Le Mans Université

UFR Sciences et Techniques
Département Informatique

ANDROID APPLICATION REPORT

Programmation d'interfaces embarquées
Master 1 Informatique
2018/2019

Teyssier Titouan, Pelloin Valentin

March 14, 2019

Contents

1 | Our application : Visual Life Configurator

Our application is a cellular automaton simulation game. The player is free to create new automatons with the rules we want (for example Conway's game of life, Wireworld, ...). For each automaton, the player configures how the automaton work, the types of cells, how neighbours are calculated, and the rules for a cell to be transformed into another one.
Then, the player can create as many worlds he wants, edit them, and play with them.
Our application is called `VLC`, it stands for Visual Life Configurator.

1.1 Persona

Name John Doe

Age 21

About John studies computer science. He is interested into algorithms and life simulation. John likes playing games on his phone while waiting for his bus while going to the University.

Goals

- Learn new things
- Build things with Legos

1.2 User stories

- As a player, I want to entertain myself by playing a game ;
- As a player, I want to learn new automaton rules ;
- As a player, I want to experiment new configurations of cellular automatons ;
- As a player, I want to express myself by creating and building worlds

2 | Functional study

2.1 Storyboards

Below, the mock-ups of our application:

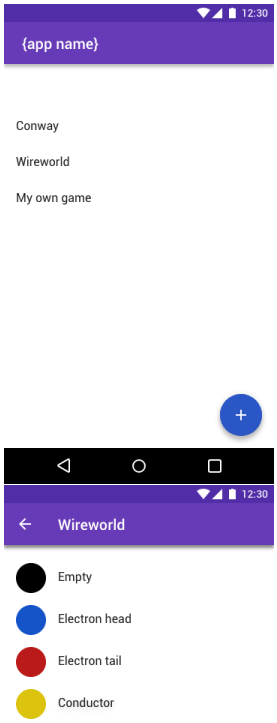

Figure 2.1 – The main activity of VLC.
The player can find all the automatons he created, and click on them to edit them or to play on a game.
The player can also click on the "plus" button to create and configure a new automaton.

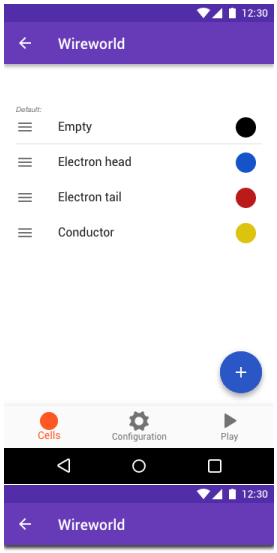

Figure 2.2 – Inside an automaton, cells tab.
The player can create and remove cell types for this automaton. He can reorganize the order of the cells. The first cell type is the default one, the type of the cell that will fill the grid when creating a blank world.

Figure 2.3 – Inside an automaton, configuration tab.
In this tab, the player can access the configuration for each cell.

Figure 2.4 – Inside an automaton, play tab.
In this tab, the user can access to its saved games. He can also choose to play on a random world.

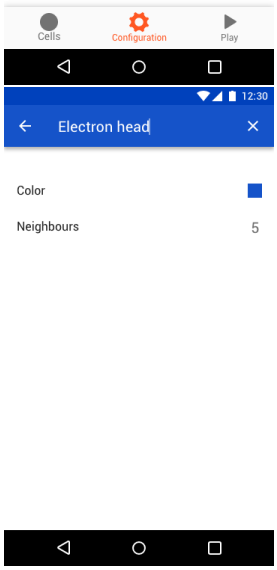

Figure 2.5 – Cell parameter activity.
For this specific cell type, the user can edit its name, and its color. He can also access to the neighbours choosing screen.

Figure 2.6 – Cell neighbours choosing.
On this activity, the player can choose which cells are being counted by the automaton as neighbours for this specific cell type.

Figure 2.7 – Cell type rules edition.
For this cell type (blue one), the player configured the rules of transformation. For example, if there are 8 neighbours around a blue cell, it will be transformed to a yellow cell in the next generation.

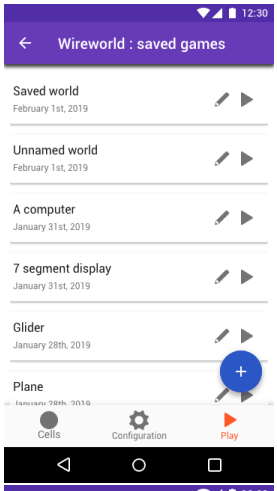

Figure 2.8 – List of saved games.
The player can list worlds that are existing for an automaton. For each world, he can edit it, or play on it. By clicking on the "plus" button, he can also create a new blank world.

