CS2112—Fall 2012

Assignment 5

Graphical User Interface Design

Due: Tuesday, October 30, 11:59PM

In this assignment you will use the Swing API to build a graphical visualization of the critter world described in the project specification. This visualization will have a user interface that permits the user to take control of one critter at a time and cause it to take actions.

0 Changes

- We want your UI to show the state of the critter. This now corresponds to the first *nine* memory locations, not just the first five.

1 Requirements

Your program should be able to display the current state of the world, which will be initialize in a random state. A random subset of hexes will contain critters and rocks.

The user interface will allow the user to single-step the world state, or start the world to run continuously. In either case the graphical display will be updated as the simulation progresses to show the new state of the world. A toggle control will determine whether the critters either take a `wait` action during the simulated turn, or take a random action during the turn.

There should be a control that limits the rate at which the simulation advances. Regardless of how quickly the simulation is progressing, the graphical display of the world will not be updated more often than 30 times per second. Thus, if the simulation is progressing very rapidly, the world state may not be displayed for some time steps.

The total number of time steps taken should be displayed on the user interface, along with the total number of critters and plants alive in the world.

Another part of the user interface will allow the user to control a single critter somewhere in the world. The user can click on the hex containing a critter to make it the currently controlled critter. The user interface will display the state of the controlled critter, corresponding to the 9 initial memory locations. It will also provide controls to cause the critter to take a particular action and advance the entire world by one time step.

The critters will not be controlled by programs in this phase of the project. Instead, they will take actions either under user control or randomly on each turn. However, the simulation should keep track of their attributes correctly. In particular, if critters run out of energy, they should die.

For more details about how the simulation of critters and other parts of the world work, consult the project specification.

1.1 Running your program

Your program must support the following command-line interface:

- `java -jar <your_jar>`
  Start the simulation with a world populated by randomly placed critters, plants, and rocks. The program should automatically read the input file `constants.txt` and set the value of the various simulation parameters accordingly.

2 Evaluation

We will be evaluating your user interface on multiple aspects. The visual appearance and layout will be factors, as will the design of the controls. We are looking for an attractive and functional user interface, but we have not specified its precise appearance and layout, nor exactly how the UI will allow the user to control the system. This is deliberate; we want you to think through the design. It's a good idea to storyboard and even to experiment with more than one UI design. See what works best—don't get locked into design decisions too early in the process.

An important consideration for the quality of your code will be how well you have separated the world model from its graphical display. The model should not depend on the user interface in any way, because such a dependency will prevent a distributed implementation of the simulation. We will also be looking for good documentation of your classes and their methods, using the documentation methodology described in class.

3 Programming tasks

You will want to figure out with your partner how to break up the work involved in this assignment. To get you started thinking about this, here are some of the major tasks involved:

- Designing the user interface with all the components needed to control critters. A good way to start this is to develop user interface sketches that show how the different user interface components will be placed on-screen.
- Implementing a new component or components to display the state of the critter world. This will involves graphically rendering hexes and critters. It should be possible to at least distinguish critters of different species, and to see the size and direction of a critter. How this is implemented is up to your discretion.
- Implementing a new component or components as listeners to make changes to the state of the world.
- Implementing the state of the critter world and all the critters in a way that is decoupled from the graphical display, according to the *model-view-controller design pattern*.

Keep in mind that you will be using this code for future assignments. Clean, modular code will make this much easier. Later on your graphical code will run on a different machine than the

simulation does, and the critter state will be located at the simulation machine too. Therefore, the critter state and the simulation code should not make any references to the graphics and UI classes. Violating this principle will create a lot of extra work for you in Assignment 7.

4 Restrictions

You may use any standard Java libraries from the Java SDK.

5 Overview Draft

We are requiring you to submit an early draft of your design overview document on Thursday before the assignment is due (October 25). You may not be able to predict what your design and testing strategy will look like in full at that point, but we want to see how far you have gotten. We will aim to get you quick feedback on this draft.

6 Extensions

For full credit, you are not required to do anything more than what is specified here. But for good karma you may add additional features. Possible extensions include but are not limited to the following:

- Make the window resizable.
- Allow users to give multiple critters a command before advancing the time step.
- Keyboard control of the program
- Panning and/or zooming the screen (you may find this worth implementing since it may be difficult to run your program with every hex in the window).

Make sure to document anything you do that goes beyond what is requested, and be especially sure that any extensions you make do not break the required functionality of your program.

7 Submission

You should compress exactly these files into a zip file that you will then submit on CMS:

- *Source code*: You should include all source code required to compile and run the project.
- *Other files*: It is possible to use other files as part of your UI. For example, you might read in image files or other data files that control appearance. Don't forget to include these.
- *Tests*: You should include code for all your test cases.
- `overview.txt/html/pdf`: This file should contain your overview document.

Do not include any files ending in `.class`. We expect you to stick to Java 6 features and avoid features found only in Java 7. You can set project properties in Eclipse so that it warns you when Java 7 features are being used.