CmpE 160 - Introduction to Object Oriented Programming

Project #2 - Snakes!

Due Date: 29.04.2018 Sunday 23:55

For this project, you are going to implement a game called Snakes! It is a game like the classic game Snake with some important differences and enhancements. Before going into the details, let us show you what the game will look like:

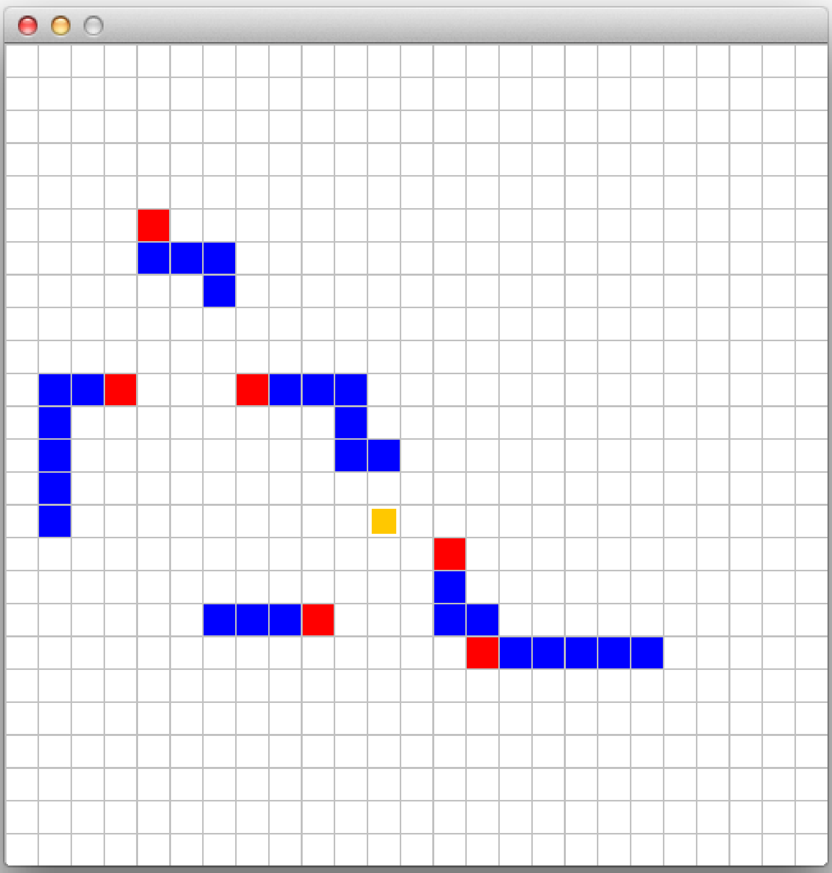

Figure 1: Snakes Game

1. Overview

- Even though Snakes is a game, you won't be playing it. It will be playing itself, just like project 1. The snakes will move and hunt for food themselves without any input from the user. That means you need to code some simple AI.
- A snake consists of consecutively placed blue squares and a special red square denoting the head.
- Initially there will be only one snake of size 4.

- Snakes are able to move in four directions (left, right, up, down) (check Section 4 for details). They cannot go past the world boundaries.
- At any time, there will be a single randomly placed food (yellow square in Fig. 1) in the game world, and the snakes will try to find and consume it.
- A snake that consumes the food will grow by 1 (check the next section for details).
- When a snake reaches size 8, it will divide, producing two separate snakes of size 4 (check the next section for details).
- The game will continue following these rules, the snakes will multiply and hunt for more food!
- This time, we do not give you any classes. You need to implement the project (including the UI) completely by yourself. But, you may (and are encouraged to) copy and use UI-related classes from project 1. Since this game is also grid-based, it should be fairly easy to adapt the code from project 1 to build this game.

2. Game Rules

- The first snake will be positioned as shown in Fig. 2, the food should be positioned randomly.

