Introduction to Programming in NetLogo

Anton E. Weisstein, Truman State University

I. Background on Agent-Based Models

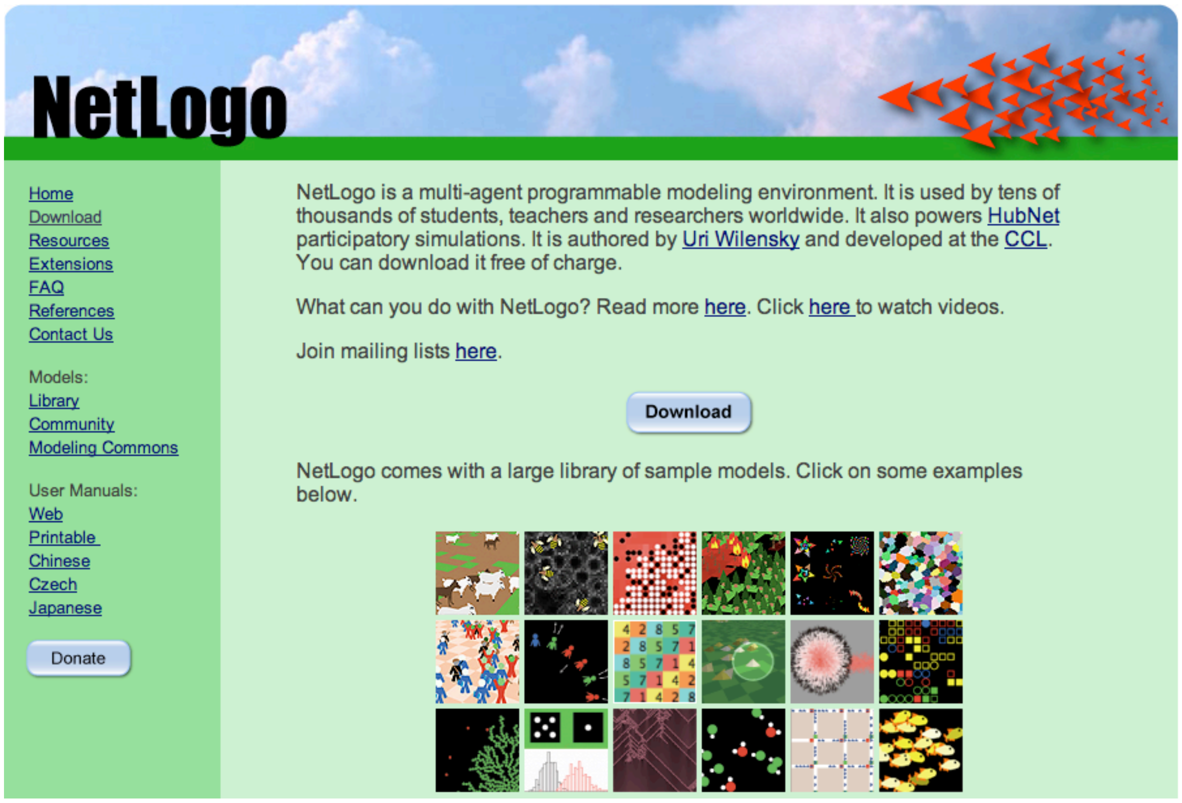
An ***agent-based model*** is a simulation in which individual objects, or ***agents***, move about and interact with each other and their environment through specified rules. The user can fine-tune these rules (e.g., by altering agents' movement speed) or make larger changes (e.g., by creating new kinds of interaction).

In NetLogo, the simulated world is divided into distinct ***patches***. Each patch can be given its own appearance, properties, and rules; however, the boundaries between patches are usually invisible. Similarly, a single model may contain multiple kinds, or ***breeds***, of agent, each with its own appearance and behavioral rules.

In most models, agents and patches interact only with other nearby agents and patches. Larger-scale patterns and behaviors then emerge from these simple, localized interactions.

II. Downloading NetLogo and Getting Started

NetLogo can be downloaded free of charge from http://ccl.northwestern.edu/netlogo/. The website's Download button will take you to an optional user information form and then to the actual download page. The download includes tutorials, examples, and pre-built models as well as the main NetLogo software package. Versions of NetLogo are available for Mac OS X, Windows, and Linux (but not currently for phones or tablets).



NetLogo is a multi-agent programmable modeling environment. It is used by tens of thousands of students, teachers and researchers worldwide. It also powers HubNet participatory simulations. It is authored by Uri Wilensky and developed at the CCL. You can download it free of charge.

What can you do with NetLogo? Read more here. Click here to watch videos.

