Instituto Superior de Engenharia de Lisboa

Konrad Wodnicki

49027

Tech Report

“Game of Life”

Modeling and Programming

20.06.2021

1. Introduction

This document is a project report for the subject Modeling and Programming at ISEL in the summer semester 2021. In this project one of the first cellular automata entitled "Game of Life" is implemented. The project is not fully realized at the moment its console version is prepared.

2. Target Audience

People interested in physics, mathematics or computer science, science in general, who want to see visually how this automaton works.

3. Project description

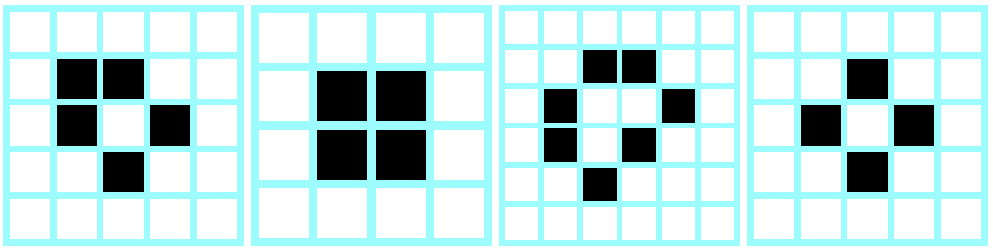
a. Game of life (Life) one of the earliest and most famous examples of cellular automata, created by John Conway in 190. From the moment the game became popular it attracted a lot of interest because of the surprising way in which structures can evolute. To this day, many people are still interested in the way complex structures are created using a few simple rules. The game is played on a board divided into square cells. Each cell has eight "neighbors", i.e., cells that are adjacent to it by sides or corners. Each cell can be in one of two states, it can be alive or dead. The state of a cell in the previous time unit is used to calculate the state of all cells in the next time unit. The state of a cell depends only on the number of its living neighbors.

b. Principles:
1. A dead cell with exactly 3 living neighbors becomes alive in the next unit of time (is born).
2. A living cell with 2 or 3 living neighbors remains alive, with a different number of neighbors it dies.

```
if (alive && (numberOfNeighbors == 2 || numberOfNeighbors == 3))
    return true;
else return !alive && numberOfNeighbors == 3;
```

Principles implementation

c. Oscillators and Constans.
1. Constans - are structures that remain identical regardless of the time step. Here are some examples.

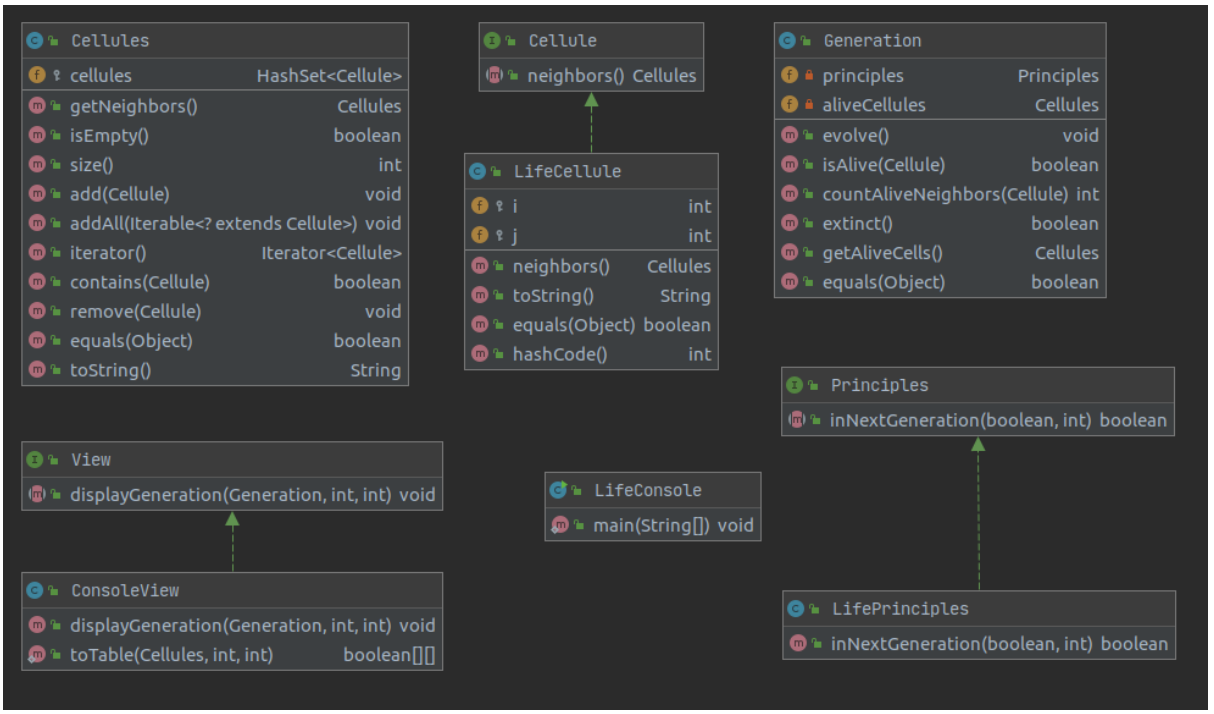


Some examples of constants

b. Oscillators, they change periodically, periodically returning to the original state, the simplest structure consists of three live cells in a row.

c. Unsteady, ships etc. Several other structures can be distinguished, but unlike the two above they will not be used by us to test this version of the application.

4. Design



UML class diagram

LifeCellule - represents the cells of the cellular automaton, thanks to the neighbors() function it allows to return neighbors, which is used to calculate its state in the next generation.

Cellules - Hashset Cellule, stores the cells of the automaton that are used.

Generation - the most important method is evolve(), which allows to create the next generation of cells according to the rules.

LifePrinciples - the rules for this automaton, whose observations we can see above.

ConsoleView - class allowing us to display our automaton in console, which allows us to observe the changes.

LifeConsole - class containing main method.

5. Design Patterns

a) Abstract Factory
1. Cellule implements it with LifeCellule, so we can change the cell behavior to other cells without changing the game logic.
2. Principles is implemented by LifePrinciples, thanks to this we can change the cell generation rules without changing the game logic.
3. View allows us to change how the cell generation is displayed, this will be useful for the next stage of the project.

6. Testing

a. Unit Tests
-notEquals()
-sameOrder()
-differentOrder()
-testNoNeighbors()
-testNeighbors()
-testEquals()
-diesBecauseOfUnderpopulation()
-staysAlive()
-diesBecauseOfOverpopulation()aCellIsBorn()

b. Testing with oscillators and nonconstants in the main() method.
-Block Test
-Boat Test
-Blinker Test

7. References

https://pl.wikipedia.org/wiki/Gra_w_%C5%BCycie
https://en.wikipedia.org/wiki/Conway%27s_Game_of_Life

8. Repository

https://github.com/konradw98/isel-mop-life-game