Case Study: Game of Life

# Introduction

Game of Life is a cellular evolution game invented by mathematician John Horton Conway of Britain in 1970. It is a zero-player game in which the initial configuration/state determines the evolution and hence does not require any further input. Once the initial configuration is set, the user observes how the game evolves.

Game of life universe consists of a two-dimensional orthogonal grid of square cells. Each of the cells can be in one of the possible 2 states i.e. either alive or dead. Every cell has maximum of eight neighbors whose state decided the state of this cell.

The Game of Life evolves on the following principles:

1. Any live cell with fewer than two live neighbors dies, as if by underpopulation.
2. Any live cell with two or three live neighbors lives on to the next generation.
3. Any live cell with more than three live neighbors dies, as if by overpopulation.
4. Any dead cell with exactly three live neighbors becomes a live cell, as if by reproduction.

The initial state of the game is derived from the input seed values. The first generation is created by applying above set of rules instantaneously to all the cells in the universe. Note that birth & death happens at the same time and the distinct moment at which this occurs is also known as a tick. Each generation is a pure function of previous generation. Above set of rules are applied continuously to create future generations.

## Specification Requirement

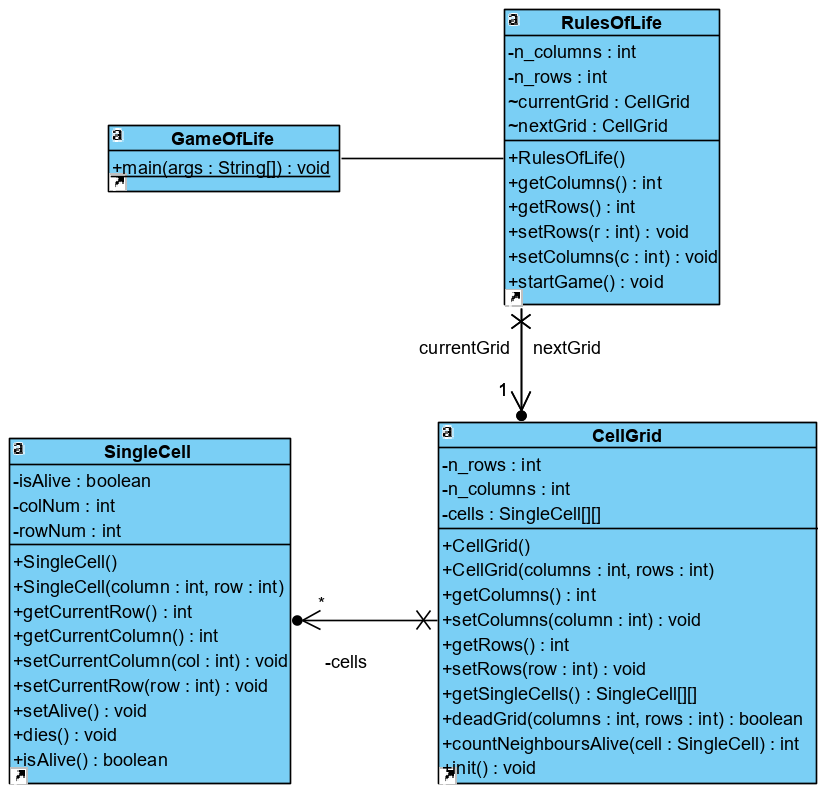
To develop any software specification requirement is the first step in the software development process. It helps the developer to understand the needs of the stakeholder and take care of any constraints if any. These requirements serve as a guiding force for developers in development of the software.  
Following are some of the specification requirement for Game of Life.

1. The software shall adhere to the “Game of Life” rules for evolution
2. The software shall display a 2-dimensional square grid of 51x51 cells for the evolution
3. The software shall display a cell in green color when it is alive
4. The software shall display a cell in white color when it is dead
5. The software shall input the seed values to the game as part of initialization
6. The software shall stop the game when all cells of a grid are dead
7. The software shall provide user buttons to close and minimize the window

We will take the above limited set of requirements to explain how it helps in design of the software.

