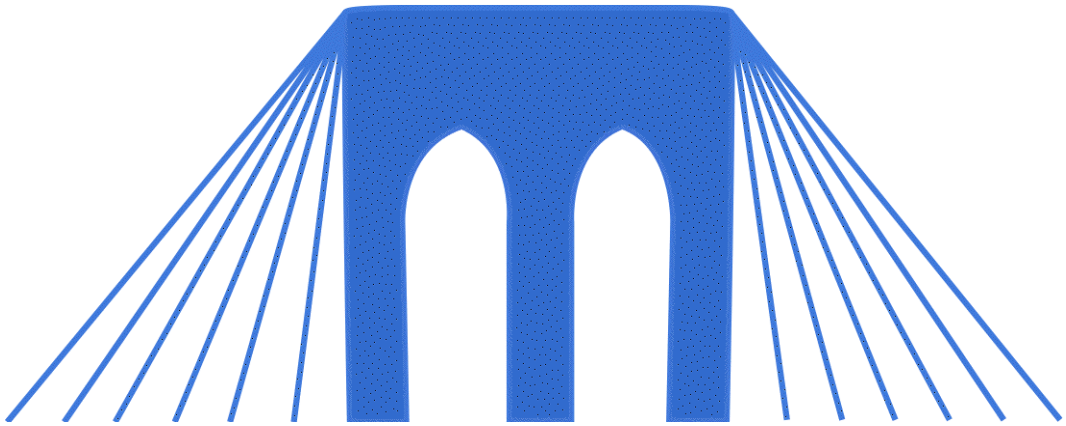
Bridges To Computing



General Information:

- This document was created for use in the "Bridges to Computing" project of Brooklyn College.
- You are invited and encouraged to use this presentation to promote computer science education in the U.S. and around the world.
- For more information about the Bridges Program, please visit our website at: http://bridges.brooklyn.cuny.edu/

Disclaimers:

- **IMAGES**: All images in this presentation were created by our Bridges to Computing staff or were found online through open access media sites and are used under the Creative Commons Attribution-Share Alike 3.0 License. If you believe an image in this presentation is in fact copyrighted material, never intended for creative commons use, please contact us at http://bridges.brooklyn.cuny.edu/ so that we can remove it from this presentation.
- **LINKS**: This document may include links to sites and documents outside of the "Bridges to Computing" domain. The Bridges Program cannot be held responsible for the content of 3rd party sources and sites.

Agent Based Programming & Simulations

Lecture 1: Agents & NetLogo

References

- Russell, Stuart J.; Norvig, Peter (2003), Artificial Intelligence: A Modern Approach (2nd ed.), Upper Saddle River, New Jersey: Prentice Hall, ISBN 0-13-790395-2, http://aima.cs.berkeley.edu/ , chpt. 2
- Stan Franklin and Art Graesser (1996); Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents; Proceedings of the Third International Workshop on Agent Theories, Architectures, and Languages, Springer-Verlag, 1996
- An Introduction to Multiagent Systems, by Michael Wooldridge, John Wiley & Sons, Ltd (2002).

Content

- Agent-Based Programming Paradigm
  - Object vs. Agents
  - Agents Defined
  - Agent Based Programming
  - Motivations
  - Computer Agents
  - Components of a Computer Agent
- Introduction to NetLogo
  - Background
  - Variables & Procedures (functions)
  - Agents (types)
  - **ask** and **show**
  - Sensors and Effectors

Objects vs. Agents

- In the Object-Oriented programming paradigm an 'object' is a data-structure (container) created from a class (a pattern or template).
- An object will have both **facts** (properties, attributes, member variables, data members, fields) and **functions** (methods).
- Agents in a computer program are similar to objects (and indeed may be created from classes) but are more powerful than simple objects.
- When we talk about agents, we refer to their "behaviors".

"Agents" Defined

- Agent is derived from the Latin agere (to do).
- What an "agent" is depends on context:
  - Real estate agent, Secret agent, Chemical agent.
- The common component in these cases is that an "agent" is something that "acts" autonomously (independently) usually on behalf of some other entity.
- In order to be autonomous, an agent needs to be able to interact with it's environment. Specifically it needs to:
  1. Sense it's environment.
  2. Reason about its environment (can be very simple if->then)
  3. Affect (act upon) it's environment.

