13.3 *AgentSheets* Tutorial 2

*Introduction to Computational Science:*

*Modeling and Simulation for the Sciences, 2nd Edition*

Angela B. Shiflet and George W. Shiflet

Wofford College

© 2014 by Princeton University Press

Introduction

*AgentSheets* Tutorial 2 provides the background to understand a *AgentSheets* implementation of the model in Module 11.4, "Introducing the Cane Toad – Able Invader," and to use the software to complete various projects in Chapters 11 and 14*.* Tutorials 1a-c are prerequisites to this tutorial. Tutorial 2 does not develop a particular model but covers some of the commands employed in the implementation of Module 11.4's model. We recommend that you work through the introduction with a copy of *AgentSheets*, answering all Quick Review Questions in a separate document.

Coloring

**Type all requested commands in an *AgentSheets* project and an answer document.**

We will create a simulation of a grassy field with one rabbit. The rabbit always hops to the nearest of its four neighbors (north, east, south, west) with the most amount of food but uses energy hopping. Eating grass to gain energy, the rabbit dies if its energy drops to zero.

The shade of gray for a patch of grass indicates the amount of food, with black indicating the maximum amount and white none. The values for food and energy range from 0 to 1.

**Quick Review Question 1** Start a new *AgentSheets* project with default agent size. This question involves resizing the world.

**a.** Form a *Grass* agent that is all black. Create a *Rabbit* agent with an appropriate depiction. So that we have an indication of the amount of food in the cell, we will want to see the field background around the rabbit instead of white. To do so, from the **Color** menu of the *Depiction: rabbit* window, select **Mask Color**. How many items are in Mask Color submenu? Select **White**. The white areas in the *Rabbit*'s depiction will be transparent.

**b.** A patch of grass is to have a certain food amount, ranging from 0.0 to 1.0, which we should initialize upon creation. Thus, edit the behavior of grass. Form a new method. What trigger should we use for the initialization?

**c.** In the rule, with no condition, give the action for making all the grass have a *food* value of 1, where *food* is a local variable.

**d.** After that rule, have a new rule that with a 50% probability gives *food* a random floating-point number between 0.0 and 1.0. Give the rule.

**e.** Will the second rule ever be executed? To change the order of the rules, click on the red part of the *Then* in the second rule and drag to the top of the method. Have a comment for the method. Apply.

**f.** Start a new worksheet, called *Field*, filled with grass and one rabbit. Verify that black instead of white shows around the rabbit. How do we view the food value for a patch of grass. With that window still open, click on several *Grass* agents to verify that some have *food* being 1, while others have *food* being a non-negative value less than 1.

**g.** We would like the color of the grass to reflect the amount of food, with white representing a *food* value of 0 and black indicating 1. In the *While running* method for *Grass*, in the *Then* area, drag the ***Map*** action. We are mapping *food* to a color, so change the first parameter from "value" to "food." The *food* values range from 0 to 1, so what should we put in place of 100? In the action, click on red to see various lines of colors. Select white from the top left. Replace blue with black from the bottom right. Add a comment to the method. Apply.

**h.** What is the appearance of the worksheet? The mapping of *food* to color has not occurred because we have not run the simulation. Click the step button on the workshop to run one step of the simulation. You should now see a rabbit on a patchwork of squares with various shades of gray. Verify that black *Grass* agent has a *food*, while a lighter cell has a fairly small *food* value.

Hill Climbing

**Quick Review Question 2** During the simulation, we want the rabbit always to be heading to the nearest neighbor in its **von Neumann neighborhood** (four neighbors to north, east, south, west) with the largest food value, which we can accomplish with the ***Hill Climb*** action.

**a.** Edit the *Rabbit* agent's behavior. In *While running*, drag the *Hill Climb* action into the *Then* area of the rule. We are asking *Rabbit* to move on top of the *Grass* agent with the most food. What should we type in place of "value"?

**b.** Do we need to change the second parameter, which appears in a drop-down menu? Apply. Step through the simulation or run it slowly to verify that the rabbit is behaving as described.

Bottom and Other Directions

**Quick Review Question 3**

**a.** Observing the simulation, the rabbit does not diminish the amount of food in a *Grass* agent. Let us adjust the simulation so that the rabbit has an initial amount of energy, can eat all the food in a grass patch, and will gain the same amount of energy as the food it eats. Start the *Rabbit* agent with an initial amount of *energy*, say 0.5, and write how you did such. Apply.

**b.** Click on the rabbit in the worksheet. Does the rabbit have any attributes?

**c.** We placed the rabbit on the worksheet before defining its *When creating new agent*. Thus, erase the rabbit, place another rabbit on the worksheet, and save the worksheet. What are its attribute and value?

