EE 422C Project 5 (Critters part 2) README

Akaash Chikarmane (avc536)
Nicholas Sutanto (nds729)
Spring 2017
Git URL: https://github.com/AkaashChikarmane/ee422c_project5.git

For the view component, we decided to have the Critter World be displayed in a new window. We did this by making a new stage object called *worldstage*, which will contain the Critter World grid that we built for this project. The Critter World was implemented using a *GridPane* layout, with a width of*Params.world_width* and a height of *Params.world_height*. The size of the whole grid is fixed to a height and width of 600px, and the size of each individual 'box' in the grid scales according to the values contained in Params.java in the Critter project. For example, if the world height and width are set to 6, the width and the height of each 'box' in the grid will be 100px, and if the world height and width are (100 x 50px), then height and width of each 'box' will be (6x12px). Our fixed grid size of 600px makes sure that our view component can still fully display even on computers with smaller screen pixel resolutions. To draw our critters, we used the JavaFX Circle, Rectangle, and Polygon shapes and made sure they scaled the same way the grid boxes did. We used the *grid.add()* method to add the critter shape objects into the grid display, and used empty rectangles to make sure that our grid rows and columns had uniform width and height all throughout.

In our controller component, we implemented start and stop buttons for animation as well as a slider to adjust the refresh rate. Our animation works by disabling all buttons except stop, start, and quit then calling a Timeline event that has a total duration of 1/(refresh rate). The animation executes one timestep, updates the stats window then leaves. So it will execute the animation (refresh rate) times per second. If the refresh rate is above 20, we found that the stats updates and world updates would lag so we decided to make the rate at which the stats window and world window are updated inversely proportional to the refresh rate to reduce lag.