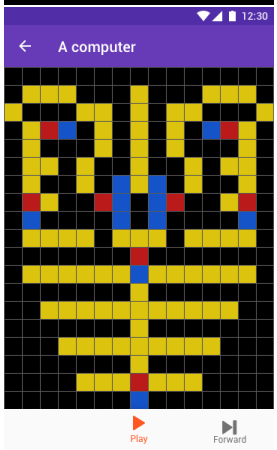

Figure 2.9 – Playing game activity : paused.
By clicking on "Forward", the player can generate the next step.

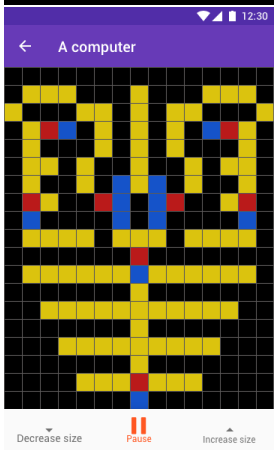

Figure 2.10 – Playing game activity : playing.
The player can zoom in or zoom out.

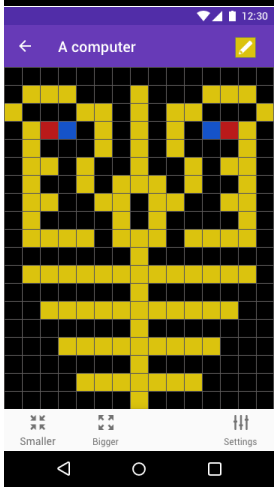

Figure 2.11 – Editing a game.
The player can drag on cells to color them.

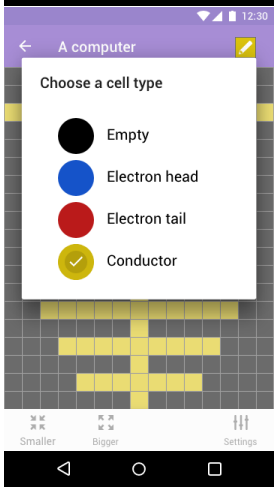

Figure 2.12 – Editing a game.
The player clicked on the pen in toolbar. He can change the color of cells he will be placing.

A demonstration of these mock-ups can be seen at `https://github.com/valentinp72/VisualLifeConfigurator/blob/master/docs/mock-ups/demo.mp4?raw=true`.

2.2 Architecture

Our project architecture is based on an MVC (Model-View-Controller) pattern. Below, a class diagram of it.