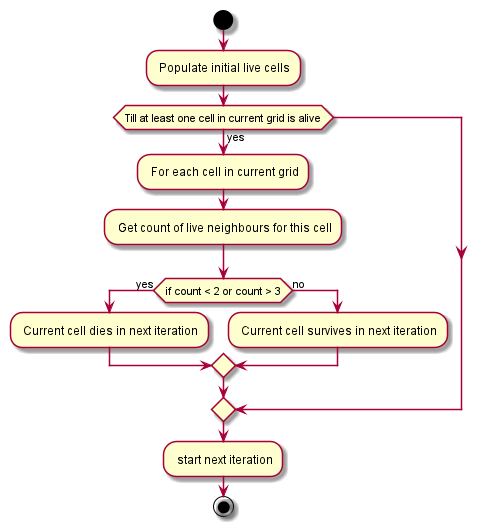
## Software Design

In this section we will explain on how we derive at the software design by using the above-mentioned specification requirements. To satisfy requirement (1) we will need to define a class named “RulesOfLife” which will implement the “Game of Life” rules. Similarly, for requirement (2) we shall define “CellGrid” class which shall provide user the 2-dimensional square grid in which cells will be arranged orthogonally. Requirements (3) and (4) will be supported by “SingleCell” class which will define helpers at cell level. The super class which will manager the overall game is defined as “GameOfLife” class. This will in turn use services offered by above mentioned classes to run the game. Requirement (5) and (6) will also be satisfied by “CellGrid” class while (7) will be handled by “GameOfLife” class. For better understanding below is a very simplistic class diagram showing the relations between different classes.

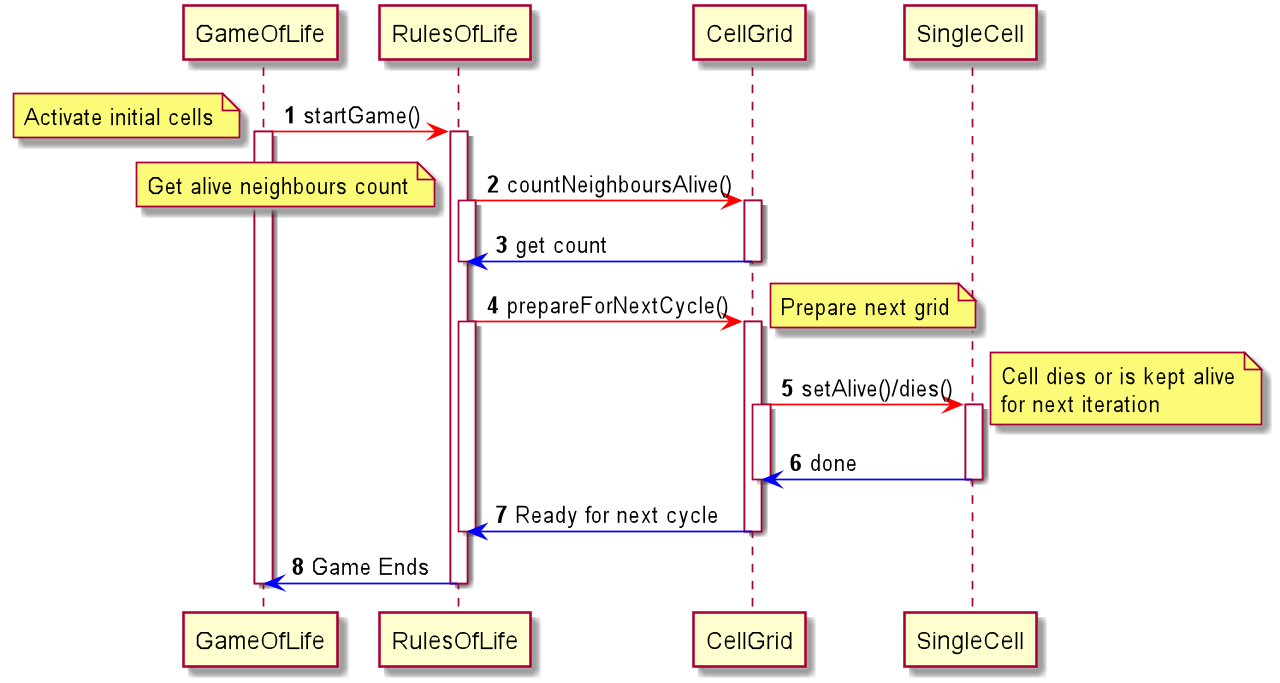


The above class diagram shows the relationships between the classes and some of the services offered by them. It provides a high-level view of the components involved in software design.

