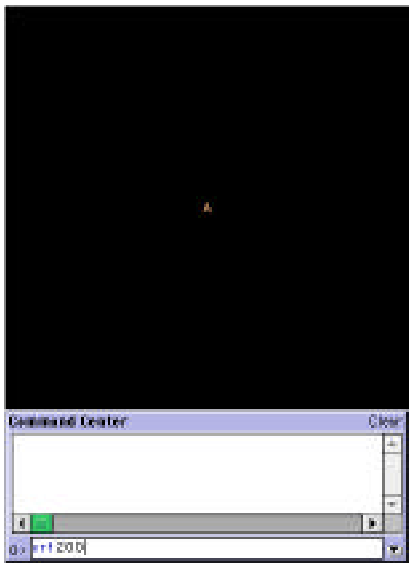
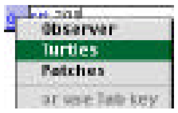
Turtle Epidemic

A NetLogo Simulation Activity

Getting Started



Type `crt 200` into the Command Center to create 200 turtles. They are all sitting on top of one another at coordinates (0,0), and have a variety of different headings and colors.



Click-and-hold on the "O>" in the bottom left corner of the Command Center window, and select **Turtles**, because we want to address the turtles we just created with our next command.

Type `forward random 100` to get the turtles to move forward a random number of steps (from 0 to 99), so we can see them all.

Type `set color lime` to turn all the turtles' colors to (uhhh…) lime.

Type `set heading random 360` to randomize their headings.

We want to make one turtle sick. For starters we can do this with color. Type `if who = 1 [set color red]` and you should see one of the turtles turn red. (**Note**: there must be a space on either side of the "=" sign!)

This is pretty close to what we need for our `SETUP` procedure, so select the list of commands in the upper pane of the Command Center window and choose **Copy** from the **Edit** menu, then click on the **Procedures** tab near the top of the NetLogo window. In the Procedures area, you can delete the text that reads "`; add model procedures here`", and paste the contents of the clipboard. Delete the "O>" in front of the first command, and after deleting all the occurrences of "T>" place the turtle commands inside a construction that looks like this: `ask turtles [ ]`, with our turtle commands sitting inside the square brackets. This construction can span several lines, and for clarity it is a good idea to indent using the tab key, so that you end up with something like this:

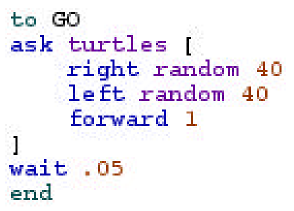
```
to SETUP
crt 200
ask turtles [
    fd random 100
    set color lime
    set heading random 360
    if who = 1 [set color red]
]
end
```

Click the **Interface** tab and type `setup` into the Command Center to test your procedure. It should work fine except for one thing; you now have 400 turtles! Probably the first thing your `SETUP` procedure should do is to "clear all". You can do this in the Command Center by typing `ca`, and you should insert it as the *first* line of your `SETUP` procedure. And so you don't have to type `setup` all the time, create a button by clicking on the Button icon in the toolbar, then clicking in the white area in the upper left area of the NetLogo window. Type `SETUP` in the box labeled "Code" in the Button Properties dialog, then click OK.

This might be a good time to make your turtles' world a bit bigger. Control-click on the black graphics area and choose **Edit…** from the popup menu. You ought to be able to set the Screen Edge values to 25 and the Patch Size value to 12 pixels. After you do this, you will probably need to move the Command Center out of the way. Command-click on it, choose **Select** from the popup, and drag it toward the lower left quadrant of the NetLogo window. Click your `SETUP` button to see how things look now.

Making Things Go (and Stop)

Next we want to define a procedure that our model can *iterate* over and over. Start by typing the following into the Procedures area:



```
to GO
ask turtles [
    right random 40
    left random 40
    forward 1
]
wait .05
end
```

This procedure asks the turtles to wiggle a little bit, then move forward one step. The `wait` statement at the end of the procedure is there to slow things down enough for us to be able to see what is going on.

Create a `GO` button similar to the `SETUP` button, except for this one, click in the ☑ Forever check box. This is what makes the procedure called by the button *iterate*. Then click your `GO` button and watch the action; clicking a forever-button a second time stops it.

For a red turtle to "infect" a turtle that crosses its path, we can use NetLogo's rich set of idioms to construct an expression like the following:

```
if color = red [set color-of random-one-of turtles-here red]
```

Type this command underneath the `forward 1` command in the `GO` procedure, then try your `GO` button out.

We'd like to see a plot of the spread of this turtle epidemic over time. Click on the icon in the toolbar, then click in the white space below your `SETUP` and `GO` buttons to create a plot. In the Plot Properties dialog, name the plot, label the x-axis "TIME" and the y-axis "Turtles", and set the plot pen name to "Sickies" and the plot pen color to red.

What do we want to plot? The number of red turtles. The NetLogo idiom for this is

```
plot (count turtles with [color = red])
```

We'd also like to see the model stop running once all the turtles have gotten sick. Here's the command:

```
if (count turtles with [color = red]) = 200 [stop]
```

Type these two lines at the bottom of your `GO` procedure (below `WAIT`) and check out the results. You should see the number of infected turtles grow according to an "ogive" (S-shaped) function of time.