Join mailing lists here.

NetLogo comes with a large library of sample models. Click on some examples below.

Following installation, we recommend that you drag the NetLogo folder from your computer's desktop to your Applications folder. Then open NetLogo by double-clicking its Desktop icon. Open the Models Library from the File menu. Choose "Flocking" from the Biology section, and click "Open."

III. Interface and Controls

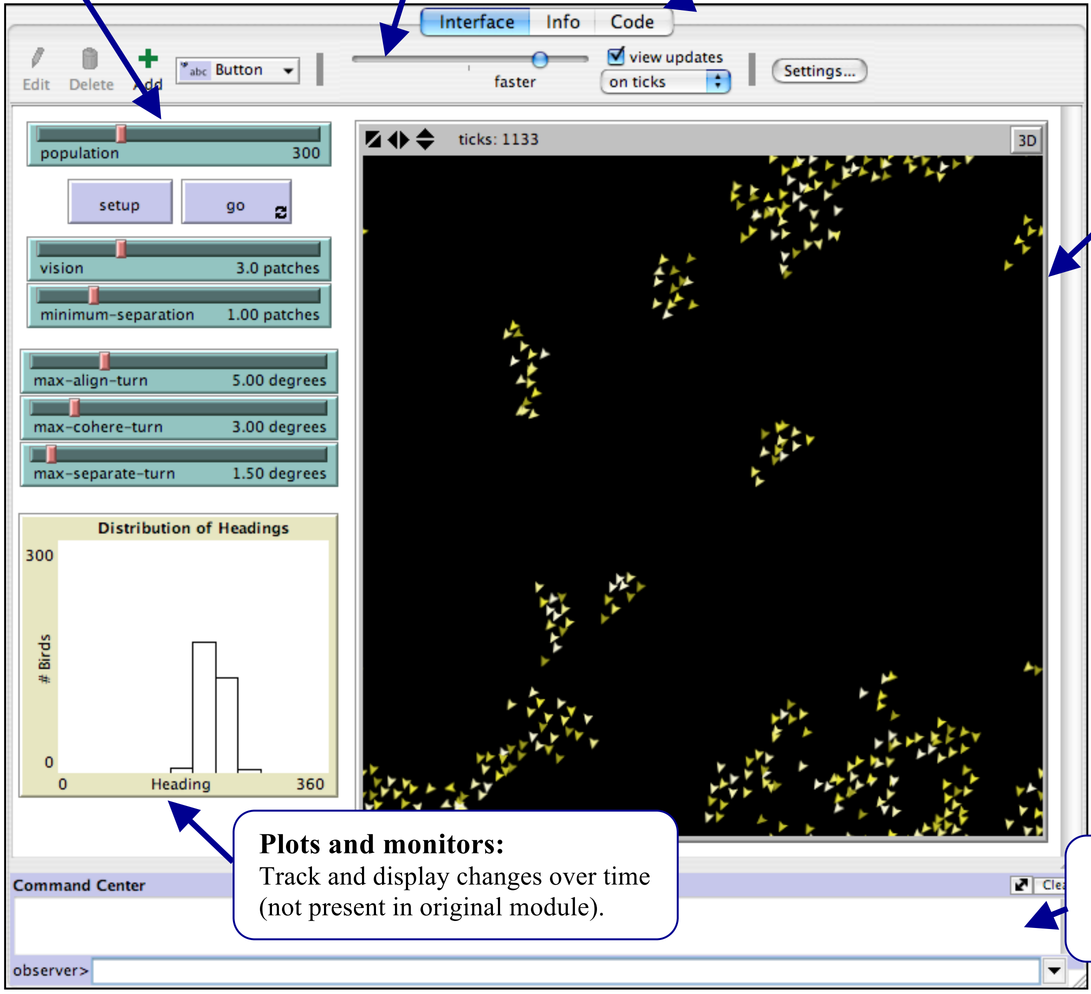
In NetLogo, the **Interface** tab lets you interact with and display the model. To begin, click the **setup** button and then click **go**. Individual agents should start moving about their simulated world and interacting with each other. A few notes:

- The model will continue to run as long as the **go** button is selected. Click **go** again to pause or resume the model; click **setup** to restart the model.
- Use the speed slider to speed up or slow down the model.
- Use the green slider controls to adjust model parameters. Most parameters can be altered even while the model is running; however, changing the value of *population* has no effect until the model is restarted.
- Many NetLogo models include one or more plots of key variables (e.g., frequency distribution of bird headings, average flock size vs. time). While the original Flocking model lacks any such graphical output, NetLogo allows the user to add such features. For more details, see Section VII (Extensions) of this guide.