**Activity diagram of the Game of Life is as below.**

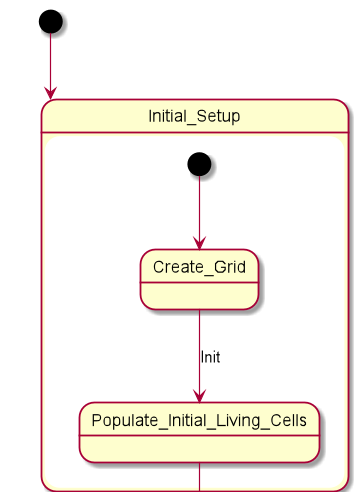


Above activity diagram provides the dynamic design information for the Game of Life. The flow helps the developer to conceptualize the software design.

**Sequence diagram of Game of Life is as be**

Sequence diagram mentioned above provides the sequencing information for the API called by the Game of Life. This dynamic design information helps all stakeholders understand the software in a better way.

Now we will show how the states are maintained in Game of Life. The initial state of the game is depicted by below State Diagram.



To show how the above state behaves with respect to code refer below screenshot from the game itself.

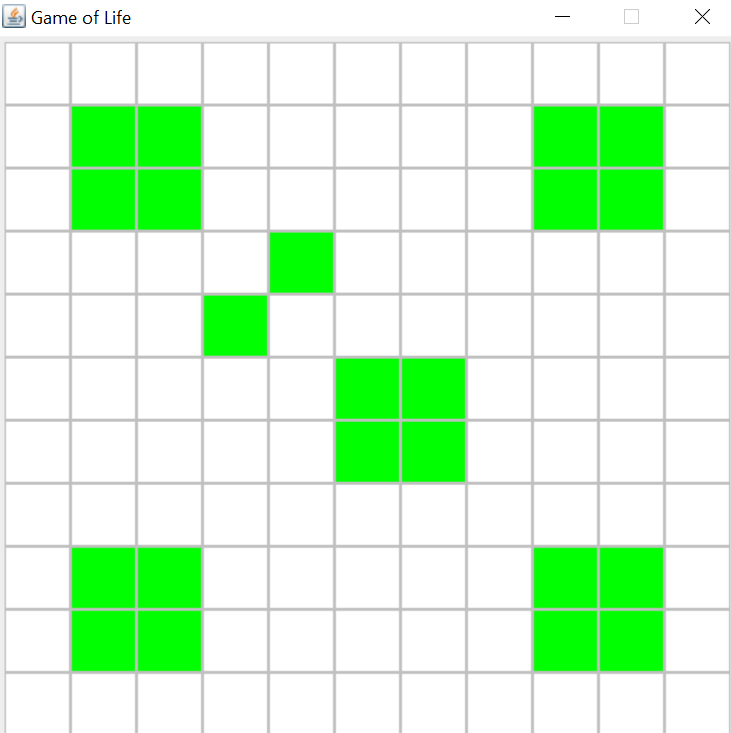
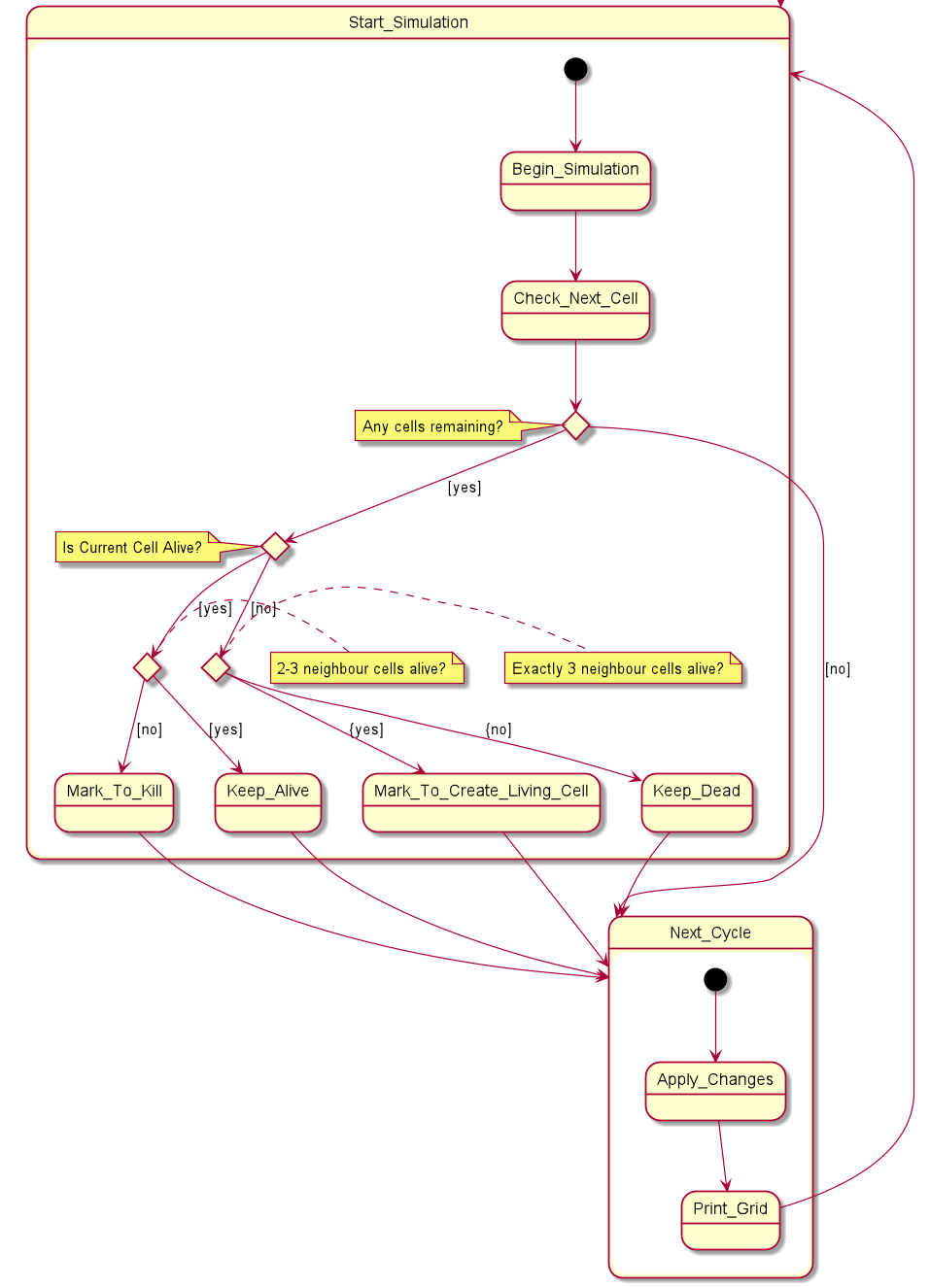


