Chapter 9 Exercise Hints and Solutions

Agent-based and Individual-Based Modeling: *A Practical Introduction, 2nd Edition*

Exercise 1

The trick for pointing turtles in the direction they moved is to use the primitive `face`. First, identify the patch the turtle will move to, then face it, then move. The move procedure can be changed to this (which also corrects the `uphill` mistake discussed in Section 6.3.10):

```
to move  ; The butterfly move procedure, in turtle context

  ; Decide whether to the highest surrounding patch with probability q
  ; Identify the patch to move to so turtle can face it
  let next-patch one-of neighbors
  if random-float 1 < q
    [ set next-patch max-one-of neighbors [elevation] ]     ; Move uphill

  ; Now face the destination and go there
  face next-patch
  move-to next-patch

  ; Tell the patch that a butterfly has been there
  set used? true

end ; of move procedure
```

One way to see the effect is to "inspect" a patch near where there are several turtles.

2. This code causes one of the turtles to move 20 steps, but you cannot see it move. That is because the View is not updated until after the 20 steps are finished. To see the turtle walk, force the View to update via the `display` primitive:

```
; For Exercise 2 of Ch. 9, make one turtle take a walk
ask one-of turtles
[
  repeat 20
  [
    set heading (heading - 30 + random 60)
    fd 1
    display
  ]
]
```

However, even the above code often executes so quickly that you cannot see the walk. Add the statement `wait 0.1` after `display`.

Exercises 3-4

With the instructor materials is a NetLogo file `Ch09-Ex1-4_ButterflyModel.nlogo` that includes all the modifications made in exercises 1 through 4.

One of the challenges in exercises 3 and 4 is that now there are several plots on the Interface, so you must say, in the code, which plot you are updating. In the NetLogo file we provide as an example solution, all the plot updates are moved to one procedure that is called from the `go` procedure, as recommended in Section 9.7.

Making the histogram work for Exercise 4 involves some tricks. First, when creating or editing the histogram's plot element on the Interface, click on the pencil icon so you can edit its settings. Select "bar" mode instead of the default line. Also in the editing dialog, you must set the bar interval: what range of elevations do you want each bar to represent?

When you first create the histogram, it may appear blank and seem not to update. This is most likely because its "X max" value is less than the elevation of any of the turtles. Right-click on the plot, select "Edit". Set X max to a number just higher than the highest elevation in the model (which you can find by entering `show max [elevation] of patches` in the Command Center). You can likewise set the "X min" value to a number just below the lowest elevation in the landscape.

For the real landscape data we provide for Section 5.5, the histogram works well with: X min = 450, X max = 700, Interval = 10.

Exercise 5

One way to answer this question experimentally is to create a procedure like this and have a monitor call it from the Interface:

```
to-report ticks-since-update
  ; last-tick is a new global variable
  let result ticks - last-tick
  set last-tick ticks

  report result

end
```

This reporter just reports how many ticks have been executed since it was last called; it is called only when the Interface monitor updates.

When added to the Flocking model, a monitor to this `ticks-since-update` reporter should show values of both 0 and 1. That means that it sometimes is updated more than once per tick. But if you speed up the model, e.g., by commenting out everything else in the `go` procedure except the `tick` statement, the monitor can report values greater than one—several ticks are completed between monitor updates. These results indicate that monitors are updated several times per second on a schedule that is not related to the model's ticks.

The NetLogo documentation for this question is in the "Interface Guide" in the table describing the various Interface elements.

Exercise 6

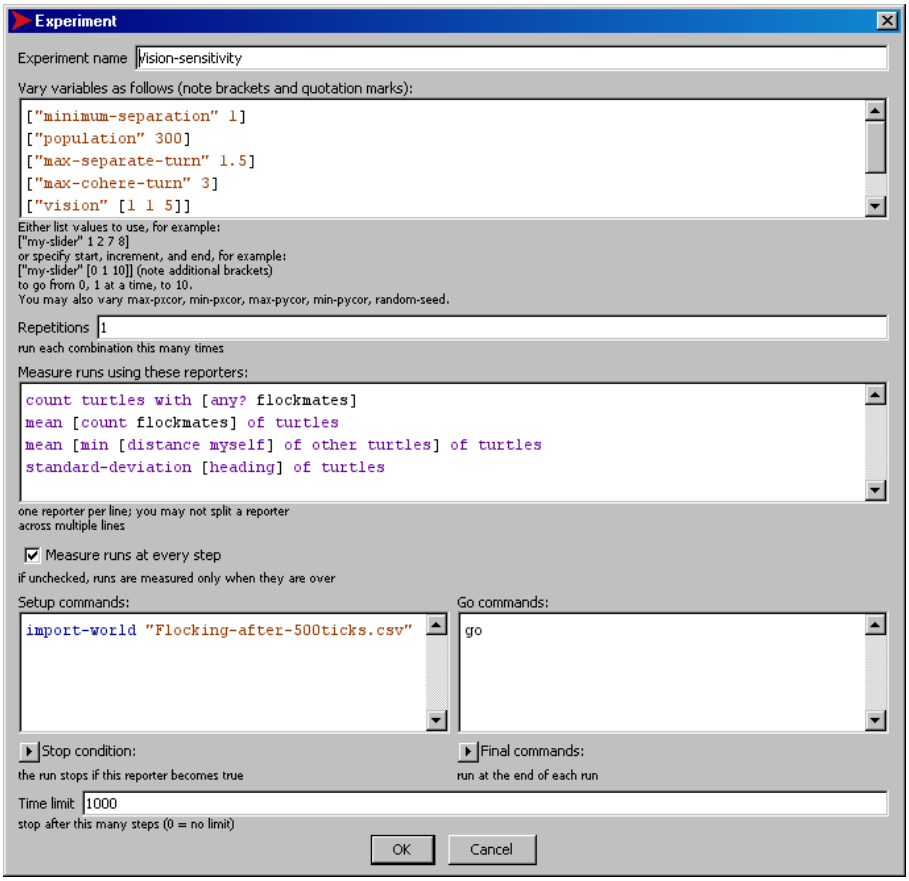
The trick is mentioned in Section 9.6: use `export-world` to save the state of the whole model after 500 ticks, then start each BehaviorSpace run by importing the saved state. You can, for example, change `setup` so it runs the model for 500 steps and then calls `export-world`:

```
to setup
  clear-all
  crt population
    [ set color yellow - 2 + random 7  ;; random shades look nice
      set size 1.5  ;; easier to see
      setxy random-xcor random-ycor ]
  reset-ticks

  ; Now execute a 500-tick warm-up period and save the model's state
  repeat 500 [ go ]
  export-world "Flocking-after-500ticks.csv"

end
```

Then, in BehaviorSpace, change the "Setup commands" to just import the saved world and run 1000 more ticks:



Exercise 8

A file implementing the refresher exercise (`Ch09-Ex8_NetLogoExercises.nlogo`) is available with the instructor materials for this chapter.

At Step 7 of this exercise, the problem is that we forgot to put `clear-all` in the setup procedure.

At Step 14, make sure the `set label age` statement is in the `go` procedure right after the age is increased (as well as in `setup`).

- If the turtles' label does not increase when you hit the `go` button, it is because you forgot to update the label after you increment `age`. NetLogo does not automatically update labels when the variable they show changes (see the Programming note on page 118)!
- If you get an error about "GO is turtle-only", it is because some of the statements in the `go` procedure that turtles are supposed to execute (like `set label age`) are not inside the brackets of an `ask turtles` statement. That makes NetLogo think that the whole `go` procedure must be in turtle context.