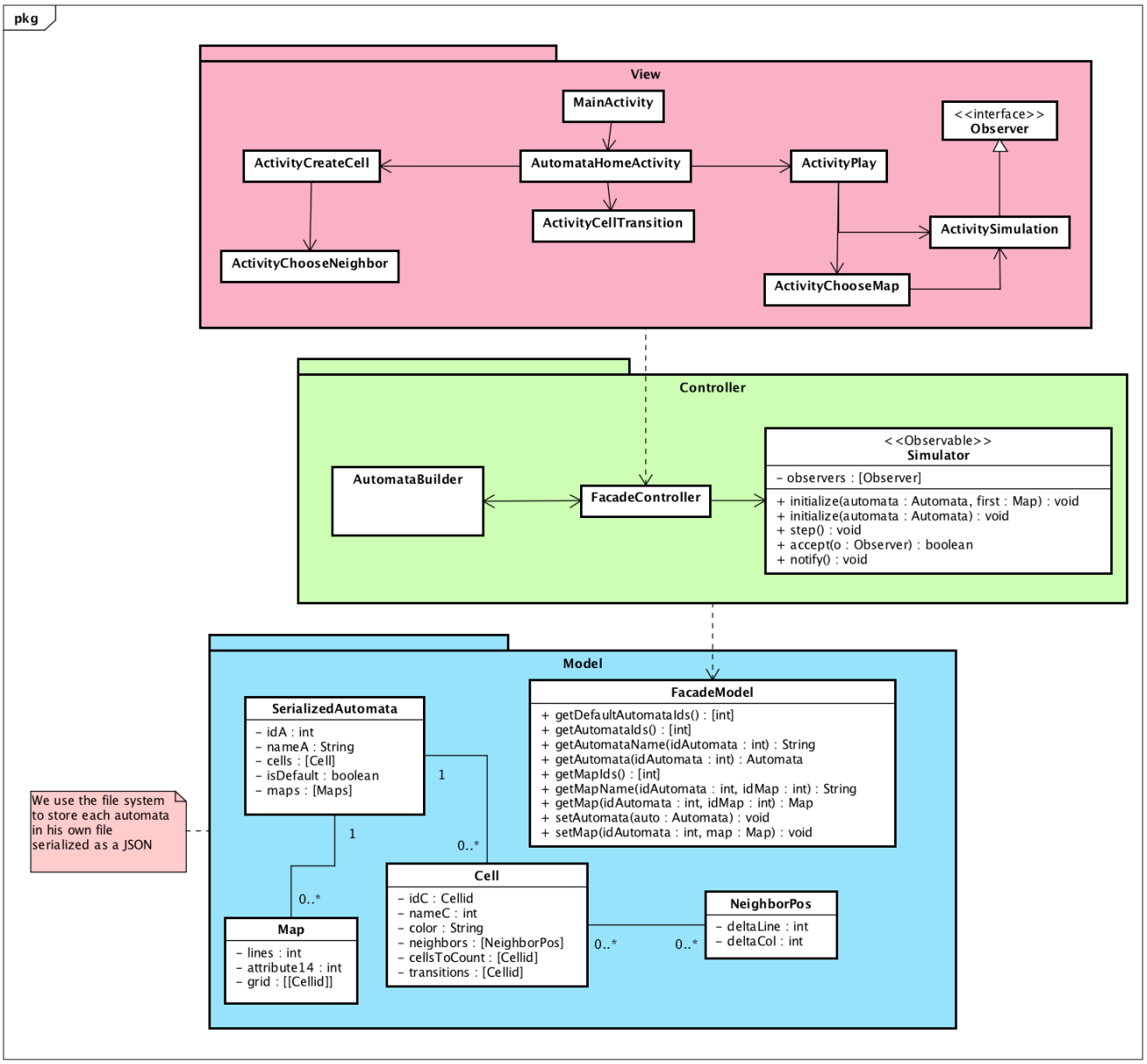

Figure 2.13 – Class diagram of our application

2.3 Github link

Our project is accessible at `https://github.com/valentinp72/VisualLifeConfigurator`.

3 Technical details

3.1 Persistence

Our application saves maps and automatons on the phone using internal files. The `Persistence` class is responsible of loading and saving objects from the device. Automatons are saved in a XML format, whereas Maps contents are saved in plain text files, lowering the size of the files.
Default maps and automatons are given in a Asset folder (read-only). When the user needs to reset these games, he can click on a reset button on the home screen.

3.2 Data communication between activities

Once loaded, a Map, a Cell or an Automaton is given to the activities needing them as `Serialized` Extras. With hindsight, everything except Maps should be saved in a SQLite Database. This way, we would not have any problem of modified Serializable objects not being known by every activity. This would also prevent many bugs and many hacks to prevent them from happening, since we actually need to resend some data after edition to other activities, and for some function it does not work (for example when creating a new Automaton or a new Cell).

3.3 Simulator operation

The `Simulator` works by loaded a configured Automaton and Map, and play the map for this Automaton. Since every map can be played with every possible automaton, it is first converted into a compatible map for the given automaton. A `SimulatorThread` is created for running the game on a loop with interfering with the UI responsiveness.

4 | Achieved work

4.1 Current application status

Currently, the application support loading and saving maps from both the assets and the internal files. Cells can be edited (name, color, neighbours to use) and configured (transitions when a specific number of cells is neighbouring cell type). When editing or configuring a Cell, the toolbar color and status bar color are updated according to the cell color, to allow a better understanding of the application by the user.

Saved maps are present in the game, and can be played by the user (even maps not initially compatible with the automaton). The user is also allowed to play random maps, even though he is not able to save it for the moment. For the moment, it's also not possible to edit a map before paying it, but it should be rather simple.
The application is available French and in English.

4.2 Tasks assignment

Both Titouan and Valentin did storyboards and architecture study. Programming-wise, Valentin did the user interface and persistence module, whereas Titouan did the back-end operations (maps, automatons, neighbours, simulator, ...).

5 | Conclusion

This project allowed us to learn to create an Android Application that could potentially be published to the public through the Play Store. It showed us how complete and developed Android is, but in the meantime, it disappointed us about some important features (data assets folder that is read-only and internal storage that can't be easily filled with default files for example).

6 | Appendix : application screenshots