Document metric

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Project | Cellular automaton | |  | Company: | MiNI PW |  |
|  |  |  |  |  |  |  |
| Name: | Buisness Analysis | |  |  |  |  |
|  |  | |  |  |  |  |
| Topics: | Requirements specification | |  |  |  |  |
|  |  | | |  |  |  |
| Author: | Aleksander Lipka | | |  |  |  |
|  |  | |  |  |  |  |
| File: | Requirements - Cellular automaton.pdf | |  |  |  |  |
|  |  |  |  |  |  |  |
| Version no: | *v 1.0* | Status: | Requirements | Final date: | 2016-06-16 |  |
| specification |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |  |
| Summary: | Specifiy the requirements demanded by the course | | | | |  |
|  |  |  |  | Last |  |  |
| Authorized by |  |  |  | modification | 2016-03-24 |  |
|  |  |  |  | date: |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
| Version | **Date** | **Who** | **Description** |
| 0.1 | 2016-03-10 | Aleksander Lipka | Definition of the main purpose of  the document + specification |
| 0.2 | 2016-03-11 | Aleksander Lipka | Added missing points to specification – schedule, dictionary, evaluation and risk |
| 1.0 | 2016-03-24 | Aleksander Lipka | Final Check |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

0. Schedule

* 2016-02-25 Thursday Lab 1 Start, theme’s offer, preliminary choice
* 2016-03-03 Thursday Lab 2
* 2016-03-10 Thursday Lab 3 requirement specification technical
* 2016-03-17 Thursday Lab 4 requirement specification - second term for LS group Thursday 24
* 2016-04-07 Thursday Lab 6 technical project
* 2016-04-14 Thursday Lab 7 technical project - second term
* 2016-04-21 Thursday Lab 8
* 2016-04-28 Thursday Lab 9 code of modules
* 2016-05-05 Thursday Lab 10 version 0.98
* 2016-05-12 Thursday Lab 11 version 0.99
* 2016-05-19 Thursday Lab 12
* 2016-06-02 Thursday Lab 13 version 1.0 testing
* 2016-06-09 Thursday Lab 14 test report
* 2016-06-16 Thursday Lab 15 acceptation, final grades

1. Overview

The project aims to develop the application that will simulate a cellular automaton with a predefined size of neighborhood relative to specified cell. The application shall visualize the step-by-step simulation according to initial configuration.

2. Dictionary

- **Cellular Automation** – a cellular automation is a finite collection of cells located in a grid, each in one of a finite number of states. In this project there will be only three states: dead, alive, inactive,

- **Rule** – a statement that makes cells either alive or dead, rules determines the states of cells and their neighbors,

- **Neighborhood** – set of nearby cells to the current one,

- **Step** – next or previous function call.

- **GUI** – Graphical User Interface, visible and interactive part of the program

3. Requirements

Functional:

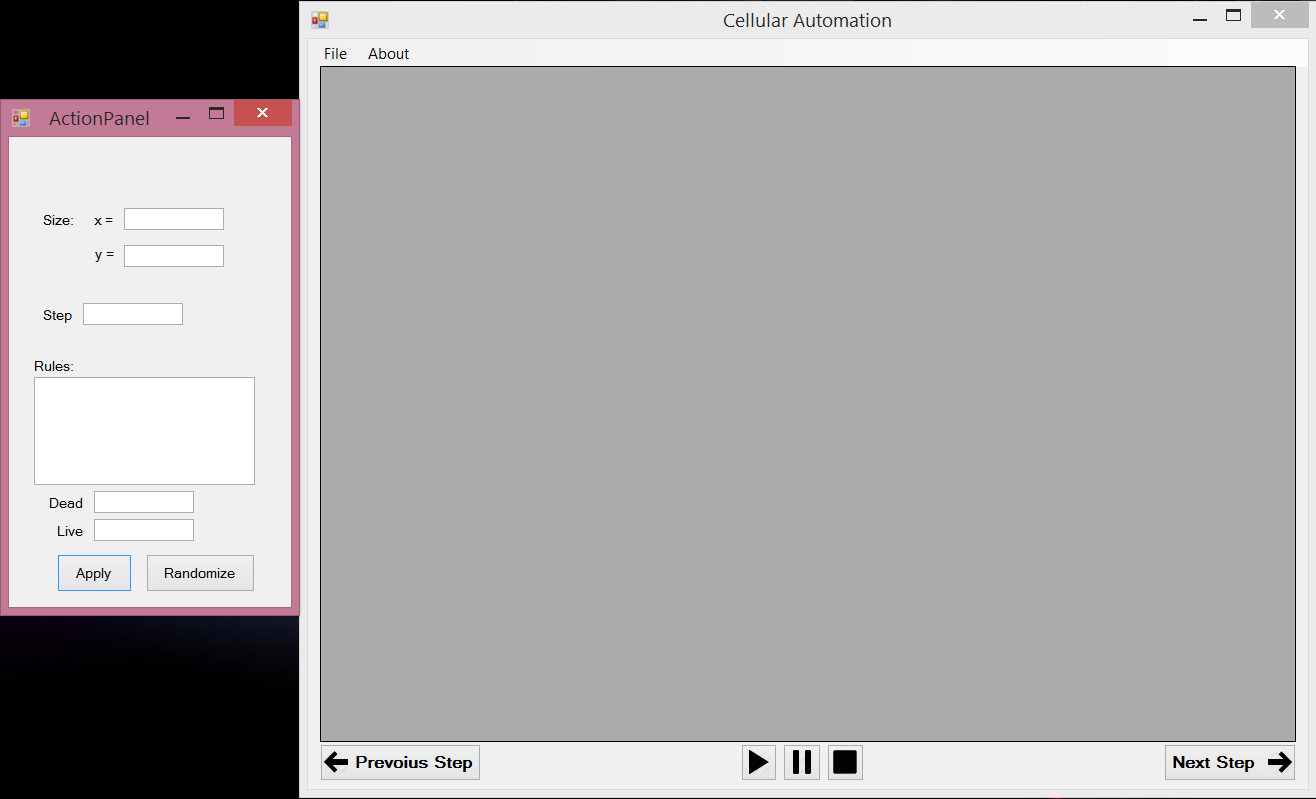
* Application shall be divided into the grid and a navigation/action panel
* Grid can be in any finite number of dimensions
* Cells in grid must have a finite number of states
* When the state of the cell is changed, the cell changes its color
* One cell will have 24 points neighborhood
* User can apply new rules from user action panel
* Application shall distinguish contrary rules and solve the problems
* Application shall start with the predefined set of rules and a starting state
* Application should run without any problem in our faculty laboratories
* Application should be presented in the executable file format

Non-functional:

* The error handler will be applied to rules maker so that the program won’t brake under the wrong syntax
* When the user will want to exit the program, the save prompt will appear first
* The program should run smoothly without any visible delays
* Application’s GUI should be done in English
* New/Load/Save options should be included in the menu bar

GUI will be composed out of two main windows – one with the grid and one with the panel with all action and navigational controls. Below is the list of all controls added to each window with a brief explanation:

4. Front-End Interface (GUI)



Picture 1 Preview(may change)

1. **Main panel:**

- matrix (with default size) 🡪 main grid, where everything will happen

- matrix fill the whole window (except for navigational bar and menu strip)

- menu strip (File -> New) 🡪Resets everything to its default state

(File -> Load visualization) 🡪Loading some saved visualization

(File -> Save visualization) 🡪Saving current visualization

- play-pause-stop buttons 🡪 action buttons for visualization

- next/previous button 🡪 One step back or forward in the visualization

2. **Settings Panel:**

- textboxes for:

* Size of matrix 🡪 height and width of the matrix
* Step 🡪the step needed to apply the rule
* Rules 🡪 place to enter custom rules (rules have to be in chosen lang. syntax)
* Live 🡪 no. of alive cells in the rule
* Dead 🡪 no. of dead cells in the rule

- apply button 🡪 Button to apply all the changed rules and settings

- randomize button 🡪 Enters random number and data into rules

5. Supplementary specification

Non-functional features are the features that does not have anything to do with the main program algorithm. They are useful for user and sometimes necessary for the program to work correctly.

