Metadata

A6: Graphical User Interface Design

Ben Grass (bdg83), Andrey Yao (awy32), and Cameron Russell (cfr59)

Summary (50-300 words)

The most difficult part of this assignment was, as always, the design. Especially in designing a GUI where so much of the work is literal design, careful planning and detailed execution was of the utmost importance. Additionally the finding the proper x,y coordinates that represented the locations we wanted to place the critters, food and other elements of our GUI proved to be quite difficult. We overcame this difficulty by tackling the design iteratively, we would adjust an x or a y coordinate, run the GUI and see where everything rendered in the window. Then, we'd go back and refine our measurements until eventually everything was in the right place.

The most noteworthy aspects of our design is the multithreading we used to speed up the simulation computation and reduce the lag we experienced when running many steps per second. Whereas before, without multithreading, we couldn't run anything above 30 steps per second without the GUI freezing up, after we implemented this change the number of steps per second we could handle shot up drastically.

We went with a frog/lake theme with our actual graphical design, the critters are represented as frog-like creatures, the food as delicious red berries, any traversable tile as a lilypad and finally any rock as a water tile (no lilypad).

Specification (10-500 words)

The assignment specification certainly allowed for much interpretation/imagination on our end, as there were really only a few requirements that we were forced to implement, and the other design choices were completely up to us. In order to keep the GUI simple and improve the user experience we opted to separate our GUI into three main "panels". The panel on the left functions as a general control panel, allowing the user to load worlds and critters into the simulation. The middle panel, the largest, is the actual rendered view of the simulation. It allows for scrolling with a bar and with the mouse (click and drag) as well as zooming using the intuitive buttons in the lower right hand corner of the center panel. Finally, the panel on the right provides more information about the tile selected by the user, as well as its contents.

Finally, the very last piece of our GUI (despite the AMAZING title screen) is the bar on the very bottom which allows the user to advance the simulation automatically (at a specified rate of steps per second) or manually (at a specified number of steps per click).

The only other large specification we filled in was how to visually display the species of a critter. We opted to hash the species name of the critter into a random color which is then displayed as a dot (with a black outline) on the critters back. Unfortunately, this became tremendously difficult as we implemented the critters growing as their size increased. The position of the dot would change slightly as the critter grew which resulted in a slightly less than optimal appearance.

Design and Implementation (150-1000 words)

We used a combination of JavaFX hardcoding and scene builder to develop the GUI for this simulation. Additionally we used a combination of open-source art assets found on the internet and our own customized art designed on the Krita art software which -- for anyone unfamiliar with the technology -- is a horrible, horrible thing that should never have been allowed to exist. In order to implement the multi-panel design we conceived we split the GUI into 4 subclasses, each of which implements the singleton design pattern. This allowed us to separate concerns and divide labour better, as we weren't worrying about what each of the other aspects of the GUI looked like or what the other group mates were working on as we coded our own subclasses. The addition of the singleton pattern implementation to this design also helped tremendously as it enabled each of the setOnAction methods inside the subclasses to access the individual instance variables in that class which proved to be an invaluable asset.

Testing

In addition to extensively testing every button and function of our GUI we also allowed some of our friends, whether they be fellow CS students, other engineers or lowly non-engineers.

In testing our GUI ourselves we found a large number of technical bugs, things that didn't do what they were supposed to. In giving our project to friends we found a number of design flaws that we then tuned to make our GUI more intuitive and inviting.

Graphical User Interfaces are inherently more challenging to test than ordinary backend applications; a more intensive, brute force, and manual approach is necessary. Thus, to adequately test our GUI implementation, we developed a list of actions to test all edge case scenarios we could think of, and ran through each of those actions. For example, we made sure invalid world files were not loaded and helpful error messages were provided, the game didn't crash when the steps per second rate was high, resizing the window while the simulation was running didn't cause weird glitches, etc,. Additionally, we tested normal functionality (not edge cases), by running through many simulations of the game to ensure nothing crashed and corresponding values were being updated as the game progressed.

Work Plan (20-100 words)

The division of labor:

Ben was responsible for the games specialized graphical components (critters, food, lilypads, etc,). Andrey was responsible for creating the starting page, the primary game page, the dynamic hex-grid, tile highlighting, zooming/scrolling, and advancing the game time; he also had prior javafx experience. Cameron was responsible for allowing the user to import new worlds/critters into the game, handle any errors associated with that, and the info page.

Known Problems (1-500 words)

The dot on the critters back that displays the species of the critter as a color was VERY difficult to attach to the critter in the right place. This resulted in the dot being positioned kind of awkwardly on the critter.

Highlighting the tile doesn't work once the program has started and you can see the previous context of the tile sometimes in the hole of the lilypad.

Sometimes the world and critter menus don't close when you press finish, you need to press the exit button.