBLLLBB
LBBLBBLLLBBLLLBBLLLBBLBBLBBLLLBBLBBLBBLLLBBLBBLBBLLLB
BLLLBBLLLBBLBBLBBLLLBBLBBLBBLLLBBLBBLBBLLLBBLLLBBLLLB
BLBBLBBLLLBBLBBLBBLLLBBLBBLBBLLLBBLLLBBLLLBBLBBLBBLLL
BBLLLBBLLLBBLBBLBBLLLBBLLLBBLLLBBLBBLBBLLLBBLBBLBBLLL
BBLBBLBBLLLBBLLLBBLLLBBLBBLBBLLLBBLLLBBLLLBBLBBLBBLLL
BBLLLBBLLLBBLBBLBBLLLBBLBBLBBLLLBBLBBLBBLLLBBLLLBBLLL
BBLBBLBBLLLBBLBBLBBLLLBBLBBLBBLLLBBLLLBBLLLBBLBBLBBLL
LBBLBBLBBLLLBBLBBLBBLLLBBLLLBBLLLBBLBBLBBLLLBBLBBLBBL
LLBBLBBLBBLLLBBLLLBBLLLBBLBBLBBLLLBBLLLBBLLLBBLBBLBBL
LLBBLLLBBLLLBBLBBLBBLLLBBLBBLBBLLLBBLBBLBBLLLBBLLLBBL
LLBBLBBLBBLLLBBLBBLBBLLLBBLBBLBBLLLBBLLLBBLLLBBLBBLBB
LLLBBLBBLBBLLLBBLBBLBBLLLBBLLLBBLLLBBLBBLBBLLLBBLLLBB
LLLBBLBBLBBLLLBBLLLBBLLLBBLBBLBBLLLBBLBBLBBLLLBBLBBLB
BLLLBBLLLBBLLLBBLBBLBBLLLBBLBBLBBLLLBBLBBLBBLLLBBLLLB
BLLLBBLBBLBBLLLBBLBBLBBLLLBBLBBLBBLLLBBLLLBBLLLBBLBBL
BBLLLBBLLLBBLLLBBLBBLBBLLLBBLLLBBLLLBBLBBLBBLLLBBLBBL
BBLLLBBLBBLBBLLLBBLLLBBLLLBBLBBLBBLLLBBLBB

The result is a very simple pattern. As can be seen from this example, due to the replacement rules, I have ended up with only 2 constituent parts, 'L's and 'B's (as the replacement rules contained no 'F's or 'R's).

As I intend to illustrate this *resulting DNA* in pseudo three-dimensions, and/or in three-dimensions proper, I needed to restrict the number of primitives I will be creating. With a more powerful computer, or more efficient programming methodology, this could easily be iterated many more times, adding further realism to the result.

Illustrating the creature

I devised a very simple method to illustrate the resulting DNA. In a similar fashion to "growing" the DNA I pluck each character off, one by one, from the head of the string, and update our coordinates in three-dimensional space. I simply start off in the centre of our three-dimensional world, x, y and z all set to zero. I use variables *xposNew*, *yposNew* and *zposNew* to keep track of the current position.

The letters simply denote a move in three-dimensional space to a new coordinate. The variable *angle* is initially set to zero, but is updated upon each iteration. The variable *distance* is simply the diameter of the spheres that ultimately make up the structure:

if letter is equal to *R:*

xposNew += cos(angle * distance)
yposNew += sin(angle * distance)

(else)
if letter is equal to *L:*

xposNew -= cos(angle * distance)
yposNew -= sin(angle * distance)

(else)
if letter is equal to *F:*

xposNew += cos(angle * distance)
yposNew += sin(angle * distance)
zposNew += cos(angle * distance)