Controls and settings:
Allow the user to initialize, begin, and stop the model, and to adjust model parameters.

Speed slider:
Controls how fast the model runs.

Display control:
Switches between NetLogo's three tabs. Use the Interface tab to *display and control* the model, Info to *describe* the model, and Code to *create or edit* the model.



ticks: 1133

population 300
vision 3.0 patches
minimum-separation 1.00 patches
max-align-turn 5.00 degrees
max-cohere-turn 3.00 degrees
max-separate-turn 1.50 degrees

World window:
Shows the current state of the simulated world, including each agent's location.

Plots and monitors:
Track and display changes over time (not present in original module).

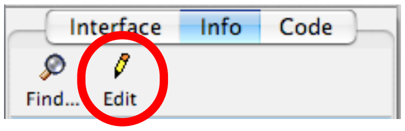
Command center:
Allows user to enter text commands directly.

IV. The Info Tab

In the Display Control (see p. 2), click the **Info** tab. This will display a text file describing many important aspects of the chosen model, including:

- a general description of the biological system
- a summary of the rules governing agents' behavior
- a description of the model's main controls and parameters
- suggested explorations and extensions of the model

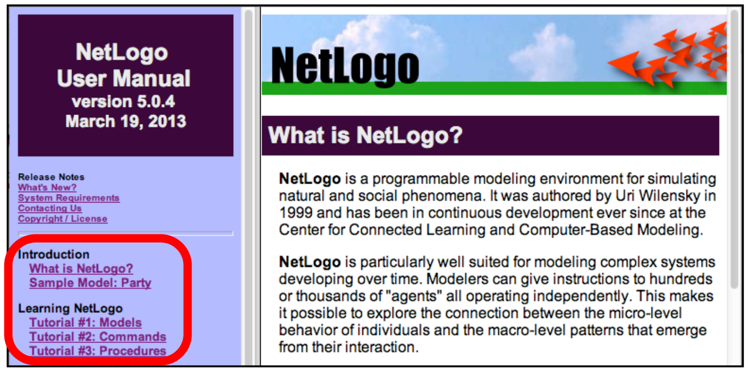
This information makes the **Info** tab a logical place to start when introducing a NetLogo model. Alternatively, you can use the **Edit** tool (see right) to remove information that you want students to figure out on their own. Edited versions of any model should be given a new name using the **Save As** command from the File menu so that you also retain the original version.



V. Code Tab: Programming in NetLogo

The **Code** tab contains the series of commands that determine how the model actually runs. This section defines the different kinds of agents and their properties, sets up the simulated world in which agents "live", and lays out agents' behavioral rules. Understanding and modifying this code allows the user to adapt the model to a much broader range of biological systems. However, you can also run the standard model and analyze its behavior without ever needing to look at the underlying code.

If you do want to understand how to code in NetLogo, the built-in User Manual under the Help menu is a great place to start. In particular, working step-by-step through the introduction and first three tutorials (see right) will help familiarize you with the programs menus, interface, commands, and possible classroom uses. This process will likely take approximately 3 hours (working at a moderate pace) but is well worth it.



What is NetLogo?

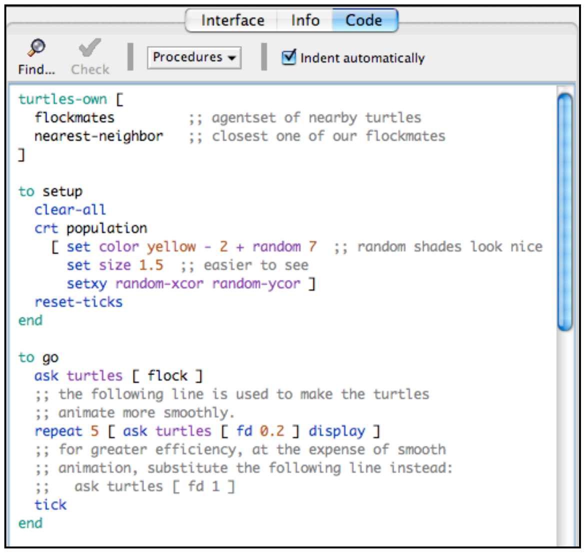
NetLogo is a programmable modeling environment for simulating natural and social phenomena. It was authored by Uri Wilensky in 1999 and has been in continuous development ever since at the Center for Connected Learning and Computer-Based Modeling.

NetLogo is particularly well suited for modeling complex systems developing over time. Modelers can give instructions to hundreds or thousands of "agents" all operating independently. This makes it possible to explore the connection between the micro-level behavior of individuals and the macro-level patterns that emerge from their interaction.