Agent-Based Programming

- The term "agent-based programming" describes a paradigm (a structured approach to solving a problem) similar to object-oriented programming.
- The term "agent" provides a convenient and powerful way to describe an entity that is capable of acting (within an environment) autonomously in order to accomplish a goal .
- Four key notions distinguish agents from arbitrary programs:
  1. Persistence (track events over time)
  2. Autonomy
  3. Reaction to the environment
  4. Goal-orientation (goal driven behaviors)

Motivations

- Why bother with "agent-based" programming?
- Five trends in the history of computing have led to the pursuit of agents and the development of agent-based and multi-agent systems:
  1. Ubiquity
  2. Interconnectivity
  3. Intelligence
  4. Delegation
  5. Human-orientation

Motivations (cont)

1. Ubiquity
   - Computer are everywhere and writing "billions" of specific applications for each possible situation would be impossible.
   - Agent-based programming lets us concentrate on the "behaviors" we want for a specific device.
2. Interconnectivity
   - Many computational devices connect to (a wide range) of other types of devices in order to perform tasks collaboratively (distributed, concurrent and parallel systems).
   - How do we design large systems that can be executed on/over multiple processors/computers?
   - Can we conceive of a system as a series of interacting devices (agents) rather then as a single system?

Motivations (cont)

3. Intelligence
   - Systems that can react to "unpredictable" events in its environment can be thought of as intelligent.
4. Delegation
   - Giving control to a computer is called "delegation" (traffic lights, power grids/plants).
   - Some tasks require ***autonomous interactivity with the environment*** (if we are going to delegate them).
   - Are you comfortable delegating:
     - Tasks that involve safety (flying, driving, nuclear power)
     - Spending money (bidding on eBay, stocks, bills, groceries)
     - Combat operations (robot warriors)

Motivations (cont)

5. Human-orientation
   - Most computer systems are designed to interact with humans.
   - Even if a human delegates tasks to a computer, there is still interaction before and after the task is completed.
   - How do we design systems that can interface effectively with a wide range of human users?
   - What kinds of human features do we need to construct/model in a system in order to increase its usefulness?
     - language?
     - emotion?
     - reasoning?
     - decision making?

Putting it all together…

- We come up with a desire for computational devices that can interact with their environment and with each other to complete tasks.
- By that, we mean that they (the agents) might:
  - Compete -> perhaps for scarce resources.
  - Cooperate -> perhaps to accomplish tasks together that one cannot do alone.
  - Negotiate -> perhaps to manage the sharing or trading of resources.

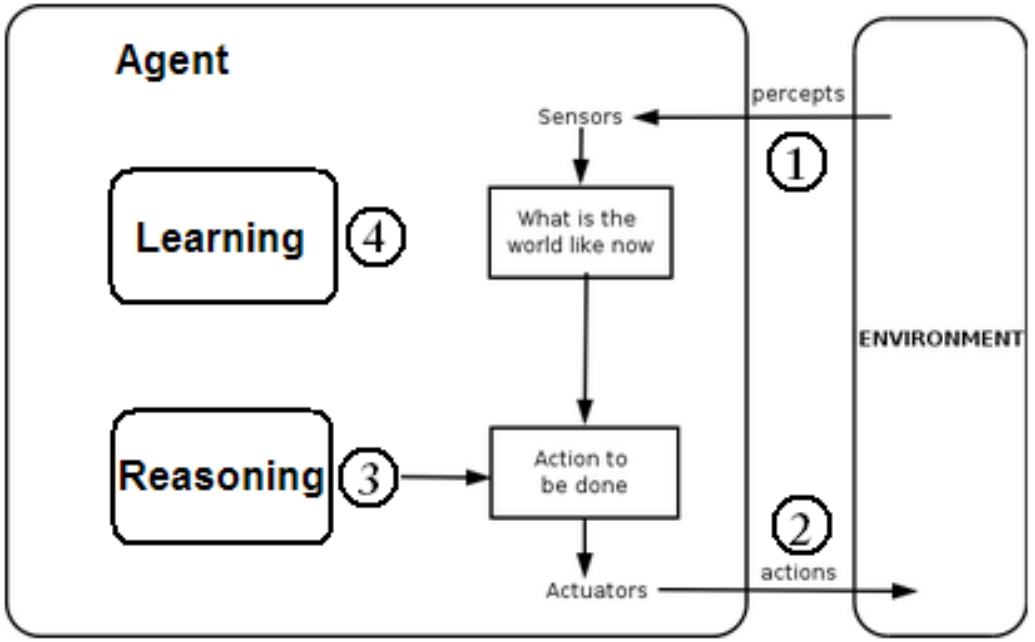
"Computer Agent" Definition

- "An agent is a computer system that is capable of independent (autonomous) action on behalf of its user or owner (figuring out what needs to be done to satisfy design objectives, rather than constantly being told what to do in detail)." (Wooldridge)
- An Agent "is an autonomous entity which observes and acts upon an environment and directs its activity towards achieving goals." Russell & Norvig 2003,
- Key Concepts:
  - Autonomous
  - Interactivity with environment
  - Goal driven (behaviors)

