

Game Of Life Project

User Manual

By: Aaron Zachariah

Table of Contents

Overview of Application Features	3
Starting the Application	3
User Input.....	4
File Formatting.....	5
Part 1: The Dimensions	5
Part 2: The Contents	5
Examples	6
Menu Bar Features	8
Settings	8
View	9
Toolbar Features	10
Reset Button	10
Single Step Button.....	10
Full Step Button	10
Configuration Window.....	11
Errors.....	12
Illegal Argument Error	12
Number Format Error.....	12
IO Error	12

Overview of Application Features

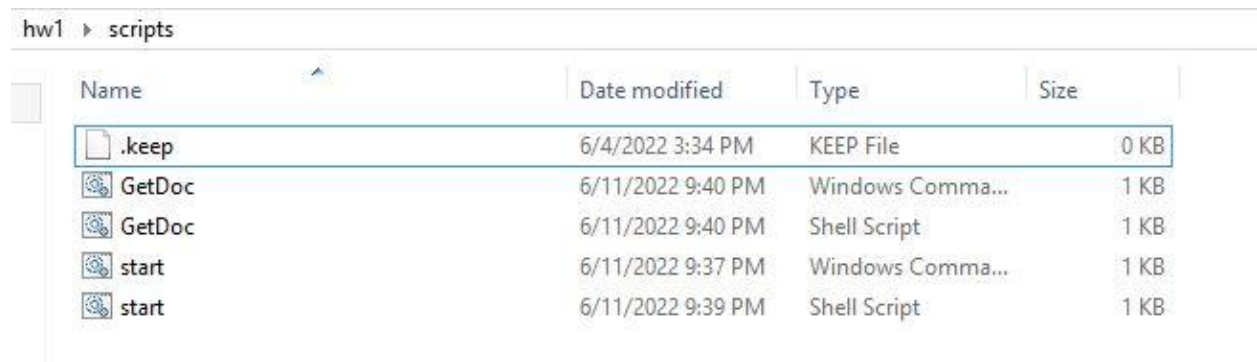
This project is an implementation that simulates Conway's Game of Life. Conway's Game of Life is a cellular automaton that models the evolution of a colony of some organisms that follow a set of four simple rules:

1. Any live cell with fewer than two live neighbors dies, as if caused 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 simulation takes some initial "seed file" which will describe the initial state of the Game of Life universe. The four rules above are applied to the grid to generate the state of the next generation. Essentially, each generation is a function of the preceding one. The rules are repeatedly applied to generate future generations. The number of generations generated, and the initial state of the grid is specified by the User (see "User I/O" for more details)

Starting the Application

Running the application is as easy as running a single script file! To start the application, first navigate to the scripts folder.



Name	Date modified	Type	Size
.keep	6/4/2022 3:34 PM	KEEP File	0 KB
GetDoc	6/11/2022 9:40 PM	Windows Comma...	1 KB
GetDoc	6/11/2022 9:40 PM	Shell Script	1 KB
start	6/11/2022 9:37 PM	Windows Comma...	1 KB
start	6/11/2022 9:39 PM	Shell Script	1 KB

Inside the scripts folder there will be two sets of scripts. The script named "start" will run the application. If using Windows, simply double click the Windows Command Script, and the application will start. If on a Unix system (MacOS, Linux) use the Shell Script to start the application. Of course, the simulation will not run without the proper User input, which will be discussed more in the next section.

User Input

The simulations requires three important user input before it can start. First it will ask for the input file, then the output file pattern, and finally the number of iterations.

Input File: This is the input file, which details the size of the grid, and its initial state. This file should be placed in the bin directory, as the program will check the bin directory for the specified input file. Refer to the section on “File Formatting” to see how to format the input file, as the format is key to building and running the simulation. Please include the file extension (e.g. If you place “input.txt” in the bin folder, type “input.txt” when asked)

Output Filename Pattern: This is the label that the program will use when generating output files. Each generation’s state will be written to a text file. The naming convention for the output files is as follows:

<filename pattern><iteration #>.txt

When specifying the pattern name, there is no need to include the file extension. All output files will be placed in the bin directory (the same folder at the input file)

Iteration Number: This is the number of generations that the application can simulate. This is configurable via a drop-down menu.