Some general notes on NetLogo code:

- The code is organized into ***blocks*** that define individual routines. For example, the turtles-own block defines key properties of each individual agent (for historical reasons, NetLogo agents are called "turtles"). Similarly, the to setup block lists the commands to run whenever the user clicks the **setup** button.

- NetLogo uses color-coding to indicate different parts of the computer code. For example, each code block begins with the command to and ends (appropriately enough) with end: both of these commands are indicated in green. Other commands and mathematical constants are marked in blue, purple, or red. In addition, any text that follows a semicolon is a ***comment*** that is not actually part of the code itself but is simply a helpful note for the user. Comments are automatically colored grey.



```
turtles-own [
  flockmates         ;; agentset of nearby turtles
  nearest-neighbor   ;; closest one of our flockmates
]

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
end

to go
  ask turtles [ flock ]
  ;; the following line is used to make the turtles
  ;; animate more smoothly.
  repeat 5 [ ask turtles [ fd 0.2 ] display ]
  ;; for greater efficiency, at the expense of smooth
  ;; animation, substitute the following line instead:
  ;;   ask turtles [ fd 1 ]
  tick
end
```

• When you first look at NetLogo code, the unfamiliar terminology and formatting may seem daunting. However, with some practice, it becomes fairly easy to deduce the general meaning of many commands. As an exercise, try to figure out what the following commands from the Flocking module might mean:

1. `setxy random-xcor random y-cor` *(Hint: This command is part of the* `to setup` *block that occurs each time the user resets the model.)*
2. `repeat 5 [ ask turtles [ fd 0.2 ] display ]` *(Hint: What might "fd" stand for?)*
3. `set flockmates other turtles in-radius vision`

Remember that you can always use NetLogo's built-in manual and dictionary if there's something you still don't understand.

VI. Possible Explorations

The following set of explorations is intended to help students build from a basic understanding of self-organizing systems to an open-ended research-like project.

Exploration 1. Run the model for approximately 1000 time steps ("ticks"). Describe as clearly as possible how the birds change their flight direction over time. What factors directly influence each individual bird's flight direction?

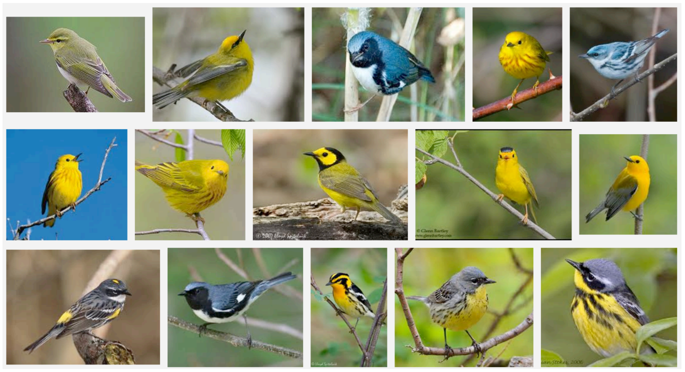
Suggested answer: *Each individual bird tends to align its flight direction with those of its neighbors. Over time, this results in the entire population flying in the same general direction.*

Exploration 2. Reset the model and run it again. Do the birds end up flying in the same direction as before? Why or why not?

Suggested answer: *No. Birds' initial headings are determined randomly, so some headings will be slightly more common than others. As each bird aligns with its neighbors, those headings will become more and more common, eventually becoming the overall direction of the entire population.*

Exploration 3. How many flocks do you see after 1000 time steps? Is this number stable, or does it change as the model continues to run? **Note:** You will have to decide how to determine where one flock ends and another begins — make sure that you can explain your reasoning.

Suggested answer: *Answers will vary depending on how students interpret the idea of a "flock." For example, some students may decide that a large group of birds represents a single flock, whereas other students might say that small gaps within that group justify dividing it into three smaller flocks. Some students may focus only on the birds' positions, while others might also consider their relative headings. This can motivate a broader discussion about classification. For example, how many distinct species do the 15 warblers shown at right represent? (Screenshot of results from Google Image search for "warbler".)*



Exploration 4. What special properties, if any, distinguish flock "leaders" from other birds in the flock? (To help answer this question, Control-click on a single bird at the head of a flock to bring up a menu of options. Choose the last option ("turtle *i*"), then select "watch turtle *i*". This will create a circle around that bird, letting you track its movements more easily. To remove the circle, Control-click on the same bird and choose reset-perspective.)

