Writing Procedures

In the previous lab you learned how to control NetLogo Models using sliders and buttons and you also learned how to use the Command Center and agent monitors to interact with patches and turtles. In this lab you will learn how to control the interactions of patches and turtles by writing procedures. Procedures are groups of NetLogo *commands* or *reporters. Commands* are actions that patches or agents carry out. *Reporters* are commands that return a value which might be acted on by another command. Together these are referred to as NetLogo *primitives.* An example of a simple procedure is given below:

```
;; The following is a turtle procedure that moves a turtle
;; one step at a heading which is within plus or minus 40 degrees
;; of the original heading.
to wiggle
     ask turtles [
          left random 40
          right random 40
          forward 1
     ]
end
```

In this procedure `ask, left, right` and `forward` are commands, and `random 40` is a reporter that returns an integer between 0 and 39. Note that a procedure starts with a `to` and end with an `end.` When you are programming it is important to indent, so that the code is easier to read. Please also get into the habit early of adding comments to your procedures so that it is clear to others (and yourself at a later date) what each component of your program does. Comments are added using semi-colons.

Tutorial #3: Procedures

Tutorial #3 from the NetLogo User's Manual is a well written tutorial that introduces many important aspects of writing procedures in NetLogo. Please work through this tutorial slowly and carefully. As you learn a new command or reporter take the time to play around with it so that you fully understand how to use it and what it can do. Resist the urge to copy and paste code from the Tutorial to your program. As you type it yourself you will reinforce the logic of each command. At the end of the tutorial you will be well on your way to writing your own programs.

Extension to the Model

In the tutorial you created a program that models how an organism can find the maximum of some quantity by following the direction of the local gradient (ie the steepest slope). As the program is written the turtles are very successful at finding local maxima, but struggle to find the global maximum, unless the smoothness is set so high that there is only one maximum! At the end of the tutorial you are asked to think about ways to get turtles to find the global maximum. There are a couple strategies you can take here. One is simply to have the turtles jump to a random spot on the screen if they find themselves at a local maximum that is not near enough to the highest value. Modify your program to implement this strategy. The problem with this strategy from the point of view of emerging order, is that the turtles may not have a reason to know they are not at the global maximum. In the homework you will investigate a more natural strategy.

Homework Questions: Finding Maximum Fitness

In *Life's Other Secret,* you read about the so called fitness landscape. The idea is that phenotypic variables such as pigment, or size form variables in a morphological landscape, whose elevation is "fitness". The fitter the organism the more likely it is to produce offspring. Offspring of organisms may mutate – appearing in a different part of the landscape from their parent. Those that emerge at a higher elevation are fitter and will breed more rapidly. Over time organisms evolve until most are at a peak of the landscape. This method of using evolution, rather than gradients, to find local maxima can be extremely efficient. It is also easy to implement an algorithm that can find global maxima.

1. Save your tutorial model, giving it the name "lastname_firstname_week_2.nlogo". When you are finished the homework drop this file in the dropbox folder.
2. Make the following changes to your interface:
   (a) Change the graphics window so that the Screen Edge X and Y are both 50 and the patch size is 4.
   (b) Change the smoothness slider so it has a maximum of 50.
3. Make the following changes to the the setup-patches procedure:
   (a) When coloring the patches have the color scale range between `lowest` and `highest` rather than 1000 to 9000. Explain what effect this has on the landscape and why?
   (b) After the command that finds the highest and lowest elevation, rescale the elevation variable so that lowest is 0 and highest is 1000. (Note: You cannot simply redefine highest and lowest – you have to rescale the elevation variable for all the patches appropriately.)
4. Make the following changes to the `go` procedure:
   (a) Comment out references to movement – the turtles do not move in this program.
   (b) Call two new procedures: `birth` and `death`. You will write these procedures.
5. Write the `birth` procedure so that it has the following properties:
   (a) a turtle hatches offspring with a probability proportional to its elevation (or fitness). (Any turtles who happen to be at zero elevation should not reproduce and any turtles who are at elevation 1000 should always reproduce.)
   (b) Hatch new offspring so that they have the same color as their parent but scaled according to their fitness. Let high fitness be a darker shade than low fitness (so that turtles can be seen on the landscape).
   (c) Hatch new offspring at a distance `mutation-rate` from their parent and have them face in a random direction. Define `mutation-rate` with a slider ranging from 0 to 10 on the interface.
6. Write the `death` procedure in a way that keeps the population size *roughly* constant. (Kill off turtles randomly if the population is greater than the starting number).
7. Add a new plot and a new monitor to your interface, showing the average fitness of the turtles as it change over time.
8. Now use your new model to investigate how the turtles find maxima. You should see that turtles find their way to the global maximum fairly often, but not always. Experiment with different landscapes. Does the smoothness of the landscape affect the average fitness? Experiment with the `mutation-rate` slider. How does increasing `mutation-rate` affect average fitness? When turtles get trapped on a local maximum, you can often nudge them over to the global maximum by increasing `mutation-rate` for a while and then decreasing it again. Can you think of evolutionary pressures that might increase mutation rates like this from time to time?