Here is an example of how the bin folder would look after running the application:

Name	Date modified	Type	Size
main	6/4/2022 4:37 PM	File folder	
.keep	6/4/2022 3:34 PM	KEEP File	0 KB
input	6/9/2022 8:42 PM	Text Document	1 KB
out0	6/12/2022 11:36 PM	Text Document	1 KB
out1	6/12/2022 11:36 PM	Text Document	1 KB
out2	6/12/2022 11:36 PM	Text Document	1 KB

Input and output files are located in the bin folder and follow the naming conventions above.

File Formatting

User input regarding the output and duration are straightforward, but the input file's formatting is very important, as a poorly formatted file will be detected by the application and cause an error to occur. See the section on "Errors" for more information about problems and their solutions.

The input file can be broken down into two parts. The first part details the size of the grid, and the second part details the contents of the grid.

Part 1: The Dimensions

The grid shape is a matrix, so to determine the size, the row and column counts are required. The first line should hold two numbers, the first of which is the row count, and the second being the column count. Both numbers must be positive integers. These numbers must be separated by commas and/or spaces. For readability, it would be best to format it like so:

5, 7

The above line would be interpreted as 5 rows and 7 columns.

Part 2: The Contents

After the first line containing the dimensions, the next lines should contain the initial contents of the grid. Just like the first line, all contents related to the grid must be separated by spaces and/or commas. The data should be represented in a matrix like shape, with the number of lines matching the specified row count, and the number of elements on each line matching the specified column count. Each element should be a 0 or 1, indicating if the cell is dead or alive respectively.

NOTE: The program does not enforce the user to input only 0/1s, however any cell that does not have a 1 will be treated as a dead cell. The standard convention would be to use only 0/1s.

Examples

```
5, 7 <-grid dimensions
0, 0, 0, 0, 0, 0, 0
0, 0, 0, 1, 0, 0, 0
0, 0, 0, 1, 1, 1, 1
1, 0, 0, 0, 1, 1, 1
1, 1, 1, 1, 1, 1, 1
```

grid contents

The file above is an example of a properly formatted grid. There are 5 lines of grid contents, and each line has 7 elements. Both counts match the data on the first line. All elements are separated by commas and spaces.

Here some examples of improper formatting:

```
5|7
0, 0, 0, 0, 0, 0, 0
0, 0, 0, 1, 0, 0, 0
0, 0, 0, 1, 1, 1, 1
1, 0, 0, 0, 1, 1, 1
1, 1, 1, 1, 1, 1, 1
```

Bad Example 1: The dimensions are not separated by spaces and/or commas

```
5, 7
0, 0, 0, 1, 0, 0, 0
0, 0, 0, 1, 1, 1, 1
1, 0, 0, 0, 1, 1, 1
1, 1, 1, 1, 1, 1, 1
```

Bad Example 2: The row count doesn't match the specified number of rows

```
5, 7
A, B, C, 0, 0, 0, 0
0, 0, 0, 1, 0, 0, 0
0, 0, 0, 1, 1, 1, 1
1, 0, 0, 0, 1, 1, 1
1, 1, 1, 1, 1, 1, 1
```

Bad Example 3: The grid contents contain non-integers, which will cause an error

```

5, 7
0, 0, 0, 0, 4, 2, 3
0, 0, 0, 1, 0, 0, 0
0, 0, 0, 1, 1, 1, 1
1, 0, 0, 0, 1, 1, 1
1, 1, 1, 1, 1, 1, 1

```

Bad Example 4: This won't cause errors but doesn't follow the standard

```

5, -1
0, 0, 0, 0, 0, 0, 0
0, 0, 0, 1, 0, 0, 0
0, 0, 0, 1, 1, 1, 1
1, 0, 0, 0, 1, 1, 1
1, 1, 1, 1, 1, 1, 1

