Math 140 Basic Forest Fire Model

> *Modeling is problem solving under constraints. We know that the model we use is not perfect, but—facing the constraints of information and time—we also know that even a simple model will probably lead to a better solution to the problem than not using any model at all.*
>
> Grimm and Railsback

> *All models are wrong, but some are useful.*
>
> George Box

Basic NetLogo code to get started[1]

Let's step through building a simple forest fire model in order to learn more about NetLogo, then you can continue developing the model to have more features.

If you want more information about a command or how to do something in NetLogo, the user manual is an excellent resource: http://ccl.northwestern.edu/netlogo/docs/, including the tutorials, the programming guide, and the NetLogo dictionary. There are also lots of sample models in the Models Library, and the Code Examples section is helpful for demonstrating how to code particular things.

Please work through the following steps to build a basic forest fire model, after which you can start implementing your own ideas for an extended model.

1. First create **setup** and **go** buttons on the interface (go should be a "forever" button), as well as a slider named **density** running from 0 to 100% (giving the initial density of tree-filled patches in the forest).

2. Hit the **settings** button, and set max-pxcor and max-pycor to 125 (or double this if you want to model a bigger area). Make sure the boxes for "world wraps" are not checked, and set patch size to 2 pixels (if you use a setting of 250, then 1 pixel).

3. We want to compare how many patches we start with to how many get burned, so we create two global variables (double semi-colons mark comments):

```
globals [
 initial-patches     ;; how many green patches we started with
 burned-patches      ;; how many have burned so far
]
```

4. There are different ways to track which patches are currently burning or have been burnt. In this model, we will track two types of turtles: *fires* (when first start burning) and *embers* (been burning for a while). A **breed** is a type of turtle, where plural then singular terms are listed:

---

[1] Code is by Uri Wilensky, ©1997, http://ccl.northwestern.edu/netlogo/models/Fire

```
breed [fires fire]          ;; bright red turtles -- the leading edge of the fire
breed [embers ember]  ;; turtles gradually fading from red to near black
```

Here we're making a choice to track which patches are on fire by placing a fire-agent on those patches, rather than have the patch itself keep track of whether it is on fire. This choice can be more efficient, as we can directly call on fire-agents to implement the fire spreading, rather than finding which subset of patches is on fire at each time step. This approach also allows us to have embers as another type of agent involved in the fire's progress.

5. Now we're ready to define a **setup** procedure. This picks a random number between 0 and 100 for each patch, and makes it a tree-filled patch if the number is less than the density on the slider bar. The percentage of patches that have trees will then be very close to the desired density. Then a fire is started on the patch at the origin (middle of the forest), and the variables initial-patches and burned-patches are given appropriate values (note that no equal signs are used in logo to set values). We will define the ignite procedure in a bit.

```
to setup
 clear-all
 set-default-shape turtles "square"
 ask patches with [(random-float 100) < density]
   [ set pcolor green ]                                  ;; make some green patches
 ask patches with [pxcor = 0 and pycor=0]
   [ ignite ]                                            ;; start the fire in middle
 set initial-patches count patches with [pcolor = green]    ;; set patch counts
 set burned-patches 0
 reset-ticks
end
```

6. Next we need to define the **go** procedure. We can find neighboring patches that still have trees (are green) using neighbors4 (set of patches above, below, to right, and to left of current patch) and let the fire spread to those patches. We also change the fire turtles to be ember turtles (assume fire is starting to go out in that spot, as tree is nearly burned).

```
to go
 if not any? turtles    ;; if no fires or embers left, fire is out
  [ stop ]
 ask fires
  [ ask neighbors4 with [pcolor = green]
     [ ignite ]
    set breed embers ]
 fade-embers
 tick
end
```

7. For the fire to spread, we need to define the **ignite** procedure. The command *sprout-<breed> number* generates new turtle(s) of that breed, in this case, one new bit of fire. Since this means the tree on the current patch is now on fire, we increment the burned-patches variable to be one higher.

```
to ignite  ;; patch procedure that creates the fire turtles
 sprout-fires 1
  [ set color red ]
 set pcolor black
 set burned- patches burned- patches + 1
end
```

8. The last procedure makes a nice visual effect of the ember gradually going from red to black as the tree finishes burning. The *die* command eliminates the current turtle.

```
to fade-embers      ;; achieve fading color effect for the fire as it burns
 ask embers
  [ set color color - 0.3  ;; make red darker
   if color < red - 3.5     ;; are we almost at black?
    [ set pcolor color
     die ] ]
end
```

9. Verify that your code seems to be working by running the model with 100% density. Does it behave like you would predict it should?

10. Add a graph to the interface displaying the number of patches burned over time.

11. Once you get this model working, experiment a bit with different parameters. Another interesting thing to try is to replace *neighbors4* with *neighbors* (set of the 8 patches that surround the current patch). Try out both choices to see what a difference a modeling choice like this can make! You can also use *patches in-radius 3*, for example, to choose all patches a distance of 3 or less from the current patch.

12. Run multiple simulations with the same parameter value to see how much the results can vary due to random effects.

By class on Friday, email me (tleise@amherst.edu) a brief report on how the density parameter affects the total area (in terms of number of patches or proportion of patches) that is burned in a fire. Be specific. For example, you could test three different values of the density (ideally that yield different qualitative results), carrying out 10 simulations for each density, then report the mean value of burned area as well as the range of burned area values.

This gives some idea of the average effect of density as well as the variability in how far the fire spreads.