Computer Agent - Disambiguation

- Unlike arbitrary programs, agents:
  - react to their environment
  - are autonomous,
  - exhibit goal-orientation
  - exhibit persistence.
- Unlike objects which are defined in terms of methods and attributes, an agent is defined in terms of its behavior.
- NOTE: A "software agent" usually implies a program, a "computer agent" can be a hardware/software combination, and an "intelligent agent", can be anything (including people/groups of people).

Components of an Agent



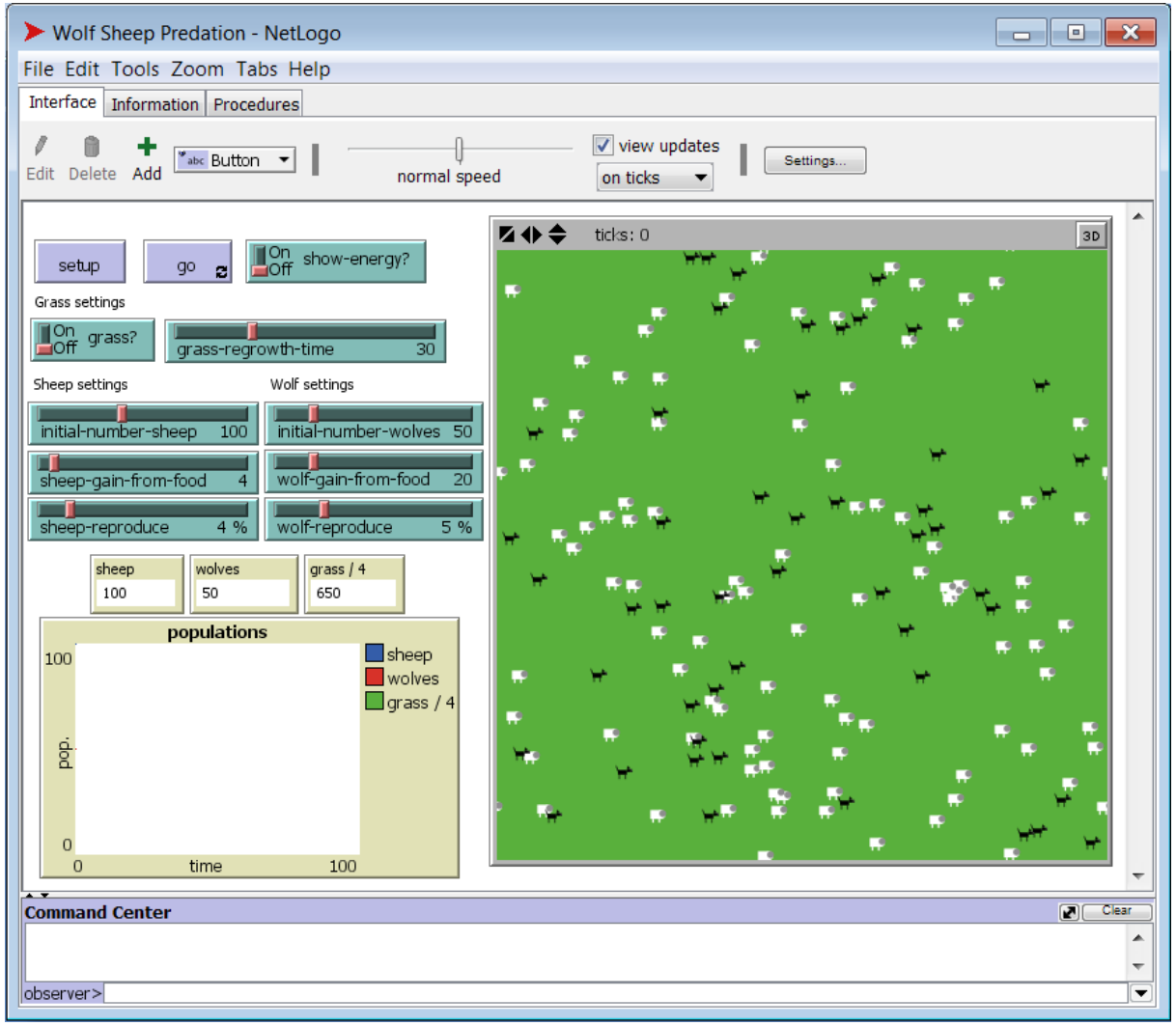
1. Sensors
2. Actuators/Effectors
3. Reasoning (maybe be very simple "if-then")
4. Learning (**optional**, defines an "intelligent agent")

