COGS 222 **Take-Home Lab #10** **Kirby & Danyluk**
Spring 2007

Due: beginning of class, Thursday, May 10th

You may work with a partner on this lab, if you like. No more than two people to a group, please. If you'd like to work with someone but don't have a particular person in mind, you can email the class. Just send your mail to cogs222@cs.williams.edu.

This lab will provide you with the opportunity to consider emergent behaviors. For this lab, you will program a bee simulation using StarLogo. As described by the authors, "StarLogo is a programmable modeling environment for exploring the workings of decentralized systems – systems that are organized without an organizer, coordinated without a coordinator." In other words, it is a programming environment for exploring complex group behaviors that arise as a result of very simple individual behaviors.

There are two parts to this lab. First, you will complete a tutorial on StarLogo. Second, you will write a simulation of your own.

Obtaining StarLogo

If you plan to work in TCL 216 or 217 or if you have an Apple computer, running MacOS X 10.2.6 or higher with Java 1.4 or higher installed, the easiest way to obtain StarLogo is by going to Prof. Danyluk's course web page. Click on the link for "StarLogo for COGS 222", which you'll find on the "Labs" page. The actual file to which the link points is a gzipped tar file. If you're working on a Mac, clicking on the link will download the file. You should then be able to just double click on the downloaded file to expand it into the StarLogo folder.

In the StarLogo folder, you'll find the StarLogo application, as well as Documentation and Sample Projects folders.

StarLogo is available (free of charge) for many different platforms. Just go to the following website:

http://education.mit.edu/starlogo/

Warning! The bee simulation you will do for Part 2 of this assignment was inspired by a sample program that comes with the StarLogo package. I've deleted it from the version that I've made available. Don't download StarLogo from the MIT site unless you can resist the temptation to look at their bee simulation program.

Part 1. Tutorial.

Naturally, you'll want to begin by learning how to use StarLogo. Fortunately, the authors of the program have put together a great tutorial. To start learning, double-click on "StarLogoDocumentation.htm" in the "Documentation" folder.

Next, click on "Getting Started". Definitely read the first three sections (Introduction, The Cast of Characters, The StarLogo Interface). Feel free to explore the rest, but you can now begin the tutorial.

To get to the tutorial, click on "Tutorial" at the top of the window. This will walk you through a termite nest-building simulation.

Part 2. Bee Simulation.

Now you're ready to program your own simulation. You will explore how bees build honeycomb through a series of very simple behaviors. Start by reading a high-level description of this project, which appears as one of the Sample Projects suggested by StarLogo. To get to the description, go to item 5 in the "Getting Started" description. This provides a link to "Sample Projects". "Bees" is the first project listed in the Biology section.

Your simulation should begin with *n* bees roughly in the center of the patch area. You needn't create a slider for selecting the number of bees as the project description suggests. You can simply hard code a number into the simulation (and edit the actual simulation code to try different numbers of bees). If you feel comfortable with the process of creating a button, however, you'll find it pretty easy to create a slider. The main difference is that you'll need to specify the name of a variable that will refer to the number selected by the user. Then in the program you write, you'll always refer to the variable name whenever you would have refered to the number of bees.

As you know from the tutorial, creating *n* bees (turtles?) will place them on top of each other in the very center of the patch space. To spread them out a bit, ask the bees to change to a random heading and jump a random amount (up to, say, a distance of 5).

Then the building behavior can proceed. Each bee should affect the patch on which it stands by leaving some small amount of a chemical. The bee should then move. The movement of the bee, however, is affected by the total amount of chemical on the patch. The more there is, the more significant the turn of the bee; the more there is, the farther the bee flies.

Since bees are leaving chemical on patches, each patch needs to know how much chemical is on it. To associate a variable (say, *chemical*) with each patch, you need to place the line

patches-own [chemical]

in either the Turtle Procedures or Observer Procedures. This associates *chemical* with each patch. Now each time a bee wants to refer to the chemical below it (i.e., on the patch where it's sitting), simply use the term *chemical* in the bee's procedure.

There are many ways a bee can place some chemical on a patch. I simply said that the amount of chemical is a number, starting at 0. Each time a bee places some chemical on a patch, the patch's chemical content is increased by 1. You can use the arithmetic operators +, -, * (i.e., times), and /. To set a variable to a new value, you need to use the "set" command, as in

set chemical …

After a bee places chemical on its patch, it needs to move. I made it turn to the right some amount dependent on the amount of chemical. I also added a bit of randomness (which makes the simulation more realistic). Be creative here.

Once the bee has turned, it should move. Again, the amount it goes forward from its new orientation should depend on the amount of chemical. (I just said it should move a distance determined by .10 * chemical, but there are more complex and interesting things you can do.)

Last, but not least, StarLogo provides a great way for you to see the intensity of a chemical on a patch. To do so, you can use the command

scale-pc yellow chemical 1 100

This says to make patches yellow, where the intensity of the yellow is determined by the value of the chemical. Color begins to appear as soon as the chemical level reaches 1. At 100, the patch is set to full yellow. (Above values of 100, the patch will turn white.)

There are various ways to use the "scale-pc" command. For example, a bee can issue this command before moving to a new patch. If you want to get fancy, you can use "diffuse" to let some of the chemical "bleed" into neighboring patches.

There are many other interesting commands you can try (including a "wait" command, so that you can slow down your simulation). You can check these out by clicking on the "Commands" button in the documentation pages.

What to turn in:

We would like you to turn in the working program, as well as paper copy of your program code. Just one per pair, please.

What follows are instructions for turning in the program from TCL 216 or 217:

First, make sure that the name of your file is something like

yourName.slogo

If there is a **Courses** icon on your desktop, skip the rest of this paragraph and simply continue with the rest. From the **Go** menu at the top of the screen, select **Connect to Server**. For the server address, type in **cortland** (or cortland.cs.williams.edu) and then connect. Connect as **Guest**. Select the **Courses** volume on cortland. The Courses icon should now appear on your Desktop.

Double click on the Courses icon and find the **cogs222** folder. Within that folder, you will see a **DropOff** folder. Copy your file into the DropOff folder by dragging it over to the DropOff icon. You will get a message saying that you do not have permission to see the results of the operation. Click ok! (The message is simply telling you that you can't look inside of the folder to see anyone else's work.)

And that's all there is to it!

Before leaving the lab, drag the Courses icon to the trash to disconnect from the server. You might also want to get rid of StarLogo, if you don't plan to use it again.

And, of course, don't forget to log out.