You may notice that the plot auto-scales. This is fine for the TIME dimension but we don't need it for the Turtles dimension, since we already know the number of turtles. Add the following statement to your `SETUP` procedure just before the `END` statement:

```
set-plot-y-range 0 200
```

To be sure, this could have been set in the Plot Properties dialog, but doing it under program control will give us more flexibility later on.

Getting Experiment-Ready: Turning Constants into Variables

We would like to know the effect of different variables on how long it takes for the epidemic to spread through the turtle population, with our main dependent variable being the number of model "ticks" before all the turtles have turned red. Since it is hard to read an exact value off the graph, we can use a Monitor for this. First we need to define a "Global variable" for the Monitor to, well, monitor. Get into the Procedures area, and right at the top, type `globals [CLOCK]` (we could have used `TIME` here instead, to match the Plot; the name ultimately doesn't matter). Then back in the Interface area, click on the icon in the toolbar, click in some empty white space, and type `CLOCK` in the text box labeled "Reporter"; set the number of decimal places to zero.

Add a `set CLOCK 0` statement outside the `ask turtles` block in `SETUP`, and add a `set CLOCK CLOCK + 1` statement outside the `ask turtles` block in `GO`. (**Note**: The "+" sign in NetLogo, like the "=" sign, must be surrounded by spaces in order to be understood properly by the program.)

Now you are set to collect precise data, because your `CLOCK` Monitor displays the exact number of model ticks elapsed when the whole turtle population "goes red." Because there will be some variability in this number from one "run" of the model to the next, you will want to collect several observations and average them to obtain a reasonable idea of the true value.

The next questions you should ask are: How robust is this value? What factors does it depend on? In what follows I provide you with some simple ways to augment your model and address some questions that might come to mind, using the basic NetLogo mechanism for turning constants into variables: the *slider*.

Population size. This one is relatively easy to implement. Create a slider by clicking on the icon in the toolbar, type `POPULATION` in the variable box, and values like 100, 1000, and 200 in the min, max, and current value boxes. Then in the Procedures area, you basically need to change all of the places where “200” appears to `POPULATION`. For the record, here they are:

```
crt POPULATION
set-plot-y-range 0 POPULATION                                (in SETUP)
if (count turtles with [color = red]) = POPULATION [stop]    (in GO)
```

Initial infection. Does the speed with which the infection spreads through the population depend on how many turtles were infected initially? Set up a slider called `INITIAL-SICK` with values like 1 and 20 for minimum and maximum values. The only statement you need to change is in the `SETUP` procedure:

```
if who < INITIAL-SICK [set color red]
```

This works because turtle ID’s start at 0; so (e.g.) if `INITIAL-SICK` is 3, turtles 0, 1, and 2 are all colored red.

Infection radius. In the base model,turtles have to be right “on top of” one another for the infection to be transmitted. With a slight variation of the code, we can generalize the model to address the possibility of infection over more realistic distances. Again, start by creating the slider; call it `COOTIE-RADIUS` and set its min and max to something like 0 and 5. For more fine-grained control, set its increment to “.1” instead of the default value of “1”.

We use a new NetLogo idiom, `in-radius`, to take advantage of our new variable, as follows:

```
if color = red [set color-of random-one-of turtles in-radius COOTIE-RADIUS red]
```

In principle, setting `COOTIE-RADIUS` to 0 ought to give us essentially identical results to the original model using `random-one-of turtles-here`. But the hour is late as I write this, and I’m afraid to check.

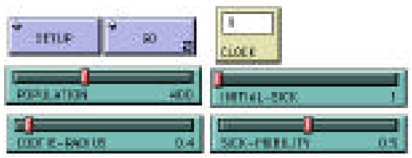
Turtle mobility. There are (at least) two ways to attack this one: mobility of well turtles and mobility of sick turtles. I do the latter here, but you are welcome to try your hand at the former (or at any other variable/variation that strikes your fancy).

The basic slider we’ll call `SICK-MOBILITY` and let it vary from 0 to 1 (increment .1, obviously). To implement it in code, change the `forward 1` statement in the `GO` procedure as follows:

```
ifelse color = red [forward SICK-MOBILITY] [forward 1]
```

The `ifelse` statement is like the `if` statement except that it takes two sets of commands in brackets instead of one; the first gets executed if the expression following the `ifelse` is true, the second if it is false. Again, here we can recreate the original model as a special case where `SICK-MOBILITY = 1`.

Pictured on the right is one possible layout for your interface, with buttons, sliders, and monitor arranged in a reasonably orderly fashion. The plot (not shown) is below all this paraphenalia (sp?).



POPULATION 400
INITIAL-SICK 1
COOTIE-RADIUS 0.4
SICK-MOBILITY 0.5

And Finally … The Experiment!

Choose *two* variables and *two* values for each variable, so that the data you collect organizes into a 2x2 matrix. Before collecting the data, formulate a hypothesis and a prediction about the effect of each variable. Given your experience running the “base model”, try to make these predictions as quantitative as possible. Predict also whether you think the combined effects of the two variables will be *additive* or not, i.e. whether the effect of each variable will be the same size, or larger, or smaller, at each level of the other variable. In collecting your data, perform at least 12-15 “runs” per cell of your matrix, and compute averages.

Do a brief writeup of your experiment, your experience with NetLogo, your predictions, your thoughts about the model and interesting ways to develop it further, While you’re at it, compare your NetLogo experience with your high school “science lab” experience(s), and reflect on the differences.