13.1 *AgentSheets* Tutorial 1b

*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

This document contains *AgentSheets* Tutorial 1b, the second of a three-part tutorial. Tutorials 1a-1c give an introduction to the system and prepare you to understand a *AgentSheets* implementation of the model in Module 11.2, "Agents of Interaction: Steering a Dangerous Course," and to use the software to complete various projects in Chapters 11 and 14*.* Tutorial 1a is a prerequisite to this tutorial, which develops a random walk simulation.

Random Walk Simulation

In this tutorial, we will create a simulation to generate a designated number of walkers on grass. The walkers move at random on the grass. If a walker finds a north-south road immediately to the east or west, the walker moves along the road to the south. Create the simulation as directed, and in a separate document, type the answers to all quick review questions.

**Quick Review Question 1** Start *AgentSheets* and open a new project, called *AgentSheetsTutorial1b*, with an agent size of 8 × 8.

**a.** Create a *Walker* agent with an appropriate depiction. What is the main tool you used for the drawing?

**b.** Create a *Grass* agent. Suppose we no longer want an agent. Select the agent. Then, under the ***Gallery*** menu at the top, select ***Delete***. Do not complete the deletion, but indicate the query that appears. Create a *Road* agent.

**c.** Edit the *Grass* depiction to be solid green and the *Road* depiction to be solid yellow. What is the best tool to use for such a fill?

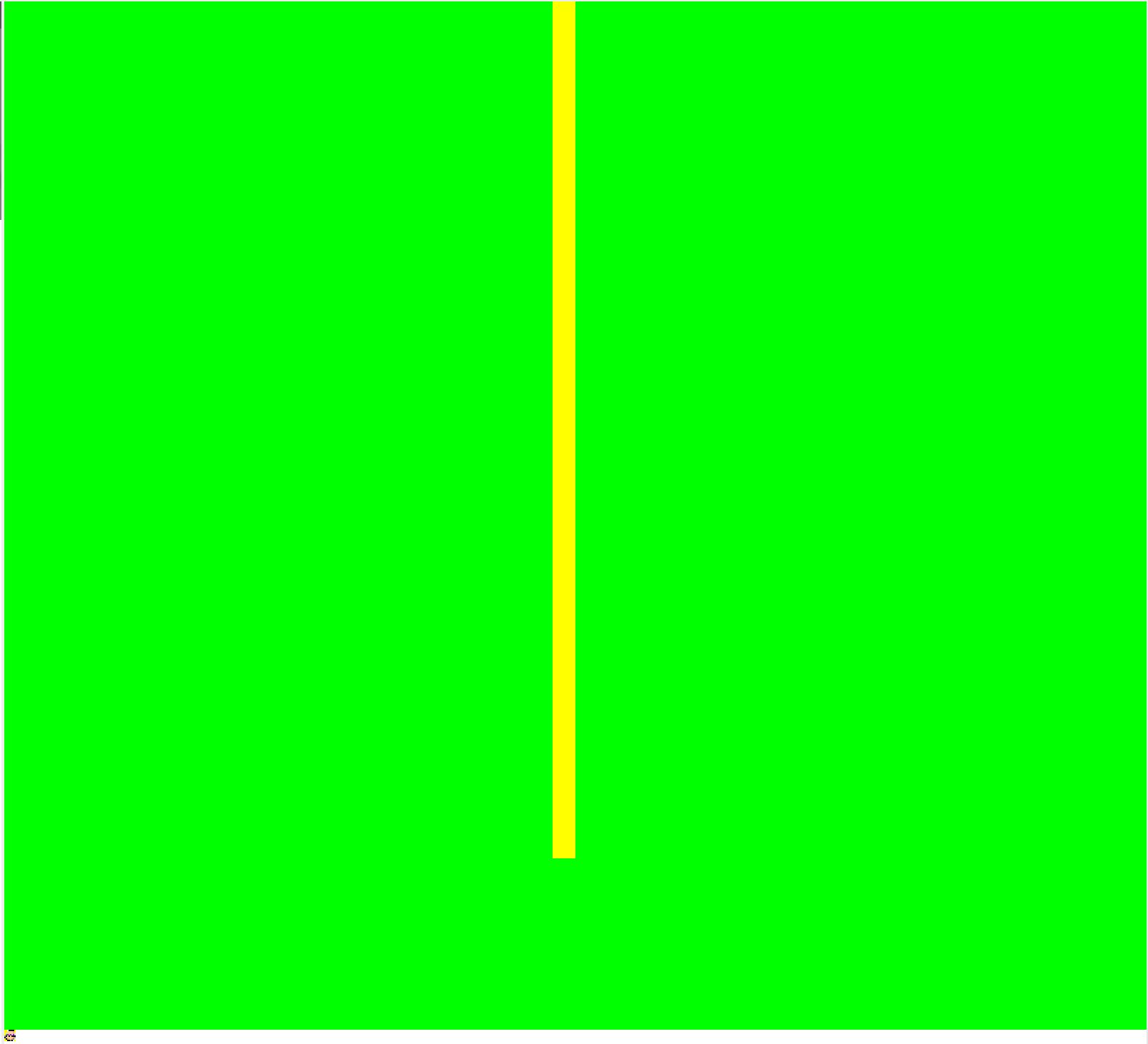
**d.** This simulation will occur in two phases. The first phase will generate the walkers at random positions on the grass. In the second phase, the walkers will travel at random on the grass and down the yellow road. Thus, we employ a *SimulationDriver* agent to manage the sequence of events. Create such an agent with the default depiction. How many agents do we have?