```

Bad Example 5: Impossible to have negative number of columns

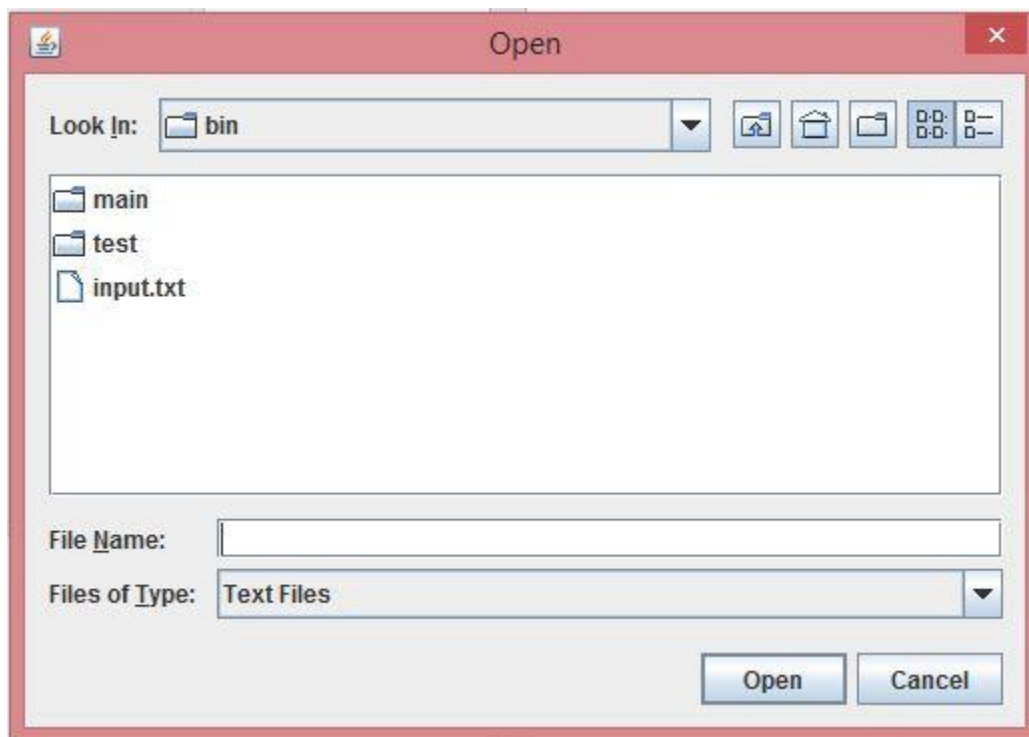
These are just a few examples of common possible formatting errors. Please refers to the section on “Errors” to see a detailed breakdown of errors and their fixes.

Menu Bar Features

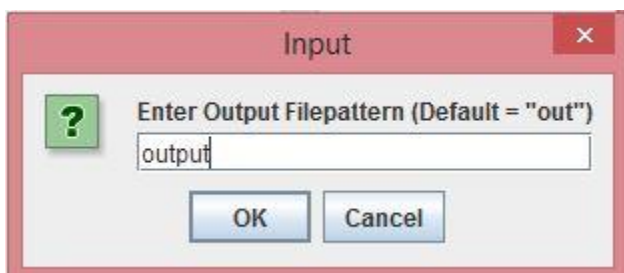
The Menu Bar offers a variety of configuration options for the user. Using the menu bar, the user can change various application parameters and modify the look of the program.

Settings

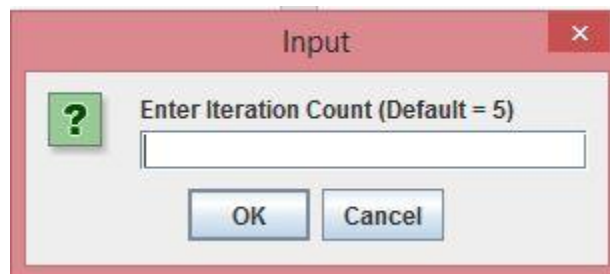
Select Input File – Selecting this option prompts a pop up to appear, which will let the user choose an input file to load into the grid. Make sure that the file formatting is done in the manner explained in the previous section, as incorrect formatting will prompt an error message to appear.



Select Output Pattern – Choosing this option will allow the user to set the naming pattern to use when writing to files. Usage of the pattern name is described in the section regarding User Input.



Set Iteration Count – Choosing this option allows the user to configure the maximum number of generations to simulate. The application will allow the user to simulate up to the specified number of generations, at which point the user will have to reset the simulation.

