Conway's Game of Life

HCI 2019-2020 Programming Assignment

Prof. Andrew D. Bagdanov

November 17, 2019

1 Overview

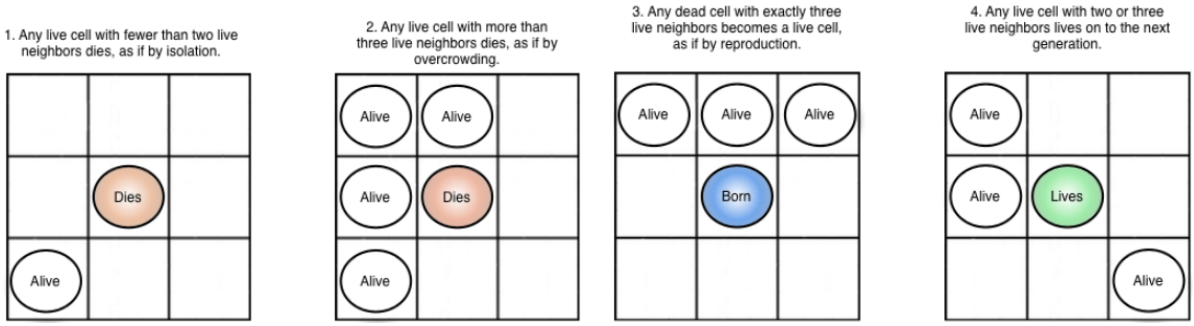
The **Game of Life** was invented in 1970 by the British mathematician John Horton Conway. Conway developed an interest in a problem which was made evident in the 1940's by mathematician John von Neumann, who aimed to find a hypothetical machine that had the ability to create copies of itself and was successful when he discovered a mathematical model for such a machine with very complicated rules on a rectangular grid. Thus, the Game of Life was Conway's way of simplifying von Neumann's ideas. It is the best-known example of a cellular automaton which is any system in which rules are applied to cells and their neighbors in a regular grid. Martin Gardner popularized the Game of Life by writing two articles for his column "Mathematical Games" in the journal Scientific American in 1970 and 1971. In this programming assignment you will implement the Game of Life, along with a complete and polished Graphical User Interface to control the simulation.

2 Rules of the Game

The game is played on a two-dimensional grid (or board). Each grid location is either empty or populated by a single cell. A location's **neighbors** are any cells in the surrounding **eight adjacent locations**. The simulation of starts from an **initial state** of populated locations and then **progresses through time**. The evolution of the board state is governed by a few simple rules:

1. Each populated location with one or zero neighbors dies (from loneliness).
2. Each populated location with four or more neighbors dies (from overpopulation).
3. Each populated location with two or three neighbors survives.
4. Each unpopulated location that becomes populated if it has exactly **three** populated neighbors.
5. All updates are performed simultaneously **in parallel**.

This figure illustrates the rules for cell death, survival, and birth:



3 Assignment

For this assignment you must implement a graphical simulation of Conway's Game of Life. You may implement this in **any programming language** and using **any GUI framework** you desire. Almost anything goes, however the features your GUI must implement to receive full credit:

- **A working visual simulation**: your GUI must implement and animate the Game of Life simulation. That is, it must start from an initial configuration (a grid of occupied and empty locations) and evolve that simulation through time. The GUI should animate – that is, **display** the current state of the board at every time setp – in a dedicated widget.
- **Start/pause/clear**: your GUI should support controls that allow the user to start and pause the simulation, and clear the current state of the board.
- **Variable framerate**: your GUI must support a control that allows the user to select the **framerate** at which the simulation is run and animated.
- **Drawing/editing of state**: your GUI must allow the user, starting from a **blank** board, to **draw** the initial state of the board (i.e. fill in occupied locations) with the mouse. This feature should also be available to the user whenever the simulation is **paused** (see previous feature), allowing the user to edit the current state before resuming the simulation.

Your GUI can also support the following features (which will be considered as **extra credit** in the final evaluation:

- **Loading of initial state**: your GUI might support the loading of a serialized version of the board state. This can be any file format you choose. If you implement this extra feature, you should also provide some **classic examples** of Game of Life Patterns that the user can play with. See http://www.conwaylife.com/wiki/Main_Page for many examples.
- **Zooming/panning of board**: your interface might allow the user to select the grid size (**zoom**) and which part of the grid is currently viewed (**panning**). If you implement this feature, you should also allow the user to select the grid size.
- **Cell history**: your animation of the Game of Life simulation might keep track of how long each cell has been **alive**. You can do this by remembering, at each location, how long it has been occupied and changing the color from **light blue** (newborn) to **bright red** (ancient).

4 Evaluation

You **must** submit the source code for your complete implementation by the **deadline for registering for the exam**. Late submissions will **NOT** be accepted.

A note on **cheating**. I realize that there are **many**, **MANY** implementations of the Game of Life out there. That is great, and you should look at them and learn from them. However, the code you submit **MUST BE YOUR OWN WORK**. Learn from others, but **write and submit your own implementation**.

Your implementation will be evaluated based on how you apply the programming models and constructs learned during the course. More specifically:

- **Functionality** (25%): your code must **work**. In your submission, include a minimal README that explains how to run your program (including any dependencies).
- **Code hygiene** (25%): keep things **clear**. This means writing concise, clear, and **reasonably documented** code. You should carefully identify **model**, **view**, and (maybe) **controller** components of your implementation.

- **Completeness** (25%): did you implement all of the **required** features? See the list above, but your GUI must implement all of the required features to earn full credit.
- **Correctness** (25%): does your GUI **correctly** implement all of the required features? Above all, your GUI should not crash and should operate consistently with the assignment requirements.
- **Extras** (max 10% bonus points): above and beyond. If you implement **extra features**, make sure you highlight them in the documentation for your submitted project.

5 Hints

Here I will collect some useful hints and advice for anyone choosing this programming assignment (this list might be updated, check back):

- The focus of this assignment is on **complete** and **correct** implementation using **best practices** (i.e. MVC). Your interface does not have to be **pretty**. Make it all **work**, make it work **correctly**, and **then** make it pretty if you have time.
- Conway's original Game of Life was defined over an **infinite** board. You are **NOT** required to implement an infinite board. Make the board **big enough** to be interesting, and small enough to not be a pain.
- Think twice about implementing your own **loop** over board locations when implementing your simulation. In Python, for example, the Numpy library implements **neighborhood** operations like **convolution** that you can use to implement the Game of Life Rules.

6 Useful Links

There are a **TON** of resources on Conway's Game of Life. It's almost pointless to list them here, just Google it. But anyway:

- The original article, archived and republished many, many times.
- A decent online implementation of the Game of Life. If your implementation works like this one (and is well-written and well-organized!), you're in good shape.
- An entire online community dedicated to the Game of Life and its variants.