4CCS1PPA Programming Practice and Applications

# Coursework 3: Life

Michael Kölling and Jeffery Raphael

## Life

A *cellular automaton* (CA) is a grid of cells, where each cell has a finite state. Cells are given an initial state and follow a fixed set of rules to determine their subsequent state in the next generation (iteration) of the simulation. Normally, all cells use the same set of rules and the rules do not change over time. Lastly, the eight cells surrounding a cell are referred to as its *neighbours*. CA has applications in fields such as geography, anthropology, political science, sociology and physics.

CAs became popular after Martin Gardner described Life (commonly known as The Game of Life) in an issue of the Scientific American in 1970. The Game of Life was designed by John H. Conway, a Cambridge mathematician. In the game, a collection of cells, on a 2D grid, live and die according to a fixed set of rules. While the game is quite simple to describe, the complexity of the emergent behaviour is surprising and fascinating to watch. You can also play game here: https://playgameoflife.com/info

For this assignment, you will modify and extend the Game of Life simulation (available for download from KEATS) to make it more interesting. You are free to add additional classes or modify any existing class in the simulation. This assignment is a pair programming task. You must work in pairs. Information about pair programming will be provided.

## Pair Programming

The pair programming document on KEATS will provide you with the information on the structure of Assignment 3. Make sure you have read this document before continuing with Assignment 3. **Once you have found a partner, it is important that you both officially register your partnership on KEATS**, *Assignment 3: Partner Selection.* If you are unable to find a partner, speak to your TA and someone will be assigned to you randomly. It's recommended you spend at least 30 minutes discussing ideas and possibilities for your solution with your partner.

## Base Tasks

This coursework allows you to have a lot of flexibility on what you deliver — be creative!

- The base code comes with a simple life form, the Mycoplasma. Your first task is to make it more interesting. Modify its rule set so that it behaves in the following manner:
  - If the cell has fewer than two live neighbours it will die
  - If the cell has two or three live neighbours it will live on to the next generation
  - If the cell has more than three neighbours it will die
  - Lastly, any dead cell will come alive if it has exactly three neighbours

  **It's important to note that all cells determine their next state prior to the transition to their new state.**

- Using the Mycoplasma as an example, develop *at least* two new forms of life. Their rule set should be different from the Mycoplasma.
  - At least 1 life form should incorporate colour in its rule set, i.e., its colour may change while alive
  - At least 1 life form should exhibit different behaviours as time progresses

## Challenge Tasks

Once you have finished the base tasks, implement one or more challenge tasks. You can either choose from the following suggestions, or invent your own. You will be graded on a maximum of four challenge tasks.

- Non-deterministic cells. Cells execute the same set of rules during in generation. Implement a life form that behaves in a non-deterministic manner. That is, given a set of rules, the cell will execute a rule, $r_n$, with probability $p_n$.

- Symbiosis. Symbiosis is any long-term relationship between two different organisms. Implement two forms of life that either have a mutualistic or parasitic relationship. In a mutualistic relationship both life forms benefit while in a parasitic relationship only one benefits (the other is *harmed*).

- Disease. Modify the code so that some cells are occasionally infected with *disease*. There should be two elements to the disease (*i*) it can spreads from one cell to its neighbours and (*ii*) once in a diseased state the cell's behaviour changes.

## The Report

You must also write a report (*less than* four pages) describing your game. The report should contain the following.

- The names and student numbers of all students who worked on the submission
- A description of each life form that was implemented (i.e., the rules that govern their state)
- A list of each challenge task you completed and how you completed it.

## Submission and Deadline

- You must submit on Gradescope via KEATS by **Wed., Mar.** 1$^{st}$ **16:00 (4pm)**. You'll submit a zip file containing the following
    1. A Jar file of your BlueJ project.
        - You can create a Jar from within BlueJ by going to Project, and then "Create Jar File...".
        - You do not need to change any of the default options, and so you should just click the "Continue" button.
    2. A report (pdf file)
    3. All of your Java files
- **The Jar file must contain your source code, i.e., the *.java files, and it must run on BlueJ.**
- Click the 'Assignment 2 Submission Link' to submit your work. Follow all instructions in the 'Student Submission Guide'. If you have any trouble submitting your work, email Jeffery Raphael as soon as possible. Do not wait until the last hour to attempt your first submission.
- Marking details can be found in the 'Marking Rubric'.

### Late Submission Policy.

All coursework must be submitted on time. If you submit coursework late and have not applied for an extension or have not had a mitigating circumstances claim upheld, you will have an automatic penalty applied. If you submit late, but within 24 hours of the stated deadline, the work will be marked, and 10 raw marks will be deducted. If this deduction brings your mark for the assessment below the usual pass mark (40%), your assessment mark will be capped at the pass mark. **All work submitted more than 24 hours late will receive a mark of zero.**