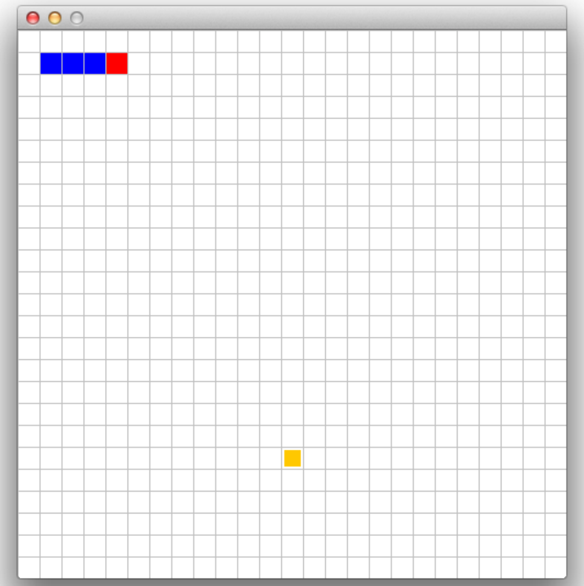

Figure 2: Initial Snake Position

- To represent a snake's parts (including its head), you are advised to use a linked list. You may either use Java's LinkedList (java.util.LinkedList) or another implementation. Linked lists are very convenient for implementing snakes and their movements, so this should actually make your job easier.
- The snakes will move like the snake in the classic game Snake (all the snake's parts will follow the trail of its head). An example is shown in Fig. 3.

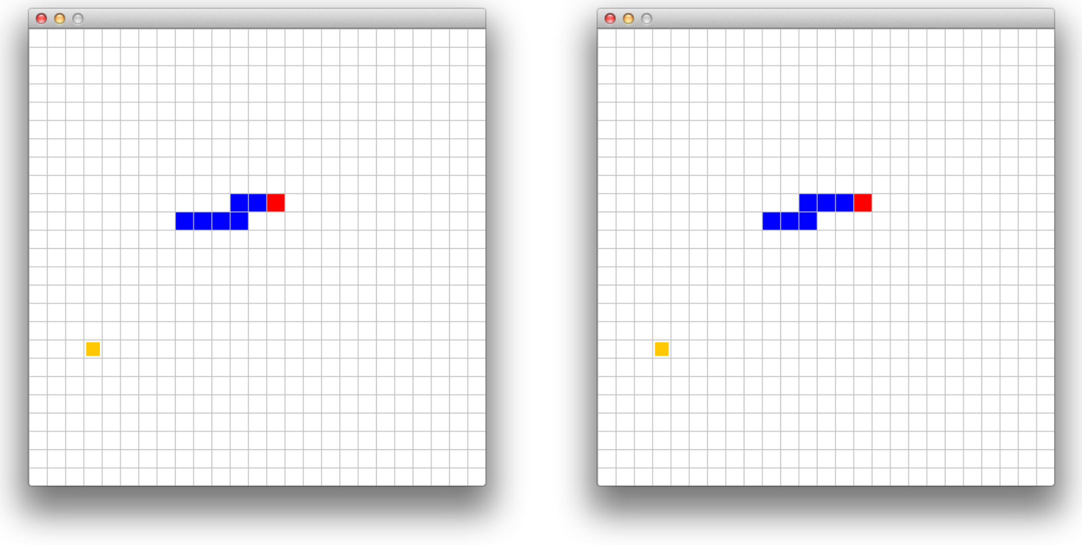
Figure 3: Snake Example Movement (Rightwards)

- A snake should be able to locate the food, turn and move to it intelligently. This is where you need to code a simple AI. You can assume that all snakes know where the food is exactly.
- A snake will consume food and grow like the classic game Snake. An example is given in Fig. 4. After a food is consumed, you need to immediately spawn a new food in a random empty position. A food is drawn as a small yellow square.

