# README

<Gaurav Nagar> <gn3544>
<Minkoo Park> <mp32454>
<https://github.com/gn3544/CrittersPart2>

Fall 2016

## Codes and Graphics

Our View and Control components are almost completely separated in our project. Control component is exclusively in Main.java. Control includes all user interfaces (buttons, text fields, and slide bar) and their event handlers using Lambda expression. We used standard JavaFx buttons, textfields, etc. We used GridPane to arrange all interfaces. View component is implemented in displayWorld() method in Critter.java. View component basically draws critterWorld, calculates critter positions, and places them on the map. View component uses GridPane to draw a board and empty Rectangle shapes to fill each cells.

We created additional class called ErrorMessageBox to display error messages to users. It has a method that takes error title and message as input then creates a window box that displays these parameters.

On the other hand, we also utilized the AnimationTimer() class to display the animation of the critter world as time progressed, dependent on the frame inputted by the user. This timer would count down depending on the frame, and reset the timer back to the number denoted by “frame”. Additionally, to make sure that user could only interact with the “Animation” button after toggling it once already, the team used a “toggle” semaphore for its simplicity. As a result, the program would stop animating after the “Animation” button was toggled once again.

## Good Features

1. we have created an error message box to notify users of all types of errors that can happen while using our simulator. A pop-up window with error type, error name, and an instructions is generated when processing errors or exceptions occur. Instead of printing out error message to a console, we wanted to make user-friendly interface that directs users more effectively. Furthermore, the message window actually disables all the other components of the simulation, so it makes sure that the user checks the message and closes it before continuing with the simulation.
2. We implemented simple and effective controller. For each command, we made a button that the user can press. For inputs, which can be the number of commands to run or type of critters to use, we have created textfields. All these interfaces are grouped and laid out according to their functionality. Additionally, for animation, we used a slide bar that allows wider spectrum of frame speeds that is easy to play with.

3. Our grid is optimized for a computer screen. Instead of resizing the grid window depending on the number of total cells to include, we have set grid window to a max size, and just resize the gridpane that holds all the cells. This guarantees that our grid window is always as big as it can be, and gridpane that holds the cells fills as much space of that window as possible.

**Features That Did Not Meet the Standards**

We made sure that all features meet the minimum standards. However, there is one specification that we did little differently than the pdf. The pdf seems to suggest that when the user presses “make” or “step” button, we should automatically show the updated world. However, if we do so, we thought the “show” button would lose its purpose and become unnecessary. So when the user presses make or step button, we do not automatically show the world on view component and the user has to press the show button to see the updated world. This approach basically follows standards in Project 4 where the user had to type “show” every time they want to see the updated world.

This is a very trivial change and can be fixed quickly by rearranging few lines of codes. We still thought it would be necessary for the TAs to know that this design was intentional.

**Unsolved Problems**

No unsolved problems found so far.