CS 2112 | Assignment 6 Overview

Graphic User Interface Design

Charles Qian {cq38}

Kelly Yu {kly24}

1 Our Main Class

Our Main class will be named Main.java. The client will run the game from jar file. After running this jar file, the client will be presented with a graphical user interface which allows the client to control the world using onscreen elements.

2 Summary

We anticipated that the most difficult parts would be connecting our preexisting code from Assignment 5 to the graphical interface. In reality, the main challenge was understanding MVC and correctly working Observer/Observables, as well as the learning curve that accompanied working with JavaFX. We ended up with a class representing the graphical interface implementation (View.java) and a Controller (Controller.java) that accessed static UI elements within an FXML file (activeUI_1.1).

3 Specification

We implemented a graphical interface that has the ability to make a new simulation, load a world file, load a number of critters based on a critter file, advance a number of time steps, control a specific critter, and return visual feedback on the state of the world. The graphical interface displays the current state of the world. In addition, the user should be able to control how fast the world is stepping, or let the world play continuously for a time. The user can control critters by selecting the critter's current hex. The user can also select a Hex and use the Place button to place the currently loaded Critter upon that Hex. There will be a display for the state of selected critters and hexes.

4 Design and Implementation

4.1 Classes and Architecture

Controller.java accesses the UI as specified in activeUI_1.1.fxml and handles events that are triggered by static UI elements. Controller has access to Main and View. View.java handles the drawing of the world, the hexes, and the critters (non-static elements). Attached in View we also decided to put Event handlers to handle mousing over hexes, for things such as highlighting the current Hex. Main.java initiates the stage. Our Model in this MVC structure is Creator.java, which extends Observable. Our Observer is View.java. We used this to aid decoupling.

We also used a Timer Design that updated the view of the world every 33 milliseconds, aka 30fps. However, because the Timer is on a different Thread, this yielded a problem because JavaFX does not like it when other threads try to modify JavaFX nodes. Thus, we had to implement Queues, so that when the Observable notified the Observer of a change, the overridden update method will add things to the queue that must be changed, such as the position or direction of a critter or the food content of a hex. This allowed the main JavaFX thread to later go through all objects in the different Queues and make the required updates.

4.2 Code design

For a few days we played around with JavaFX tutorials and sample code, as well as FXML code, before deciding to incorporate FXML code into the assignment because it felt like a practical way to distinguish UI elements neatly and in one place. We first were unsure of choosing which class would reference the FXML file, but decided on having Controller.java instead of Main.java, because we found that in this way we could instantiate the Controller class and have it operate more independently.

We also used HashMaps for storing images of hexagons and critters, because this allowed us to access critters by inputting that critter as a key, rather than relying on using an index (We had initially used ArrayLists).

4.3 User Interface Design

We made sure not to use bright colors. We tried to make it as intuitive as possible, by putting UI elements with similar functionality on the same tab, and setting the default tab to commands we felt would be used more often.

Charlie had the brilliant idea of having our game inspired by the popular board game Settlers of Catan, so we found some good fanart from DeviantArt and used those critters. The default art had too much contrast, so Kelly took the images into Photoshop to add the alpha layer, and to also fix color (by decreasing vibrance, for example). The artists are credited in the Acknowledgements tab.

We also felt that we needed a better Hex to represent food, so Hexes with food are colored green for grass. This also made our world a lot more colorful. We considered using CSS for styling at the beginning, but as the project progressed it didn't seem necessary, as JavaFX is very well equipped.

Charlie also improved user experience by providing a smooth animation that allowed the user to expand and hide the right panel. This allows more experienced users to opt for this minimalistic interface. In the future, we would also like to implement hotkeys for critter control and other tasks (Shift + L: Load World, N: New World, L: Load Critter...). Also, users can manually put in the fps inside the text field located below the fps slider. Charlie figured out how to slowly darken the screen when the dialog box prompting the user to choose whether to override an existing world appears.

4.4 Programming

During this project, Charlie took the lead in implementing Controller.java, as well as designing the UI for static elements. Kelly took the lead in implementing View.java, which dealt more with non-static elements. Both experimented heavily with JavaFX and FXML, and both did incremental testing throughout the project. Kelly did the handling of user control over a specific Critter.

5 Testing

5.1 Testing Strategy

We ran the Main.java after every major change to see how the GUI was updated or how the critters were responding. We would do a variety of tasks that a typical user might do to simulate a real-world situation. We also clicked random things to simulate a new user experimenting with an unfamiliar user interface.

5.2 Test Results

Once in a while, we have Concurrent Modification Errors, likely due to the fact that many fields are being updated constantly by different objects.

6 Known Problems

In our GUI, sometimes the Hex or Critter Info is cut off. Big worlds will have low performance. **We also discovered on the day of submission that our project is also completely dependent on the user having Java 8.** Another note is that when a Critter is selected to be controlled, the highlighted Hex does not follow the critter, and the critter the user controls will not change until the user either clicks a empty Hex or until the user clicks on a different Critter.

7 Comments

We started this project one week before the deadline. There were a lot of issues at first with working with GUI programming, but after half a week, both of us felt a lot more confident and capable of JavaFX and FXML and are grateful for the experience. There were times that a lot of frustration would impede the feeling of having fun, but all in all, we had a lot of fun, especially towards the end, when our ladybugs started moving and being alive.

7.1 Time

Charlie spent around 25ish hours; Kelly spent around 30ish.

7.2 Thoughts

Kelly is slightly disappointed that none of her original artwork got to be in the final version, as there was no time to import them into Photoshop and make pngs out of them, and then import into Flash and make animations for when critters were idling around. She is also happy she was able to work with FXML, because aside from animation, Kelly really likes Web Development. Charlie feels relatively good and he believes this project has pushed him to be a better problem-solver.

In office hours, all the TAs were really helpful in describing MVC and Observer/Observable design. However, most were accustomed solely to Swing, and had GUI advice that was only relevant to Swing, and in some cases could not be applied to JavaFX.