

Biomorphs

In *The Blind Watchmaker*, the author, Richard Dawkins, discusses how organisms evolve because random mutations in the genotype result in a change in phenotype, which then subject to selection pressure. The set of all possible phenotypes forms a high dimensional phenotype-space and evolution can be viewed as wandering through the various possible phenotypes on a path that increases in fitness. Dawkins creates digital creatures, which he called biomorphs, to illustrate this idea and this is what we will do in this lab. We will use NetLogo to build a geometric structure following a simple set of rules. The rules then become a genome for the creatures, which is subject to random mutations from one generation to the next. The selection pressure comes from you. You choose which mutated form to breed and which to let die. In the end you wander through biomorph-space until a pleasing form emerges.

Making Branching Creatures

The basic form we will choose for our creatures is a branching structure. The main idea is to create a turtle that plays the role of a seed. It produces a bud, which moves forward, with the pen down to draw a stem, it then hatches two new buds, and dies. Each new bud then repeats the process a certain number of times until a tree-like structure is formed.

So that we all start with the same code, I've included two procedures that work together to implement this idea below. The first creates the "seed" turtle wherever the mouse is clicked and assigns the seed some parameter values that will determine how it grows. The second procedure makes the seeds grow into a tree-like structure. You will need to create two new breeds (seeds and buds) and make the variables `n`, `step`, and `angle`, into turtles-own variables. Make a `go` procedure that calls the two procedures below and create a `setup` procedure that clears the screen. Add a `setup` and a `go` button on your interface. Give yourself lots of space by changing the world-view to 100 by 100 with patch size 5.

```
to plant-seeds
  if mouse-down? [
    create-seeds 1 [
      setxy mouse-xcor mouse-ycor ;put seed where the mouse is
      set heading 0 ; face the top of the screen
      set n 5 ; number of repetitions
      set step 4 ; length of each branch
      set angle 25 ]] ; angle of branching
  wait 0.1 ; wait 0.1 seconds before planting next seed
end

to grow-seeds
  ask seeds [
    pendown ;get ready to draw the branches
    hatch-buds 1 [] ; make a bud to draw the main stem
    repeat n [
      ask buds [
        fd step ;draw the stem
        hatch 1 [rt angle] ;make a new bud on the right
        hatch 1 [lt angle] ;make a new bud on the left
        die ]] ; central bud dies after drawing stem
    ask buds [die] ; branching buds die after drawing branches
    die ] ; remove seeds so they don't grow again
end
```

You should get a basic tree structure whenever you click the mouse on the screen. Try changing the values assigned to `n`, `step` and `angle` in the `plant-seeds` procedure to see how changing the parameters changes the shape of your biomorph. Choosing very large angles make for interesting structures, that don't look much like trees. Play around with these values for a while to see what gives interesting shapes. You could make sliders called `n-value`, `step-value` and `angle-value` corresponding to the values you give these variables. Careful, don't let the `n-value` slider go higher than 12 or you may crash NetLogo.

With the current set-up you have three variables you can adjust to vary the biomorphs. You can think of these variables as the genes of the creatures, and changing the variables corresponds to sweeping through three-dimensional biomorph-space.

Now let's try modifying our procedure so that it has a few more genes to vary. One thing we might consider doing is breaking the symmetry of our creatures a little.

Symmetry breaking

We will keep bilateral symmetry – that is, mirror symmetry on the main axis – but after the first branch we will allow left branch angles to be different from right branch angles. We'll need to keep track of which bud is on the right and which is on the left. First add `turtles-own` variables called `right?`, `angle-1` and `angle-2`. Now add lines in your `plant-seeds` procedure to give values to `angle-1` and `angle-2` (try 30 and 90). Then modify your `grow-seeds` procedure so that a seed hatches not one bud, but two buds, one which will have `right? set to true` and will turn `right` by an amount `angle` and the second will have `right? set to false` and turn `left` by an amount `angle`. Do this by replacing the line

```
hatch-buds 1 []
```

with the lines

```
hatch-buds 1 [set right? true
              right angle]
hatch-buds 1 [set right? false
              left angle]
```

Then we want the right bud to make branches that are asymmetrical, and the left bud to make branches that are mirror images of the right branches. So in the `repeat` command block replace the lines where you ask buds to hatch a new bud on the right and another on the left, with the following lines which hatch buds at different angles depending on whether they are right or left buds.

```
ifelse right?
  [ hatch 1 [rt angle-1]
    hatch 1 [lt angle-2] ]
  [ hatch 1 [lt angle-1]
    hatch 1 [rt angle-2] ]
```

You now have increased the number of genes from three to five genes. Try it out using different values for the new variables, you should get some more variety, depending on how large you make `angle-1` and `angle-2`. You could add sliders called `angle-1-value` and `angle-2-value` to represent the values you give these variables. This will help you explore the model.

We will now add three more genes (variables) to the genome of our creatures. The creature is made by repeating a series of commands inside the `repeat` command block `n` times. In an actual tree the branch length (`step`) decreases in size as you go up the tree. A similar thing happens with the branching angle. Each time we run the commands in the `repeat` command block lets change the `step` size and the two angles `angle-1` and `angle-2` by multiplying by some factor. Add new variables called `step-factor`, `angle-1-factor` and `angle-2-factor` to the `turtles-own` list and then in your `plant-seeds` procedure assign values to these variables (try using something like 0.8 for the `step-factor` and 2 for the angle factors).

Now, inside the `repeat` command block of your `grow-seeds` procedure include lines that change `step`, `angle-1` and `angle-2` by the appropriate factors. Now see what happens for different values of these factors. You should get interesting creatures – especially if the angle-factors are large. (You get a mess if the step factor is large!).