We now develop the worksheet.

**Quick Review Question 2**

**a.** How do we start a worksheet, called *Field*? Fill the worksheet with green *Grass* agents. Have one *SimulationDriver* in the bottom left corner.

**b.** We would like to have a road, two *Road* agents in width, going from the north border towards the south but ending before the border, such as follows:



First, let us erase the *Grass* agents that are there. Use of the eraser is a possibility, but rather difficult to handle. Instead, use the unfilled rectangle tool () above the question mark on the left of the worksheet. Carefully drag the tool using width two from the top of the worksheet to about two centimeters above the bottom border. If you make a mistake, you can make corrections with the filled rectangle or you can close the worksheet without saving and reopen. After using the unfilled rectangle tool, what is the appearance of the worksheet? With the filled rectangle tool, fill in the newly created blank area with *Road* agents. Save the worksheet. To avoid inadvertently changing the worksheet, click the selection arrow.

We now define the behavior of the *SimulationDriver* agent, the main driver of the simulation.

**Quick Review Question 3** Pseudocode for the action of *SimulationDriver* follows:

if the phase is 0, then

ask the *Grass* agents to initialize the walkers

change the phase to 1

else

ask the *Walker* agents to perform the appropriate moves

**a.** Thus, we need a local variable, say *phase*, to keep track of the phase. By default, the initial value of *phase* is 0. If *phase* were a global simulation property, which other agents could access, what symbol would need to appear before "phase" in the code? However, because only *SimulationDriver* uses *phase*, we can use *phase* as is in the code.

**b.** In the first rule, the condition should check if *phase* is 0. Under **attributes** of the Conditions menu, find the **Is** command () and drag the command to the condition area of the first rule. Give the replacement name for *value* in the command. Recall, that we are employing a local variable, not a global simulation property.

**c.** For future reference, notice that the equals sign in the command is in a drop-down menu. How many such condition operators are there?

**d.** For the action, we need the simulation driver to send a message to all the *Grass* agents to initialize the walkers. We have *SimulationDriver* broadcast the request using the **Broadcast** command (). Drag the action to the *Then* area of the rule. Give the value to indicate in the dropdown menu, which now indicates *Visual\_Agentalk\_Agent*. The text area beneath the dropdown menu indicates a method name, such as *method\_70* or *method\_* followed by another number. Change the name to *initialize*, a method we will define in *Grass*' behavior.

**e.** In the same rule, under the *Broadcast* action, drag an action for changing the phase to 1 and change "value" appropriately. State the complete action command.

**f.** Add a new rule to the *While running* method to accommodate the *else* in the pseudocode. Does the rule need a condition to test if *phase* is 1?

**g.** Drag the appropriate action to the *Then* clause and make appropriate changes to tell the *Walker* agents to execute their *move* method. Give the completed command. Write an appropriate comment for the method. Accept the changes.

We now define the *initialize* method for each *Grass* agent.

**Quick Review Question 4** In *initialize*, we want each agent to possibly generate a *Walker* agent on top of it. Pseudocode for *initialize* follows:

With probability *probWalker*,

create a new *Walker* agent on top of this *Grass* agent.

The following is another way of expressing the pseudocode:

If a random number is less than *probWalker*, then

create a new *Walker* agent.

**a.** Thus, we require a global *probWalker* simulation property. Define this variable and click the checkbox ***Control With Slider***. Recall that probabilities range from 0 to 1, so the ***Max*** value should be 1. Give *probWalker* a value representing 2%. What is *probWalker*'s value? Click *OK* and *Save*.