Intro to NetLogo

NetLogo **Background**

- NetLogo is an IDE (integrated development environment) which can be used to create programs that simulate natural and social phenomena.
- NetLogo is particularly well suited for modeling complex systems that develop over time.
- Using NetLogo you can create programs containing hundreds or even thousands of "agents" all operating independently.
- For us NetLogo will serve as another programming environment in which to explore the Imperative, Procedural and Object-Oriented Paradigms.

NetLogo **IDE**



There are 3 tabs:

1. You will create buttons and graphs in the **Interface** tab.
2. You will write procedures (functions) to control agents using the **Procedures** tab.
3. The **Information** tab is for text about the project.

NetLogo **Syntax**

- The syntax for NetLogo is different then what you may be familiar with:
  - Comments are specified using ;;

    ```
    ;; this is a comment
    ```

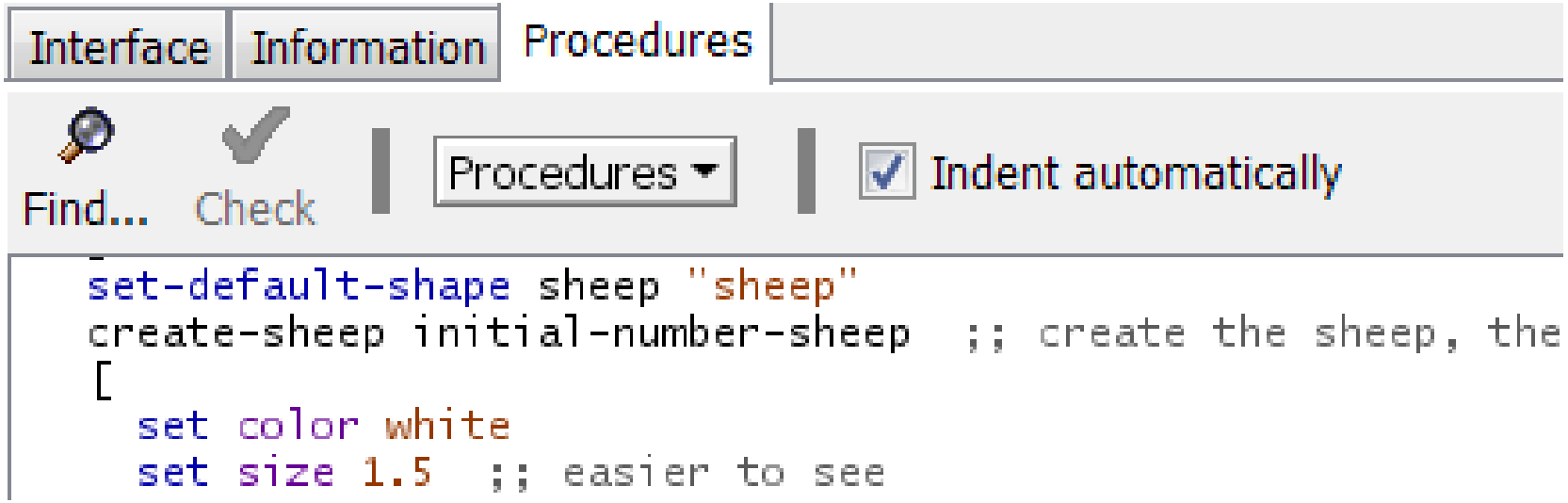
  - Each statement should be on it's own line and does not need a semi-colon at the end. *(Variables & commands can't have spaces).*

    ```
    clear-all
    ```

  - Commands are processed left-right top down. Commands are usually enclosed in functions (procedures) and these functions can be called using buttons on the interface window.
  - Blocks of related code (such as those that belong to an if statement) are designated using square brackets.