Figure : Initial State

We will now check the next state diagram which will show how the game progresses.



The next diagram shows the corresponding state in actual code.

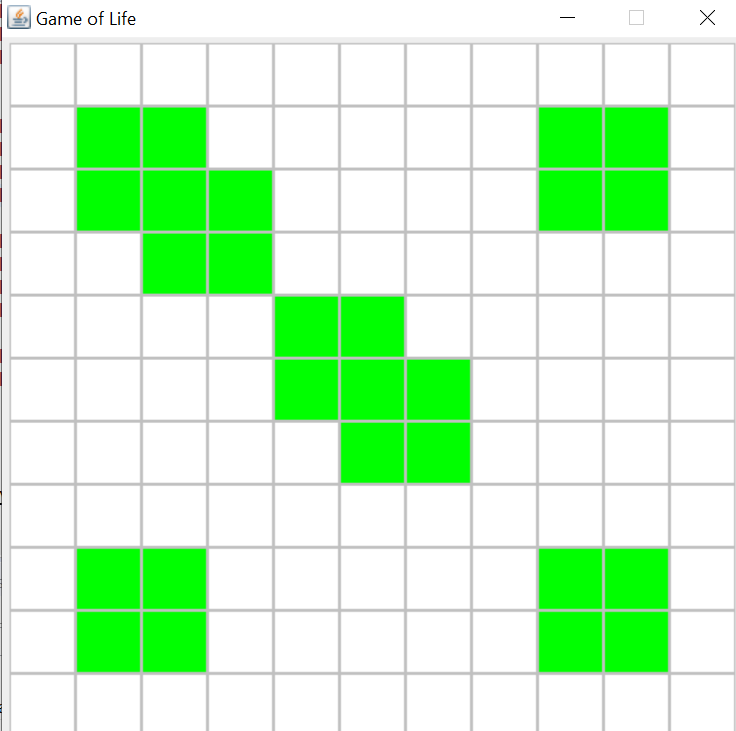


Figure : Life after 2 ticks

Below screenshots give more information on the progression of life in the game after third and fourth ticks.