(else)
if letter is equal to *B:*

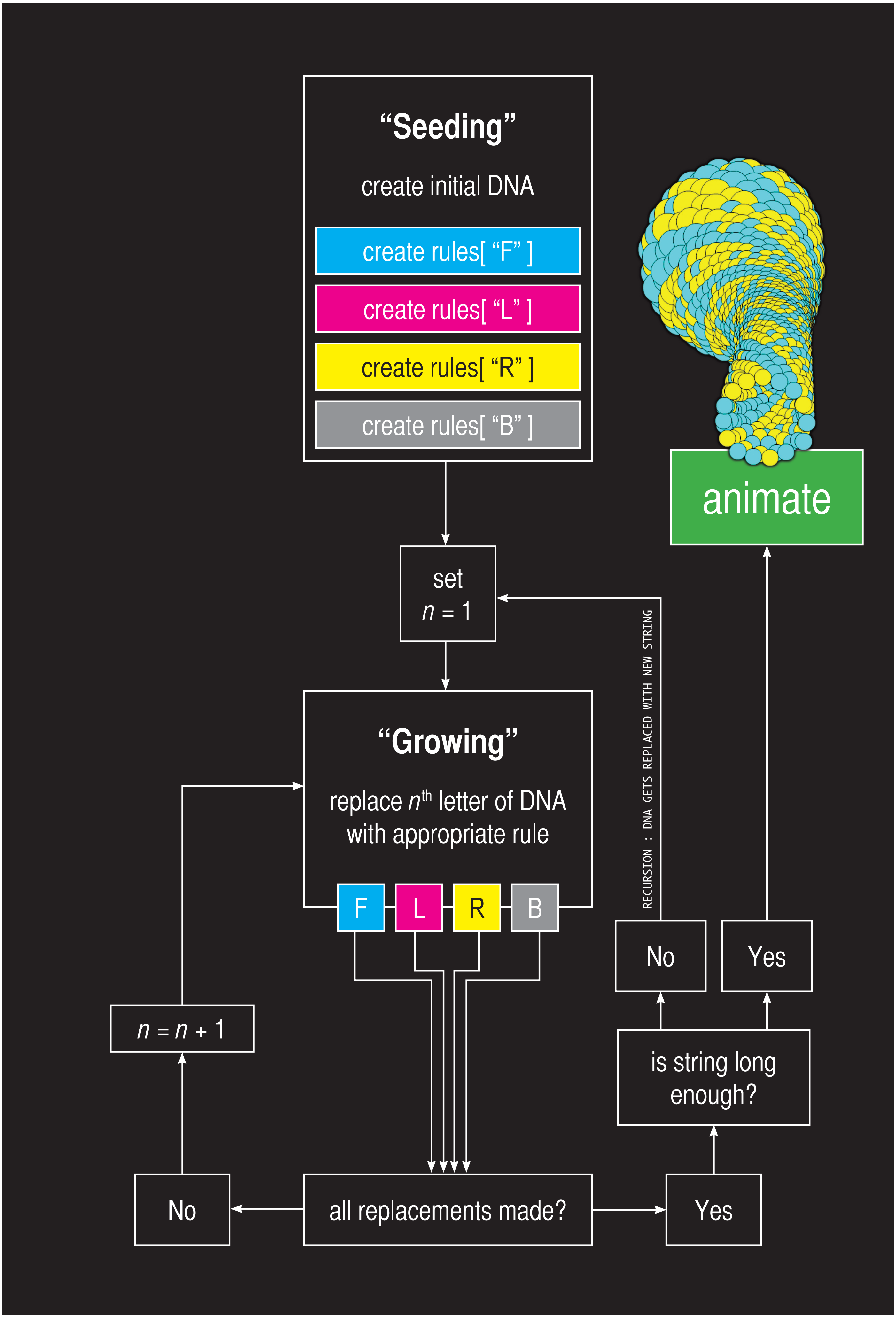
xposNew -= cos(angle * distance)
yposNew -= sin(angle * distance)
zposNew += cos(angle * distance)

Note:
+= *add result to current value*
-= *subtract result from current value*

The increment for this structure and motion is 56.95122°. A modification of the increment of just 1/10,000° (one ten-thousandth of a degree) causes a vast change in its behaviour.

Upon each iteration, a modification of the *angle*, incremented by the value of the *increment*, is applied to each sphere that makes up the creature (the *output DNA)*, and the structure morphs. Sometimes this produces a very unexpected result; in that occasionally the *output DNA* and the *increment* (in this case 56.95122°), seem to be in harmony with the one another, and a rich, natural, stentor-like structure and movement is revealed.

http://www.my-mot.co.uk/automata/


"Seeding"
create initial DNA
create rules["F"]
create rules["L"]
create rules["R"]
create rules["B"]
animate
set
$n = 1$
RECURSION : DNA GETS REPLACED WITH NEW STRING
"Growing"
replace n^{th} letter of DNA
with appropriate rule
F
L
R
B
No
Yes
$n = n + 1$
is string long
enough?
No
all replacements made?
Yes