13.1 NetLogo Tutorial 1c

Introduction to Computational Science:
Modeling and Simulation for the Sciences, 2nd Edition

Angela B. Shiflet and George W. Shiflet
Wofford College

Introduction

This document contains *NetLogo* Tutorial 1c, the third of a three-part tutorial. Tutorials 1a-1c give an introduction to the system and prepare you to understand a *NetLogo* implementation of the model in Module 11.2, "Agents of Interaction: Steering a Dangerous Course," and to use the software to complete various projects in Chapters 11 and 14. Tutorials 1a and 1b are prerequisites to this tutorial. Unlike those tutorials, Tutorial 1c does not develop a particular model but improves your facility with using the *NetLogo* Dictionary and covers some of the commands employed in the implementation of Module 11.2's model. We recommend that you work through the introduction with a copy of *NetLogo*, answering all Quick Review Questions in a separate document.

Patches

Type all requested commands in a *NetLogo* file and an answer document. When requested to "look up" a command, consult the *NetLogo* Dictionary.

Quick Review Question 1 Start a new *NetLogo* simulation and save the file.

a. How do we make a *setup* button? Do so.
b. In the *NetLogo* Dictionary, look up ***patches-own*** and give what the command defines.
c. In the code, define *agentType* as a variable that patches own. Give the command in the answer document.
d. Write the shell of a *setup* function, which initially has no commands, but has opening and closing lines.
e. Inside *setup*, write a statement to clear everything.
f. Look up ***world-height*** and ***world-width*** and give what they return.
g. On the Interface level, in the *observer>* box at the bottom, type a command to show the world's height and give the returned height.
h. Look up ***pxcor*** and ***pycor*** and give what they hold.
i. Right-click on the patch in the upper-left corner and inspect this patch, which is *patch -16 16*. Notice that the patch has an associated variable called *agentType*, which you declared in Part c. Give the patch's *x*- and *y*-coordinates.
j. In the same inspection window for *patch -16 16*, change the patch's *plabel* to "x", which is a string containing the letter x, and press <RETURN> or

<ENTER>. What appears in the world? Change the patch's *plabel* back to the empty string, "", and close the inspection window.

k. In *setup* and in the answer document, write code to ask each patch to do the following: if your *y*-coordinate is 0, make your *agentType* be "Fence" and your *plabel* be "|".
l. How do you check your code?
m. On the Interface level, click *setup* and indicate how the world appears now.
n. Inspect a patch in the middle of the world and give its *plabel* and *agentType*.

Selection

Quick Review Question 2

a. Look up ***or***. Suppose variable *boolean1* is true and *boolean2* is false. Give the value of *boolean1 or boolean2*.
b. Suppose variable *boolean1* and *boolean2* are both false. Give the value of *boolean1 or boolean2*.
c. Look up ***and***. Suppose variable *boolean1* is true and *boolean2* is false. Give the value of *boolean1 and boolean2*.
d. Suppose variable *boolean1* and *boolean2* are both true. Give the value of *boolean1 and boolean2*.
e. In *setup* after the closing square bracket,], of the *if* statement while still in the closing bracket for *ask patches*, write another *if* statement that does the following: If the patch has *y*-coordinate less that 0 or has *y*-coordinate greater than 0, make *agentType* be "Farm". Thus, there will be a farm to the north and one to the south. Check your work. We must be careful to connect two complete true-false clauses with *or*. *NetLogo* will not allow us to use "or > 0". We must compare 0 to a variable.
f. On the Interface level, click *setup*. Inspect a patch above and below the "fence." What are their values of *agentType*.
g. Look up the command ***ifelse***. This command gives a two-way selection choice. In *setup* inside the block for *ask patches*, have *NetLogo* do the following:

```
if the patch is to the south of the fence
      color the patch green
else
      color the patch brown
```

Recall that *pycor* being less than zero indicates a patch to the south. Check the code.

h. On the Interface level, click *setup*. What color is the background of the fence?
i. Still within the block for *ask patches*, write an *if* statement to turn the patch at location (0, -8) blue. Thus, the patche's *x*-coordinate is 0 *and* its *y*-coordinate is -8.

Breeds

It is often useful to have several "breeds" of turtles. For example, we might have a simulation that has sharks and fish, each with different appearance and behavior. At the top of the program, we declare the plural and singular form of the agent with a ***breed*** command, as follows:

```
breed [sharks shark]
breed [fish aFish]
```

Instead of *create-turtles*, we can ***create-****sharks* or ***create-****fish*, such as the following statement that creates five sharks:

```
create-sharks 5
```

Quick Review Question 3

a. Declare a *cattle* breed, where the singular form is *cow*.
b. Look up ***set-default-shape***, and read the description. In the *NetLogo User Manual*, whose scrollbar is to the left of the window, under *Features* click *Shape Manager*. Notice that there are default shapes for "fish" and "cow," as well as for a number of other breeds. In *setup*, after *clear-all* declare the default shape for *cattle* to be "cow." Note that we must put the shape, such as "cow," in quotation marks.
c. In *setup*, after the closing bracket for *ask patches*, create one agent of the *cattle* breed, make its color black, and make its *x*-coordinate 5 and its *y*-coordinate 10. Click *Check* and make corrections as necessary. On the Interface level, click *setup* to check the correct cow creation.
d. Look up the ***sprout*** command. What does it do?
e. At the end of *setup*, using the plural form, *cattle*, with *sprout*, ask each patch to create a white cow. Check your work and click *setup* on the Interface level to observe the results.
f. To only have the white cattle on the green grass, we can use the ***with*** command. Look up *with*. Adjust the first line of the *ask patches* command to only ask the green patches to sprout cattle.
g. On the Interface level, create a *go* button but do not check the *Forever* checkbox. Thus, we must click *go* each time we want to execute the function. On the *Code* level, write a *go* function to ask all cattle to move to one of its neighbors with *agentType* equal to "Farm." Check your work.
h. Click *go* several times. Do the white cattle stay on the green grass?
i. Let's make them stay on the green grass by stopping the periodic boundary conditions. Right-click on the world and select *Edit*. What boxes do we uncheck? Click *OK*. Then, click *setup* once and *go* several times to observe the results.