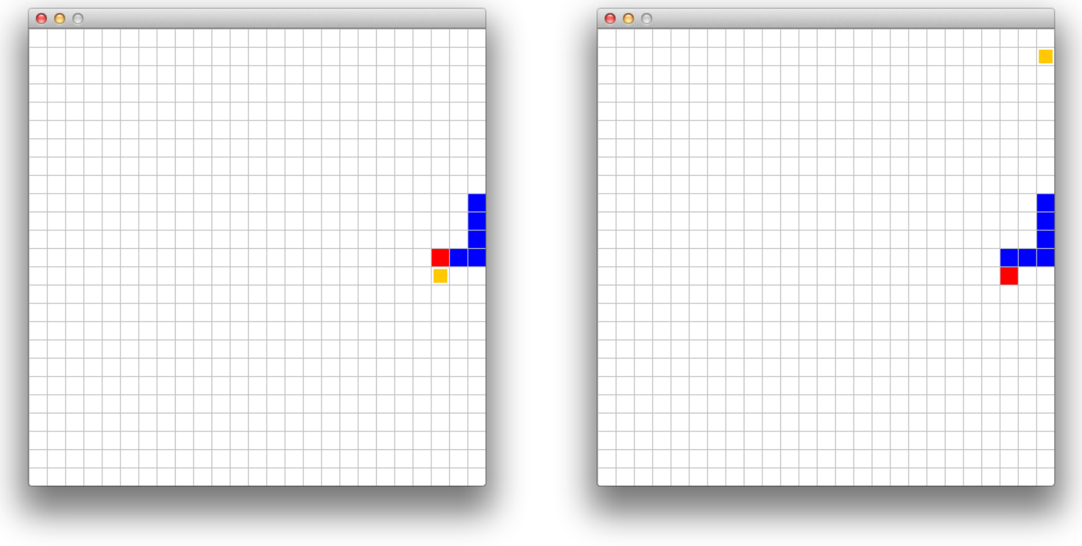
Figure 4: Snake consumes food and grows

- A snake <u>cannot</u> move over its own body or over another snake. Moreover, a snake cannot go beyond the world boundaries.
- If a snake reaches size 8, it will divide (reproduce) into two snakes of size 4. The newly born snake's head will be the old snake's tail. An example snake division is depicted in Fig. 5.

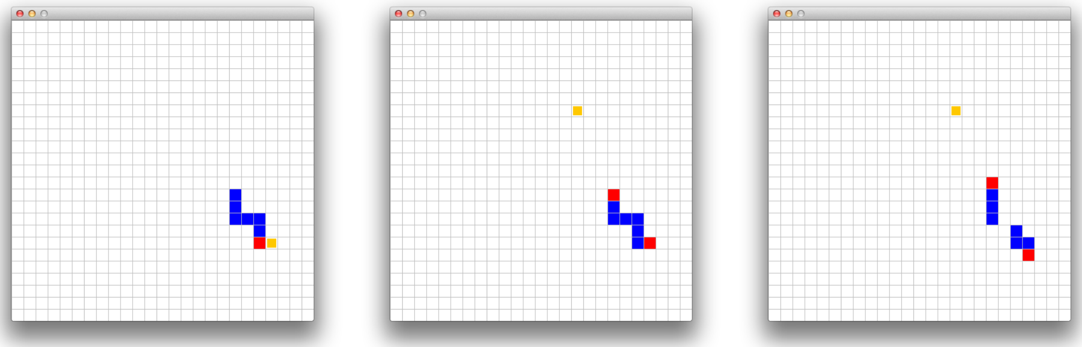

Figure 5: Snake consumes food and divides into two

Tips and Important Remarks

- Code incrementally. In other words, do not try to code everything at once, instead, implement a small feature at a time, test it, and only when you make sure it works, go on to implementing the next feature. For example, first implement a single snake moving around, then a more intelligent snake looking for food, then a snake growing and finally implement dividing snakes. If you do not follow this approach, you may face difficulties and may not have the chance to receive partial credits.
- You are required to write a project report, explaining your classes briefly, outlining the class hierarchy, including details about how your AI works, and explaining how to run your code and which parameters are configurable in the code (like world size and game speed - see below). You should also explain how you chose to represent your snakes and how you handled movement and dividing. The report should be in pdf format. You should put it under the `main` package (the directory where `Main.java` resides in), so that it can be submitted along with your code.
- You can create any number of classes and packages you'd like. Design is an important part of this project, so you should aim to have a modular and easily-understandable code structure. (Project 1 was a decent example of good design)
- The examples provided in this document are all for a world with size 25x25, but we expect your game to work with any sizes, just like the first project. We also want these sizes to be configurable (in case we wish to test with different parameters). Please specify how to set the world size from your code, in your project report.
- Similarly, the speed of your game should also be configurable. You should specify how to set it, in your report, as well.
- The colors and the actual look of your game may be slightly different from the examples in this document. As long as we can clearly understand which part is the snake's head, or which square is the food, it is OK. You are advised to explain these visual choices in your report as well.
- Your snakes may sometimes get stuck by curling up onto themselves, near a wall or sometimes even with each other. We want you to try to prevent this from happening as best as you can, but we do not expect you to prevent it completely since it is a really difficult problem. So in case one of your snakes gets stuck and unable to choose a direction to move, make sure that the program does not crash and does not go into an infinite loop. You may let the stuck snakes stay motionless while the game animation continues and other snakes continue moving and looking for food.
- You are required to comment your code in the Javadoc style, but you do not need to generate Javadoc htmls.