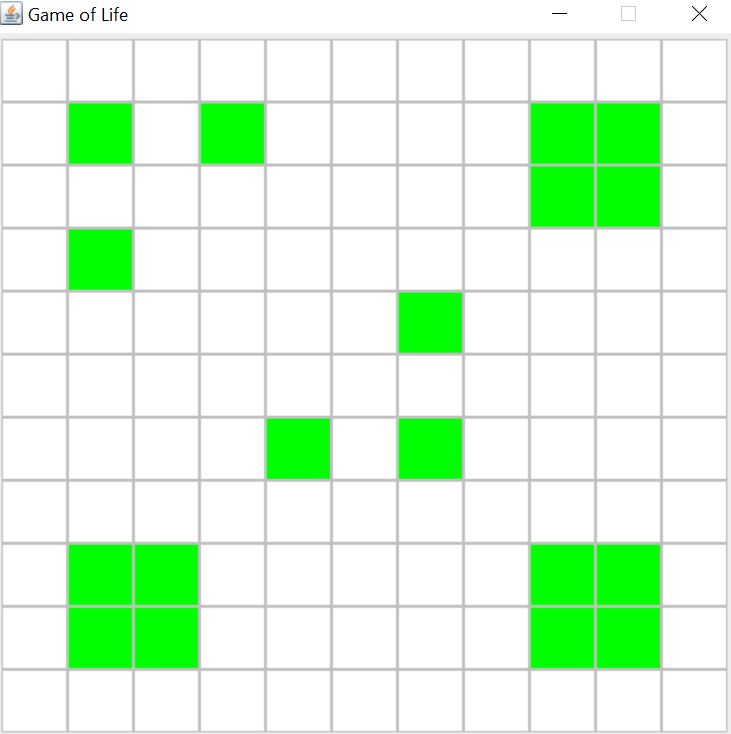


Figure : Life after 3 ticks

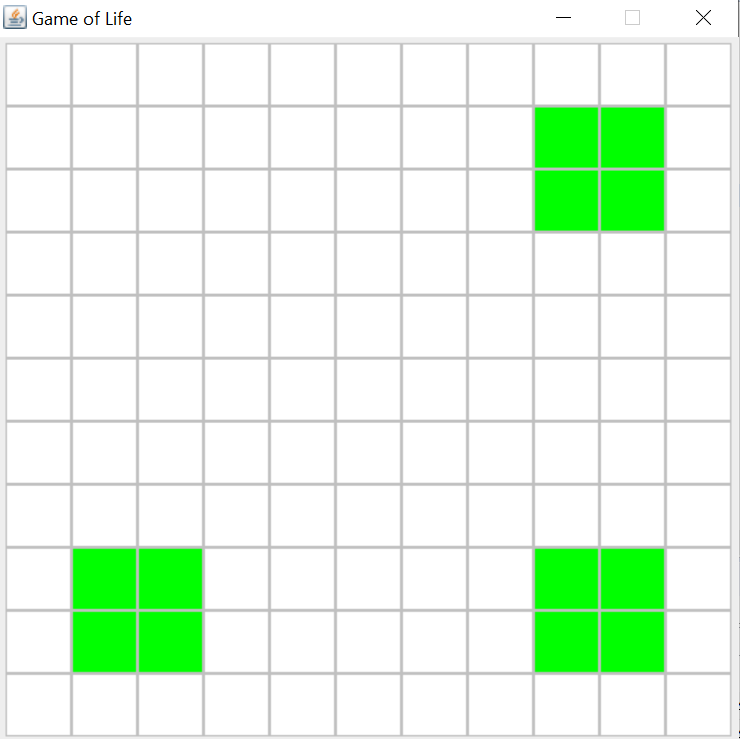


Figure : Life after 4 ticks

As explained in this section, the software should start with specification requirements and then followed by software design. The software design provides the template for developer which is later used in software implementation. The requirements can also be used for unit and integration testing.

In conclusion, software architecture and design helps developer understand and implement the software in cleaner and effective way.