CS 2112 Fall 2019
Assignment 6
Graphical User Interface Design

Due: Tuesday, November 19, 11:59PM
Design document due: Monday, November 11, 11:59 PM

In this assignment you will use the JavaFX API to build a graphical visualization of the critter world described in the Project Specification. The visualization will have a graphical user interface (GUI) that will display the positions of critters, rocks, and food, permit the user to load and inspect critters, start and stop the simulation, adjust the rate, or advance the world one step at a time.

The majority of the work for this assignment will be on new functionality. However, you will also be expected to fix bugs in your code for Assignment 5 as necessary.

1 Changes

- November 7 - Scrolling AND zooming are mandated.

2 Instructions

2.1 Grading

As usual, solutions will be graded on design, correctness, and style. A good design makes the implementation easy to understand and maximizes code sharing while maintaining modularity. Your program should compile without errors or warnings and behave according the requirements given here. Your code should be clear, concise, and easy to read.

We will evaluate your user interface on visual appearance, layout, and design of the controls. We are looking for an attractive and functional interface that offers an enjoyable experience for the user. We have not specified precisely what this means, as we would like you to think it through and come up with your own design.

A good idea is to storyboard your design, and also to experiment with different layouts to see what works best. Do not get locked into design decisions too early in the process.

2.2 Final project

This assignment is the third part of the four-part final project for the course. Consult the Project Specification to find out more about the overall structure of the project.

2.3 Partners

You will work in groups of two or three for this assignment. This should be the same group as in Assignment 5. Remember that the course staff are available to help with problems you run into. For help, read all Piazza posts and ask questions that have not been addressed, attend office hours, or set up meetings with any course staff member.

2.4 Restrictions

For the first time, you will be starting from scratch. We are not releasing any starter code for this assignment, other than the Gradle file used to allow your code to be properly compatible with JavaFX.

You may use any classes from the standard Java system library. If there is a third-party library you would like to use, please post to Piazza to receive a confirmation before using it. You may code the GUI in JavaFX's XML, use a GUI builder such as the JavaFX Scene Builder (older version from Oracle, or a newer open-source version from Gluon). You may also hand-code your GUI. There are no restrictions on design tools.

3 Design overview document

We require that you submit an early draft of your design overview document before the assignment due date. The Overview Document Specification outlines our expectations. Your design and testing strategy might not be complete at that point, but we would like to see your progress. This is also a good time to submit design sketches for the GUI. You can go over your design document with the course staff and receive feedback in lab.

4 Version control

As in the last assignment, you must submit a file `log.txt` containing the commit history of your group. Additionally, you must submit a file showing differences for changes you have made to files you submitted in Assignment 5. You can get the differences by using `git diff` against the appropriate commit hash for your A5 submission.

5 Requirements

Your program should be able to display all aspects of the current state of the world. It should graphically render hexes and their contents, including food, rocks, and critters. It should be possible to distinguish critters of different species and to see the size and direction of each critter.

The GUI should allow the selection of world files, preferably using a FileChooser. After the user has selected a world file, the program should load the file and initialize the

world as done in Assignment 5. The critters will be controlled by critter programs using the interpreter and simulation engine you built in Assignment 5.

The GUI should allow the user to step the simulation one step at a time or let the world run continuously. It should be possible to pause and resume a continuously running simulation. The graphical display will be updated continuously as the simulation progresses to reflect the current state of the world.

The total number of time steps taken during the simulation and the total number of critters alive in the world should be displayed. As in Assignment 5, the user should be able to create a new random world, load a world, or load a specified number of critters.

When loading a critter program file, the user should be able to either specify a number of critters to be randomly placed throughout the world or select a particular hex to place a critter.

The user should be able to set the maximum rate at which the simulation advances (including 0). Regardless of how quickly the simulation progresses, the graphical display should not be updated more often than 30 times per second. If the simulation is progressing faster than this, some intermediate world states will not be displayed.

Another part of the user interface will allow the user to inspect a single critter somewhere in the world. The user can click on the hex containing a critter to make it the currently displayed critter. The user interface will indicate which critter is currently displayed and will also display the state of the selected critter, corresponding to the 7 initial memory locations, along with the critter program and information about the most recently executed rule. As the simulation progresses, this information will be updated accordingly.

The particulars of the design are up to you. You should strive for an interface that is intuitive, user-friendly, and visually appealing.

5.1 Preview of Assignment 7

The GUI that you are creating for this assignment will have to serve you for Assignment 7 as well. In Assignment 7, the GUI will be running on a local machine and the game engine and model will be running on a remote server. You will write both the client and the server.