**d.** Let's have the rabbit eat all the food in its patch. Thus, the *Rabbit* agent must discover the *food* value of the *Grass* agent, which is on bottom. *AgentSheets*' documentation for [Mathematical Operators](http://www.agentsheets.com/Documentation/windows/Reference/tm_ti_formulasyntax.html) indicates what to do under "Remote Agent Attribute Access." The attribute is *food*, which is on the bottom. Thus, how should we refer to the food on the rabbit's grass patch? Note all the directions: **up** (north), **down** (south), **left** (west), **right** (east), **bottom** (below), **top** (above).

**e.** In *Rabbit*'s *While running* method, before performing the hill climb, increment *energy* of the rabbit by this food amount. Give the command. Apply. With the rabbit selected and viewing its attributes, step through the simulation and note how the rabbit's energy increases.

**f.** However, the grass' food does not diminish. To rectify the situation, we need to ask the corresponding *Grass* agent to make its *food* zero. After eating the grass, have the rabbit ask all *Grass* agents to execute their *eaten* method, which we will define shortly. Give the action for the rabbit to make the request. Apply.

**g.** In the behavior of *Grass*, add an *eaten* method. Upon the rabbit's instruction, all *Grass* agents, and there are many, will execute *eaten*. However, not all *Grass* agents should make their *food* have a value of zero. What *Stacked* condition should we use?

**h.** Give the corresponding action for the *Then* area. Apply.

**i.** Run the simulation on a medium speed. How does the appearance change?

Killing Thumper and Stopping the Simulation

**Quick Review Question 4**

**a.** The rabbit is certainly getting fat and happy hopping around the field. However, shouldn't the rabbit be expending energy with all that hopping, and what happens after all the food is gone? First, have the rabbit expend 0.4 amount of energy after hopping (hill climbing). Give the action. Apply. Step through the simulation and verify that the rabbit's energy level is changing correctly.

**b.** After decreasing the energy level, we need to check if that level is less than or equal to zero, in which case "Thumper" (the rabbit!) cannot live. How do we handle having such an *if-else* rule?

**c.** After the command from Part a, call a method, *checkState*. Give the action.

**d.** Form this new method. Give the condition for the rule. In the *If* clause, we **Erase** *Rabbit*. Moreover, there is no point in continuing the simulation without Thumper; nothing will change. Thus, include another action, ***Stop simulation***, which is under "simulation control" in the list of actions. Write a comment and apply. Following the rabbit's energy level, verify that the simulation behaves correctly.

Maximum and Minimum

**Quick Review Question 5** Our rabbit certainly is getting obese, with lots of energy, eating so much grass, but the problem statement indicates that the food and energy values should remain between 0 and 1. Let's adjust the simulation to do so.

**a.** Currently, after eating, the rabbit's new value for *energy* is *energy* + *food*[bottom]. However, we do not want this value to be greater than 1. Thus, instead, we set *energy* to be the minimum of *energy* + *food*[bottom] and 1. That is, if *energy* + *food*[bottom] is greater than 1, we use 1 instead. In *AgentSheets*, the formula for the minimum of *a* and *b* is ***min***(*a*, *b*). Make the change in the *Set* action, and give this assignment in your document.

Note that *min* has only two parameters. If we want to take the minimum of *a*, *b*, and *c*, we must take the minimum of two of the values and then take the minimum of the result and the other value, such as *min*(*a*, *min*(*b*, *c*)).

**b.** We only want to diminish the *food* value of the grass by the amount the rabbit eats, which is the difference between 1 and the rabbit's energy level before eating. Thus, before changing *energy*, define another local attribute, *oneMinusEnergy*, for *Rabbit*. Give this action.

**c.** We must also adjust the *eaten*. If the rabbit eats *energy* + *food*[bottom], then the new *food* value of the corresponding *Grass* should be 0. Otherwise, the rabbit only eats the amount *oneMinusEnergy*. How does *Grass* reference the value *oneMinusEnergy* for the *Rabbit* agent on top of the grass?

**d.** For *Grass*, the new value of *food* should be 0 for when the rabbit eats all the grass or the old value of *food* minus what the rabbit can eat, which is *oneMinusEnergy* for the rabbit on top of the grass (see Part c). In this case, we take the maximum of the two values. The *AgentSheets*' formula for the maximum of *a* and *b* is ***max***(*a*, *b*). Make the adjustment to *eaten* and give the action. Apply.

**e.** Step through the simulation. What color does a black cell turn after the rabbit has eaten some of the grass?

**f.** Run the simulation. After a while, what is the predominant value for the energy level of the rabbit?

**g.** Why?

**h.** Make sure all methods have comments. Save.