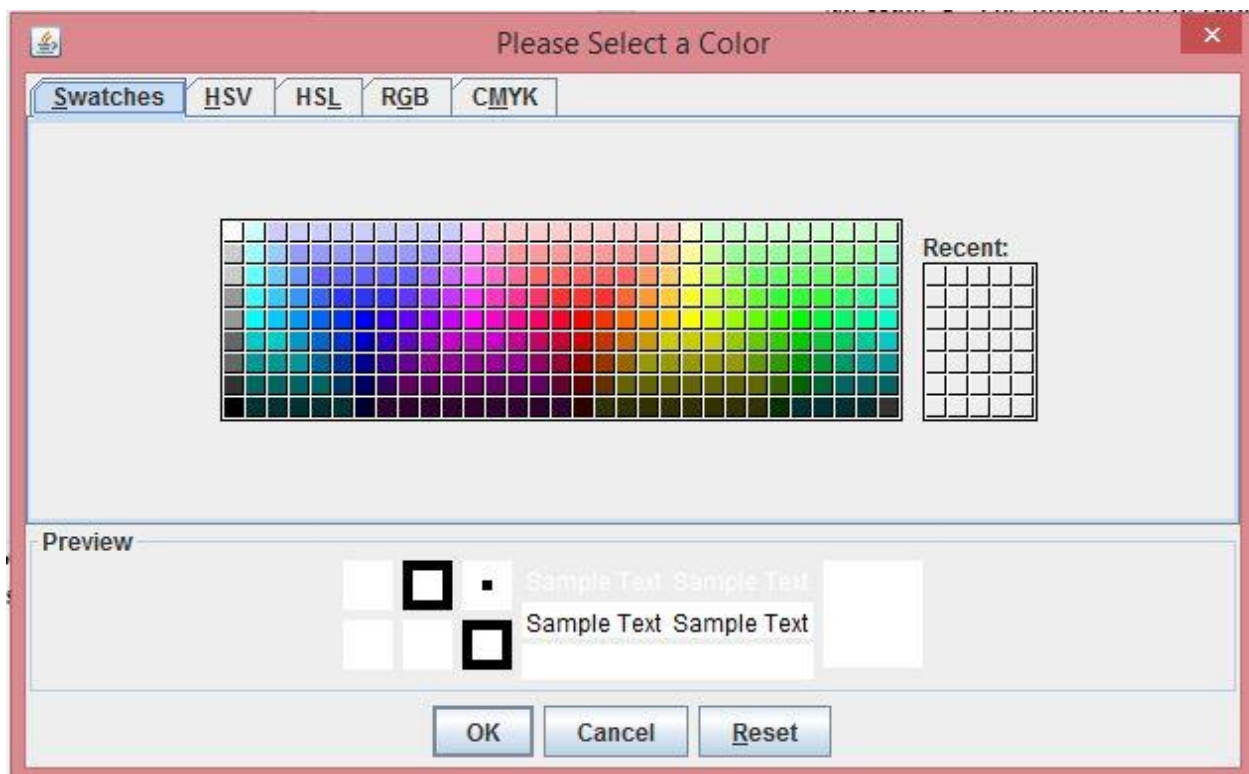


Save Lower/Upper Tick - Choosing either of these options allows the user to set the bound for the generations which should be saved. By default, the application will save an output file for every generation simulated. This can be modified, such that only a subset of generations are written to output files.

View

The View drop down menu allows the user to change the look of the UI.

Set Background Color – If this option is chosen, a pop up appears, with various ways to choose a Color to be the background color for the simulation. By default, the background is white, but this can be changes to almost any color. Note that the other information, like text will never change, so be sure to choose a color which allows the information to be visible.

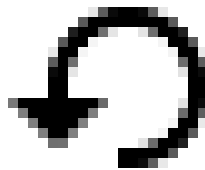


Toolbar Features

Just below the grid interface, there is a useful toolbar which contains various buttons for commonly used commands. This section will provide insight on each function and how they work

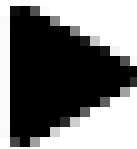
Reset Button

This button is for resetting the state of the simulation. If no file has been previously loaded into the application, it will reset all cells to deceased, even if the user had configured the grid using the configure function. Once an input file has been loaded, the reset function will always reload that file in its initial state.