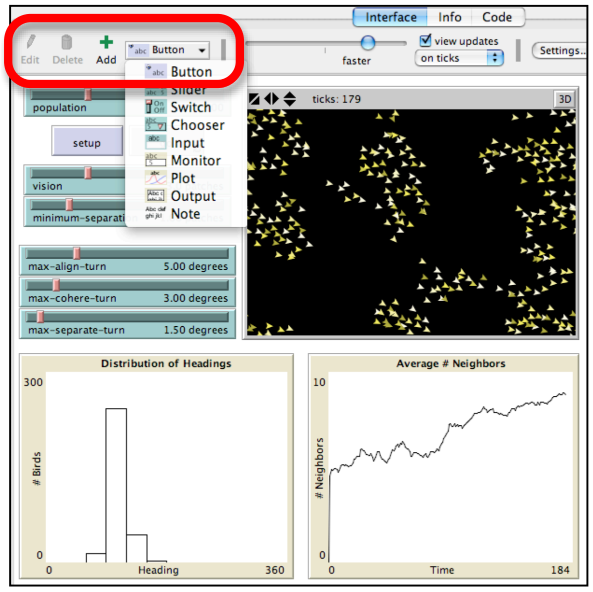
Suggested answer: *There is no intrinsic difference between flock leaders and other birds. As the simulation runs, flocks will split and merge, with different birds emerging as (temporary) leaders. Each bird's position within the flock emerges purely from local interactions between interchangeable agents, a defining property of self-organizing systems.*

Exploration 5. Working in pairs, choose some aspect of the system to study further. For example, you might explore the effect of a particular variable on the birds' behavior, or analyze the change in some property over time. Write down the question you have chosen to research, then take 15-20 minutes to study that question using the model. Clearly record your procedures, results, and inferences. Then share your findings with the entire class as a one-minute presentation.

Suggested answer: *Results will depend on the questions chosen. However, each group should produce a clearly stated question, an appropriate experimental design, and a general statement of their results. For example, students exploring how visual range affects flock size might run the model for values of* vision *ranging from 0 to 10 patches. They might then report that no flocking occurs when* vision *$\leq$ 1.0 patches, but that flock size rapidly increases thereafter, eventually producing a single large flock when* vision *$\geq$ approximately 8 patches. They might also note that the simulation runs more slowly for larger values of* vision *(because the model must make more calculations as each bird sees more and more of its neighbors).*

VII. Extensions

Once you have worked with the Flocking model for a little while, you may want to customize the display, add more complex and realistic behaviors, or explore other models of self-organizing systems. NetLogo's built-in tutorials, user manual, and code examples provide many resources to help you do so. A few particularly useful features:



- You can add plots and controls using the toolbar on the **Interface** tab (see screenshot at right). For example, add a switch to turn a particular property on or off, or put in a monitor to output the value of a specific variable within the model. Plots can be customized to display frequency histograms or to track one or more numeric variables over time, using pre-built code examples from the Models Library (Histogram Example and Tutorial 3, respectively).

- The Code Examples section of the Models Library also contains pre-built examples of specific agent behaviors that may be useful when modeling particular biological systems. You can copy and paste this code into your own model, then modify it as needed. For example, when an agent in the Flocking model moves past the right-hand edge of the simulated world, it reappears on the left-hand edge (i.e., the edges "wrap around"). If you instead want agents in your model to bounce like billiard balls when they hit the edges, use the code from Code Examples > Bounce Example. You can make agents look ahead before moving (Look Ahead), define different kinds of agents (Breed and Shapes), and bind together any agents that meet (Mobile Aggregation), as well as many other behaviors. In general, if there's a particular behavior you want in your model, there's a good chance that one of the pre-built models already has the code you need.

• Because bird flocking is so familiar, it serves as a good starting point from which to introduce the concept of self-organization. However, self-organizing systems are not restricted to the organismal scale, or even to biology. The following NetLogo models demonstrate self-organization in a wide variety of settings:

- Biology > Ants; Fireflies; Fur; Heatbugs; Membrane Formation; Moths; Slime; Termites
- Chemistry & Physics > Crystallization > Crystallization Basic
- Chemistry & Physics > Diffusion Limited Aggregation > DLA Simple
- Chemistry & Physics > Polymer Dynamics
- Computer Science > Cellular Automata > Life
- Earth Science > Erosion; Fire
- Mathematics > Voronoi – Emergent
- Networks > Giant Component; Team Assembly
- Social Science > Segregation; Traffic Basic

To help communicate the shared features of self-organization, consider having each pair of students choose one of the above modules, read about it (through the **Info** tab and/or external resources), explore the model's behavior as described in Exploration 5 (p. 5), and share their findings with the class through a poster session or series of short presentations.