Communication between the two machines will be done using the `http` protocol. The set of operations that your client and server will use to communicate is known as an *Application Programming Interface* (API) and is documented in the API specification. You might take a look at this to get an idea of the features that you will need to implement in Assignment 7 to support this communication. (The currently linked API is from 2018 and will be updated soon with some modifications.) As emphasized in lecture and recitation, attention to the MVC (model-view-controller) paradigm and the strict separation of concerns between the GUI and the world will make for a smoother transition to the client-server environment.

6 Running your program

It should be possible to run your program with the following command:

```
java -jar <your jar>
```

Your program should start up with a default world populated by randomly placed rocks, initialize the GUI, and wait for user input. The simulation should not be running initially.

Note that there are no command line options. All further user interaction should be done through the graphical user interface.

7 GUI Design Choices

Good GUI design can be difficult. It is largely subjective, and there are no hard and fast rules for what makes a good design. That said, there are a few simple strategies you can follow that will enhance the experience for your users.

7.1 Buttons and Control

Users should not need to play a guessing game to find out what buttons do. They should be clearly and succinctly labeled to describe their function. In some cases, an icon can be better than words.

Placement of buttons depends on function and frequency of use. A small button way off on the side of the screen is difficult to access compared to a large button near the focus of the window, and can be annoying if the user needs to use it repeatedly. On the other hand, the close/resize buttons at the top right of windows in Windows and the top left on a Mac are typically used only infrequently. Placing them far away from the central area of the screen makes it unlikely to click them accidentally.

A GUI can provide *keyboard shortcuts*, so that the user doesn't need to move the cursor to initiate an action, or *context menus*, where a user can right-click to display a menu wherever the cursor happens to be, enabling actions that make sense at that particular location. *Tool tips* can be displayed when hovering with the mouse over a component to describe its function.

Consider using some of these features to provide an intuitive and manageable interface.

7.2 Color Schemes

As a general rule, use highly saturated colors sparingly. Saturated colors make sense in the (few) places where you want to draw the user's attention. Avoid having too many colors at once; monochromatic, adjacent, triad, or tetrad schemes work well. A useful site for picking color schemes is paletton.com.

7.3 Navigation

Scrolling and zooming must be implemented to make it easy to view and navigate large worlds.

8 Overview of tasks

Determine with your partner how to break up the work involved in this assignment. Here is a list of the major tasks involved:

- Implement a GUI to display the state of the critter world and respond to user input.
- Connect the simulation engine from Assignment 5 to the display. This means allowing the display to update as the simulation progresses and to start, stop, and step the simulation.
- Implement the loading of critter and world files and the placement of critters into an existing world.

9 Tips and Tricks

Take care not to entangle your world model too tightly with this new interface. Proper separation of the simulation and GUI is very important and something we will be looking for. The model should not depend on the user interface in any way, because such a dependency will interfere with the distributed implementation of the simulation in Assignment 7, in which the world and the GUI will be running on different machines. We will also be looking for good documentation of your classes and their methods.

GUI code can become quite long, and you will likely have to make a conscious effort to keep it clean and readable, more so than with previous assignments. In addition to organizing your classes in packages, think about how to organize your resources (fxml files, images, icons). Never access resources using absolute pathnames; they should load properly even if the project's location changes. Instead, use one of the methods in ClassLoader or Class that look for resources in your program's classpath.

10 Submission

You should submit these items on CMS:

- `overview.txt/pdf`: Your final design overview for the assignment. It should also include descriptions of any extensions you implemented. Additionally, you should document the different aspects of your GUI. Do not assume that all observable features of your GUI are noticeable to an unfamiliar user. You should also indicate the operating system and the version of Java you use.
- Zip and submit your `src` directory. This directory contains:

  - **Source code**: You should include all source code required to compile and run the project. All source code should reside in `src/main/java` with an appropriate package structure.
  - **Resources**: All resources for your GUI should be under `src/main/resources`. These will be included by simply zipping up `src`.
  - **Tests**: You should include code for all your test cases in `src/test/java` and resources for your tests in `src/test/resources`. You are welcome to create subpackages to keep your tests organized.

  Do not include any files ending in `.class`. Git users can save space by excluding the hidden `.git` folder when zipping.
- `build.gradle`: As mentioned, you are allowed to include external dependencies for this project. (If you do, please clear it with the course staff in advance). You should submit your `build.gradle` file containing the correct main class name and any dependencies you require. Make sure your `build.gradle` includes everything the original released file contained along with any additional dependencies.
- `screenshots.pdf`: A `.pdf` file containing 3–4 pages of screenshots showcasing your GUI.
- `log.txt`: A dump of your commit log from your version control system.
- `a6.diff`: A text `.diff` file showing the changes to files carried over from Assignments 4 and 5 and used in this assignment. This can be obtained from the version-control system.