Assignment

At the moment you change your creatures by choosing values for the various genes. This is not how evolution does it. For evolution you need variation (different genes for different seeds), selection, mutation and reproduction. We will start by creating nine seeds, each of which is a slightly mutated version of its neighbor and arranging them evenly around the screen. Then we will select the “fittest” creature, and this one will be allowed to reproduce. Its offspring will all be displayed on the screen, each of which will have a slight mutation, and then we select again. By a gradual process of mutation and selection we will see the creatures evolve.

1. In your `plant-seeds` procedure remove all reference to the mouse – you will not `plant-seeds` with the mouse any more. Create 9 seeds instead of 1. After the `create-seeds` command block call an `arrange-seeds` procedure. You need to write this procedure to place the seeds (which will have `who` numbers 0 through 8) evenly around the world. Leave turtle 8 at the center of the screen and then place turtles 0 through 7 in a square around the center using the `setxy` command. Make good use of space.
2. Now we want to give each of the seeds a genome. We can do this by defining a `seeds-own` variable called `genome`. This will be as a variable that is a list that contains the values you assigned to the different variable names using the sliders. As an example, suppose I only have the variables `n`, `step` and `step-factor` as genes, then I would include the following line as the last entry in the `create-seeds` command block in the `plant-seeds` procedure to make my genome list

```
set genome (list n step step-factor)
```

You will have a longer list. At the moment each seed will have the same genome, that will change after we mutate them. (Remember the order of your the list. It is important.)

3. In order to allow your seeds to mutate define a `globals` variable called `mutation-rates`. This will be a list that defines the increments by which each of the genes can change in each generation. The order of this list should be the same as the order for the genome list. For example if I want `n` to change by 1, and `step` to change by 1 and `step-factor` to change by 0.1 each generation I would define `mutation-rates` as follows:

```
set mutation-rates [1 1 0.1]
```

Add a line like this, with your complete list of mutation rates, to your `setup` procedure right

after you `clear-all`. When choosing the mutation rates, make sure there is scope for small, but still significant change. Use your earlier experiments to choose sensible values.

4. Now you need a `mutate` procedure that mutates the genome of turtles 0 through 7. (turtle 8 will be left un-mutated) . Since we have 8 genes lets have turtle 0 be the one with gene 0 mutated and turtle 1 have gene 1 mutated and so on. To mutate a gene we have to randomly add or subtract the corresponding value from the `mutation-rates` list for the gene we want to mutate. The syntax for changing values in lists is a little awkward so I'll include the code for the `mutate` procedure below. Make sure you understand what each part does. Call the `mutate` procedure right after arranging the seeds in the `plant-seeds` procedure. Here is the code:

```
to mutate
  ask seeds with [who < 8 ][
    ifelse (random 2) = 1      ; add the mutation rate if 1 and subtract if 0
    [ set genome replace-item who genome ( (item who genome) + (item who
mutation-rates) ) ]
    [ if (item who genome) > (item who mutation-rates)
      [ set genome replace-item who genome ( (item who genome) -
(item who mutation-rates) ) ] ] ]
end
```

5. Now that your seeds are planted and mutated they are ready to grow. For the seeds to grow they need to know what the values of their genes are. Although you have given these values to the `genome` list you haven't yet assigned these values to the variable names you use in your `grow-seeds` procedure. Define an `assign-values` procedure where you assign each variable the appropriate value in the `genome`. For example, if the variable `n` corresponds to the first entry in the `genome` list and `step` was the second value in the `genome` list then you would have the following lines in the `assign-values` procedure.

```
set n (item 0 genome)
set step (item 1 genome)
```

Note the typical programming convention that the first item in a list is called item 0 and the second item is called item 1. Complete this procedure, and then call it in the `plant-seeds` procedure immediately after the `mutate` procedure.

6. Now remove calls to `plant-seeds` and `grow-seeds` from your `go` procedure and put them in your `setup` procedure. When you press `setup` you should get nine slightly different creatures. It should all work wonderfully now, once you remove any bugs. Actually it would be helpful if you also remove the last `die` command in your `branch` procedure, where you removed the `seeds`. We will want to keep the seeds around for the final part of the program.
7. You are nearing the end of this lab. You now need to modify your `go` procedure so that you can use the mouse to select your favorite creature. Define a new `globals` variable called `fittest` at the top of the procedures tab. This variable is the creature we will select with the mouse. In the `go` procedure include a line that sets `fittest` to be the seed closest to the mouse whenever it is pressed down. You'll need to use the `distancexy` reporter and the `min-one-of` reporter for this. I'll let you look these up in the manual.
8. Now in the `go` procedure, whenever the fittest seed is selected with the mouse have all seeds set their `genome` to the `genome` of the `fittest` seed, then clear the screen of all the drawings (but not the seeds), `mutate` the seeds, `assign-values` to their genes and

then grow your newly evolved creatures by calling `grow-seeds`.

9. You should now be able to select your favorite turtle with a mouse click. When you do, you should see your selection appear in the middle. If your mutation rates are about right you should be able to weave your way through biomorph-space, selecting creatures without much care for the actual genes. If you want to keep track of the actual gene's you should add monitors showing each of the gene's for turtle 8, who will always be the latest selected.
10. Although we have no room left for genes, you might want to change the color of your creatures by linking the color to one of the other genes. For example in the `repeat` command block you could add a line: `set color (angle-1 - angle-2)` and see what it does (feel free to try other things).
11. Now have some fun. Evolve three new creatures, and include the genetic code for these creatures in the information tab of your interface. Then Save your model, with naming convention "lastname_firstname_biomorph.nlogo". When you are finished the homework drop this file in the dropbox folder. The due date is March 4th.