NetLogo **Primitives** (Semantics)

- "Primitives" are variables and functions (methods) that are part of the semantics of the Netlogo language.
- Primitives will change color when typed in your program:
  - Blue = commands
  - Purple = reserved variable names
  - Brown = Input parameters to a command



NetLogo **Variables**

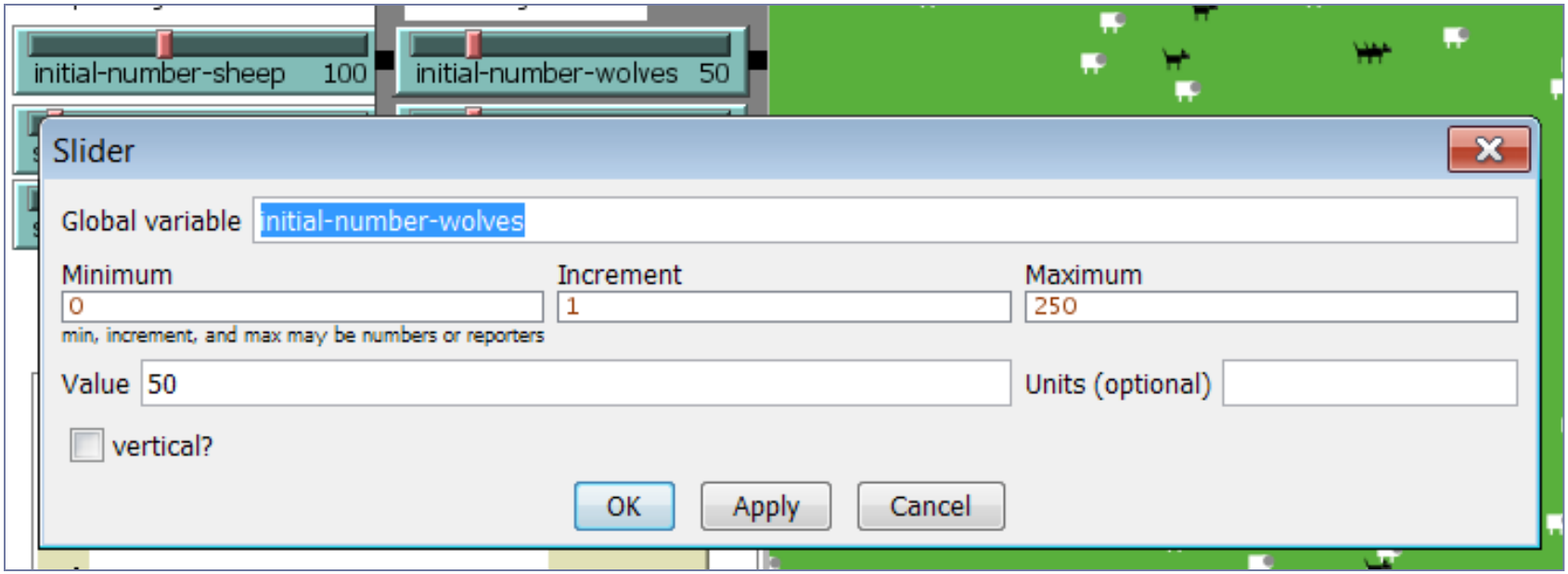
- Variables can be created in several ways, and can either be:
  - global -> anyone can see and use.
  - agent-based -> limited to use by a set of agents.
  - local -> only available within a procedure.

```
globals [ number-of-trees ]
;; There is only one number-of-trees variable
;; and everyone can see it.

turtles-own [ energy]
;; Each turtle has it's own private variable called
;; energy. 'energy' is a property of the turtle agents.
```

