London Metropolitan University

Faculty of Life Sciences and Computing

CU6051 Introduction to Artificial Intelligence

Emergence: Cellular Automata

Documentation

Name:

ID Number:

Date: Wednesday, 01 January 2014

# Contents

[1 User Documentation 1](#_Toc376003344)

[1.1 Conway’s Game of Life 1](#_Toc376003345)

[1.1.1 Tab: Home 1](#_Toc376003346)

[1.1.2 Tab: Patterns 1](#_Toc376003347)

[1.1.3 Tab: Graphics 1](#_Toc376003348)

[1.1.4 Tab: Task 1](#_Toc376003349)

[1.2 Wolfram’s Rule 30 2](#_Toc376003350)

[2 Technical Documentation 3](#_Toc376003351)

[2.1 Conway’s Game of Life 3](#_Toc376003352)

[2.2 Wolfram’s Rule 30 3](#_Toc376003353)

[3 Conclusion 4](#_Toc376003354)

[4 References 5](#_Toc376003355)

[51. Appendix 6](#_Toc376003356)

[5.11.1. Code Listings 6](#_Toc376003357)

# Table of Figures

[Figure 1 – create\_rfmon.sh 3](#_Toc376003460)

# User Documentation

This section describes the Graphic User Interface and its functionalities of developed web application. The web application presents itself by drawing a grid with dimensions matching the browser window size. The user interface is divided into two dialog windows. Following chapters describes operations of controls for each window.

## Conway’s Game of Life

Both dialog windows use Tabbed Document Interface, called “Tabs”, which allows multiple documents to be contained in a single window. Conway’s Game of Life window contains Home, Patterns, Graphics and Task tabs. Each tab is composed of group of menu buttons. Following sections in chapter 1.1 describes the main modes of the web application. Application controls are explained through the description of modes.

### Plain Mode

Tab Home has four groups of controls: Selection Mode, Navigation Controls, Speed and Stats.

In this mode user can use “Navigation Controls” to start and stop the game, or step forward through the game by pressing button “Next Generation”. Grid can be cleared using button “Clear”. Speed of redrawing the grid can be manually changed using the slider or by entering the value instead of the orange number and pressing enter. The last group contains non-editable statistics about the game, for example number of past generations.

This mode allows loading of patterns using drop-down list from tab “Patterns”. Application provides 34 patterns. All patterns have a word “glider” in the name.

Tab “Graphics” allows user to resize the grid using spinner. Spinner generally enhances a text input for entering numeric values with up/down buttons and arrow key handling. In addition to this functionality user can manually rewrite the numbers. By pressing ENTER user confirms his attention to resize the grid. The current version of this application initializes the grid on every resize event. That causes the loss of the current pattern on the grid.

The last feature is “Display Mode”, where user can select between four different display modes. The default mode is “Rectangle”, which displays rectangles with white [inside with] black border for dead cell, and black inside of rectangle for live cell.

continue description of display modes...

#### Create Pattern Mode

This is first half of the solution to the given task.

#### Create Glider Mode

ghfhtr

### Tab: Task

This tab describes the task. Task is to create an interactive implementation of the Game of Life grid, where a player can select starting points for a variety of blinkers. The system then sends gliders to destroy them, resulting in chaos. Alternatively, system randomly sets some blinkers and player sets starting point for gliders. Documentation should clearly explain algorithms used to generate the life-forms (chapters 2.1 and 2.2).

## Wolfram’s Rule 30

Both researchers and

# Technical Documentation

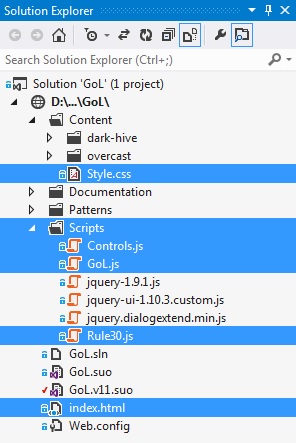
Technical documentation describes algorithms and main thoughts behind the produced solution. Documentation of the solution is introduced by chapter 2.1 Code Organization, which outlines how the code in these files fits together.

## Code Organization

The whole development of this project has been tracked using Git. Software source code has been uploaded to GitHub server where the functional website is hosted too [1].

Development branch ‘CleanWebsite’ has following file structure displayed on Figure 1. Files highlighted with blue color are files containing all developed source code.

Figure - Code Organisation

File ‘index.html’ is a main file and loads all JavaScript and Cascading Style Sheets. The controls created by main file are managed by JQuery code contained in ‘Controls.js’ and ‘Rule30.js’. ‘Controls.js’ manages Conway’s Game of Life dialog functionality and ‘Rule30.js’ controls Wolfram’s Rule 30 dialog window. This Jquery code initializes the whole website and activates appropriate functions chosen by the user.

The underlying code which draws the canvas, catches the user’s click and executes the algorithms (of interest) is contained in file ‘GoL.js’, which is a shortcut for ‘Game of Life’.

The original source code was initially downloaded (forked) from GitHub [2]. Since [then] most of the code has been changed. At present the only thing remained is the main skeleton of objects, few variables and methods. I have downloaded 34 patterns and kept them within the solution.

## Graphics

The grid has been drawn using this equation :

## Conway’s Game of Life

Speed: time delay is recalculated to "speed" with scale from 1 to 100. This speed is similar to frame rate.

## Wolfram’s Rule 30

vsvsa

**Error! Reference source not found.** deletes the interface.

Figure 2 – create\_rfmon.sh





fsdds

# Conclusion

This project was the first attempt to develop a system of this nature. I aimed to achieve developing a tool that will recognize Probe Request attacks in a real time. Furthermore, I have analyzed wireless traffic captured on a home wireless network and used supervised feed forward neural network to determine the results. I have also considered the ethical side of development of this tool.

It was identified from the beginning that producing a complete result would be impossible within the given timeframe. Software solution meets the functionality in accordance to the design strategies determined becoming a starting point for researching in this area. The working system has been created using many programming techniques and at least three programming languages. However, as it has been demonstrated in previous chapter, training neural network has been partly successful.

Personally I consider this project a success if the ideas described in the report can become a useful reference for a future work on the subject.

# References

[1] <http://pavolondzik.github.io/GoL/>

# Appendix

## Code Listings

APPENDIX A