Additional features in my program:

- **Stop button** 🡪 permanently stops current visualization (resets the board)

- **Pause button** 🡪 pauses current visualization

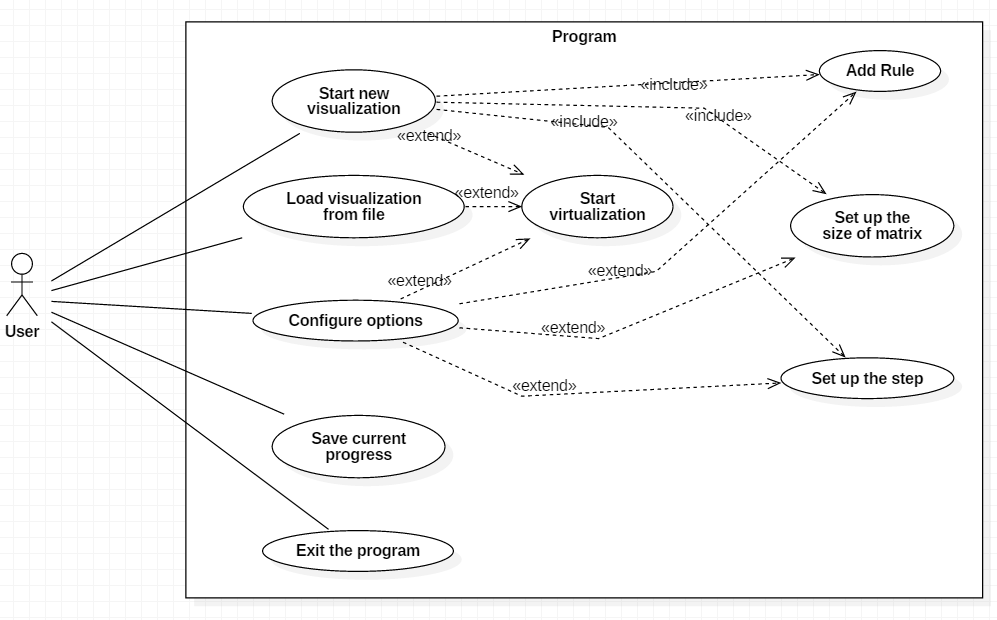
- **Play button** 🡪 start or resumes visualization

- **About menu button** 🡪 the about section will have the information about the current version of the program as well as the manual for the application itself

- **“Make a step” button** 🡪 (while in pause/stop mode) makes one step of the grid

- **“Reverse a step” button** 🡪 (while in pause/stop mode) go back with the previous step

- **Randomize button** 🡪 fills rule making section with randomize data and rules



6. UseCase Diagram

Above Use Case diagram shows us the way the user can communicate with the program. We assume that the program is already launched. The only things user can do right after the start are: create, load or edit visualization and also save current progress or exit the application. Depending on what user chose the options he can then start the visualization or customize the settings.

7. Solution evaluation

The project will take in total approximately 120 hours of work. 20 hours for specification, another 20 hours for technical specification. Project code implementation will take about 80 hours including the testing part. No delays are taken into the account.

8. Risk Analysis

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Risk | Probablity (1-5) | Severity (1-5) | Score (PxS) | Action to Prevent / Manage Risk |
| Human |  |  |  |  |
| Illness | 3 | 2 | 6 | Systematical workflow |
| Organisational |  |  |  |  |
| Failure to meet schedule deadline | 3 | 3 | 9 | Systematical workflow, good time management |
| Technical |  |  |  |  |
| Software output is unusable | 1 | 4 | 4 | Develop a wider expertise to the problem |
| Fatal crash | 2 | 4 | 8 | Testing |
| Legal |  |  |  |  |
| Intelectual property right issues | 1 | 2 | 2 | Consult used resources |