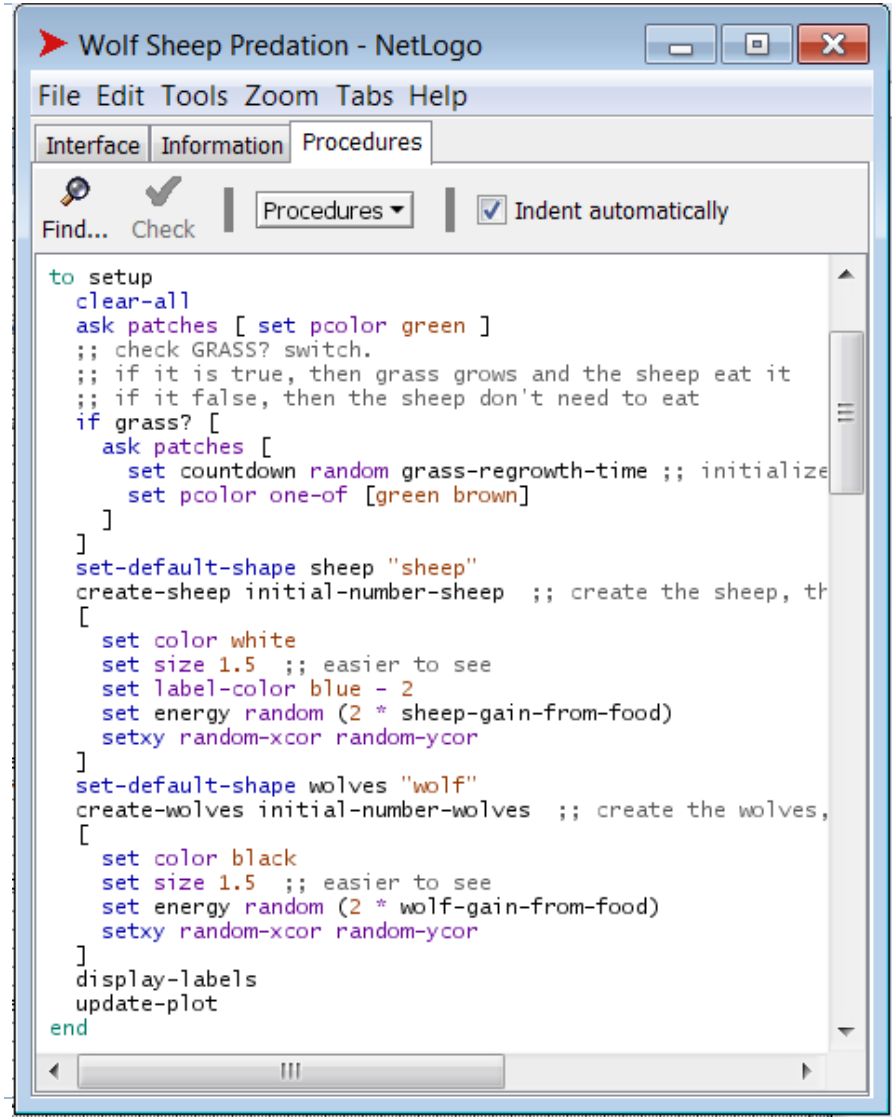
Variables (cont)

- You can also create variables by creating sliders in the interface window.
- These sliders will allow users of your program to modify the variables without having to change the code.



NetLogo **Procedures**

- A procedure (or function) is created using the reserved keyword 'to'.
- The name of the function follows 'to'.
- The end of a procedure is marked by the 'end' keyword.
- Procedures can be called by other procedures or by buttons on the interface.



NetLogo - **Agents**

- NetLogo creates programs by specifying behaviors (using procedures) for sets of agents and then allowing those agents to interact in a controlled "chain reaction".
- NetLogo has 4 types of agents:
  - Patches -> The squares on the world grid.
  - Turtles -> Agents that can move.
  - Links -> Lines between agents.
  - The Observer -> The master controller.

NetLogo **ask** and **show**

- Instructions can be given to agents and the values for variables can be set using the ask command.
- The show command in turn can be used to display the values of variables.
- In Netlogo groups of related commands are placed within square ('[' ']') brackets.

```
ask turtle 5 [ show color ]     ;; print turtle 5's color

ask turtle 5 [
    set color blue              ;; turtle 5 becomes blue
    rt random 360               ;; make right turn 0-360 degrees
]
```

NetLogo **Sensors** and **Effectors**

- Agents in Netlogo can use special primitives (built in functions) to "see" their environment.
  - turtles-here (returns a list of turtles nearby)
  - patches-here (returns a list of patches nearby)
  - turtles-at (returns a list of turtles at a specific location)
  - patches-at (returns a list of patches at a specific location)
  - self (returns local state information)
- Agents can also use special primitives (built in functions) to "affect" their environment, including other agents (**ask**).
  - setxy (change a turtles location)
  - set color (change a turtles color)
  - set pcolor (change a patches color)
  - hatch (create NEW turtles)
  - die (deletes a turtle)

DON'T PANIC

- You will have several labs on NetLogo.
- Lost of help is available online:
  - Fantastic (short) NetLogo Quick Guide
  - Full Netlogo Manual
  - NetLogo Dictionary

The End