**b.** Because *probWalker* is a global simulation property, how must we reference the variable in the code? When defining the simulation property, we type "probWalker," but in a method, we must preface "probWalker" with a particular symbol.

**c.** Open the behavior window for *Grass*. Double-click *While* *running* to display other triggers, which indicate when to execute the method. Alternatively, click the **Triggers** icon () at the top of the *AgentSheets* window. How many triggers are there? Drag the ***On*** trigger, which indicates that we must explicitly call the parameter method, over *While running* in the behavior window.

**d.** For the condition part of the rule, we employ *% chance*. However, *% chance* expects a parameter between 0 and 100 (percent), while *probWalker* is a probability between 0.0 and 1.0. What should we multiply *@probWalker* by for the appropriate parameter? Save.

**e.** To complete the rule, we employ the basic ***New*** action (). The command tells us where and what agent to create. After dragging the action to the *Then* clause of the rule, by dragging change the arrow to a dot. Thus, we do not want to generate the new agent to the north but stacked on top of the current *Grass* agent. Change the agent parameter that *Grass* will create, too. After completion, what depiction appears in on the right of the action? Add an appropriate comment to the method, and apply the changes.

Before observing the simulation, we must define the *move* method for *Walker*.

**Quick Review Question 5** The *move* behavior for *Walker* has the following pseudocode:

If walker is on the grass next to the road

move to the road

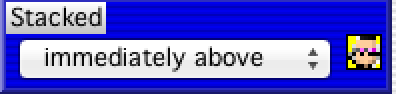
else if on the road

move to the south

else

move at random

**a.** Open the behavior area for *Walker*. Give the steps to change the trigger from *While* *running* to *On*.

**b.** To test if a walker is on grass, we employ **Stacked** (), which is a basic condition. Dropdown menus give us options for two parameters. How many choices do we have for the first parameter? Because walkers can stack on top of each other and a particular walker might not be immediately above the grass, the parameter *somewhere above* is probably best. Change the second parameter to be the depiction for grass. Alternatively, we can employ the more general **Stacked** condition (), which is particularly helpful if an agent has several depictions and the test is for stacking above any of the depictions.

**c.** Besides being above the grass, we need to test if the walker is next to the road. We are testing if the walker sees the road in which two possible directions?

**d.** To test if a walker observes an agent, we employ the ***See*** condition (). However, we must handle two directions: seeing the road to the right and subsequently moving to the right, and similarly for the left. Thus, how many rules do we need to employ for initially getting on the road?

**e.** Handling the road to the right, drag the *See* condition to the *If* area, which also contains the *Stacked* condition. Thus, with these two conditions in the *If* area, we are testing if the walker is on the grass **and** sees the road to the right. To what should we change *See*'s two parameters?

**f.** To move onto the road, we use the ***Move*** command (). After dragging the action to the *Then* area of the rule, do we need to make any changes in the arrow parameter for this rule? Apply.

**g.** Handling possibly moving to the left to get on the road involves the same process with only a change in direction. Thus, click on the rule, and then click the ***Duplicate*** button (). What do we change in the condition and action? Apply.

**h.** Once on the road, the walker should walk south. In this situation, in what direction should the walker see the road? In a new rule, drag the *See* condition to the *If* area and change its parameters appropriately.

**i.** Drag the appropriate action to the *Then* area so that the walker moves south, or down. Adjust the parameter as necessary. What are the action and parameter? Apply.

**j.** If not dealing with the road, the walker should move at random on the grass. Thus, generate a new rule as the last rule in the method. Because this is the default case, we have no condition. Drag the appropriate action to the *Then* area and adjust the parameter. What is the action and parameter? Write a comment and apply.

**k.** If the rule developed in Part j were listed first in the *move* method, would the walker ever get on the road? Why?

**Quick Review Question 6** Right-clicking can provide shortcuts to several options.

**a.** Open *SimulationDriver*'s behavior. Click on the *Is* condition in the first rule. How many options are available? Note the options available in the submenu *Add Condition*.

**b.** Click on the blank area for *If*. How many options are available? Note the options available in the submenu *Add Condition*.

**c.** Click on the blank area for *Then*. How many options are available? Note the options available in the submenu *Add Action*.

**d.** Right-click on the red background of a rule. How many options are available?

**e.** Open *Road*'s behavior. The *While running* method is not used. Using a shortcut, after clicking the method's name, what should we do to remove the method?

Save, run, and debug the simulation.