Quick Review Question 4

a. Let's count turtles in the water, that is, on the blue patch. First, look up the ***count*** command, and indicate what it does.
b. Look up ***turtles-here***, ***<breed>-here***. On the Interface level, right-clicking, inspect the blue patch. In the text box at the bottom of the popup inspection

window, type the command to count the number of cattle on this patch using *cattle* instead of *turtles*.

c. The answer appears in the *Command Center* at the bottom of the *NetLogo* window. What prefaces the answer? Click *go*; and in the command area of patch's inspection window, press the up arrow and <ENTER> or <RETURN> to observe a new count.

d. For readability, we first define a **local variable**, *water*, that only appears in the *go* function. Such a local variable cannot be used in another function or on the Interface level. The first time we use a local variable, giving it a value, we employ ***let*** instead of *set*. Look up the ***patch*** command. At the end of *go*, using *let*, assign to *water* the patch with coordinates 0 and -8, the water location.

e. Then, write a command to ask *water* to assign the count of the cattle in its location to a variable *inWater*. In a subsequent part, we will display the value of this variable on the Interface level, so use *set* instead of *let*. Clicking *Check* we get an error, "Nothing named INWATER has been defined," which we rectify next.

f. The problem is we have not indicated that *inWater* is a local variable by using *let* and we have not declared it to be a global variable with *globals*. At the top of the Code level, form such a *globals* command.

g. On the Interface level, add a monitor named "Cattle in Water" to report the value of *inWater*. What do we type in the *Reporter* box? After saving, click *go* several times and observe the changing value in the monitor.

h. In *setup* create a yellow turtle at location (4, 10). This turtle should not be of the *cattle* breed. Moreover, the shape will be that of an arrow. Check your work, and click *setup* and *go* to observe the results.

i. In *go*, define a local variable *arrow* to be this turtle. We need to indicate that the agent is *one-of turtles* with the color *yellow*.

j. In *go*, define a local variable *blackCow* to be the black cow. Again, we must use *one-of*.

Agents and Patches

Quick Review Question 5 We can ask an agent about a patch and move the agent to another patch.

a. On the Interface level, click *setup* several times. Notice that the yellow arrow faces in a random direction. Look up ***face***. In *go*, ask the arrow to face the black cow. After checking your work, click *setup*. Click *go* several times and notice how the arrow turns in the direction of the black cow.

b. Look up ***heading***. What does *heading* indicate?

c. What is the *heading* value for south?

d. Look up ***patch-at-heading-and-distance***. What does *patch-at-heading-and-distance* report?

e. Ask *blackCow* to assign to a local variable, *south*, the patch that is to the south at distance one.

f. Look up ***forward***, ***fd***. What does it do?

g. Look up ***of***. Continuing in the same block, after asking the black cow to assign a value to *south*, have an *if* statement to do the following:

```
if the agentType of the patch to the south is "Farm"
    make black cow face south (i.e., assign to heading the number for south)
    have the cow more forward a distance one
```

Check your work. On the Interface level, click *setup* once and *go* repeatedly. With each click of *go*, the black cow actually moves twice—once because we ask all cattle to move to a neighbor and once because we specifically ask the black cow to move south as long as it is still on the farm.

h. Can the black cow ever move on top of the fence?

i. In this part, we mark with red the locations of the black cow at the end of each execution of *go*. Look up ***patch-here***. After the closing right bracket of the *if* statement but within the block for *ask blackCow*, ask the black cow's current location (the patch here) to turn red (assign a new value to its *pcolor*). Check your work; click *setup* once and *go* several times.

j. Look up ***nobody***. What does it indicate?

k. Let's turn a cow orange if a neighboring patch ever has four cows. Thus, at the end of *go* do the following:

```
ask each cow  (that is, ask cattle)
    assign to local variable nbr one of the neighbors with a count of
            cattle-here being 4
    if nbr is not nobody
        make the cow's color orange
```

Check your work; click *setup* once; and *go* several times. Notice that once turning orange, a cow stays that color.

Mathematical Functions

Quick Review Question 6 NetLogo provides a number of mathematical functions that are useful for models and that appear under the Dictionary's "Mathematical" category.

a. Look up ***round***. On the Interface level, after *observer>* at the bottom of the window, type a command to return the rounded value of 6.2.

b. What is 6.9 rounded?

c. Look up ***sqrt***. Type a command to return the square root of 6.2.

d. Look up ***mod***. Type a command to return the remainder when 31 is divided by 7.

e. Look up ***sum***. Type a command to return the sum of the count of the neighbors of the cattle. A cow is likely to be a neighbor to several other cows. Thus, many cows are will be counted several times.