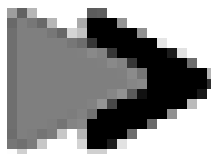
Single Step Button

This button is for simulating one generation of the simulation. It can be clicked multiple times, but once the generation limit is reached, it will do nothing. The output files will be written to and stored based on the user configurations covered in the previous section.



Full Step Button

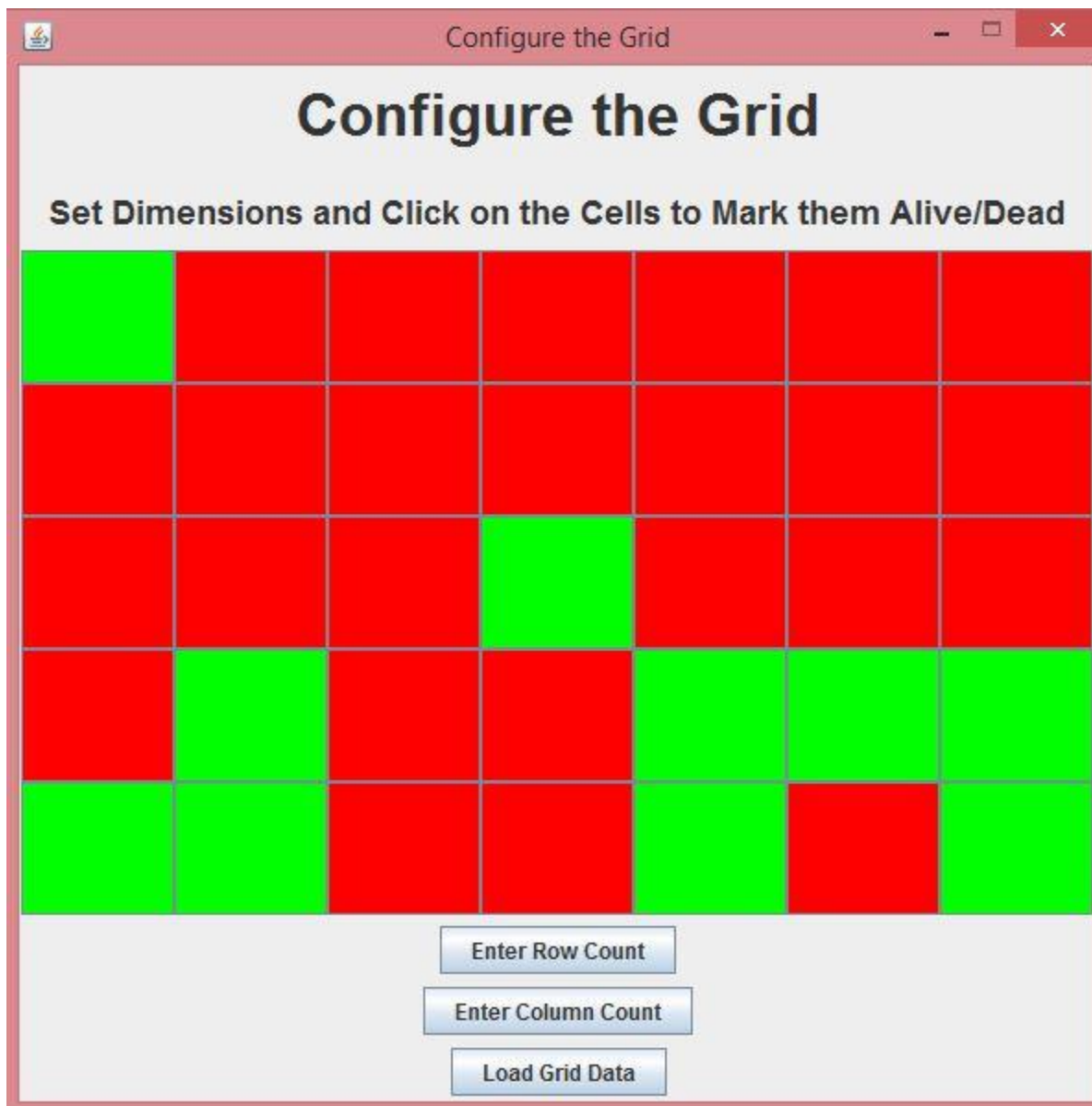
This button will simulate all the way to the end of the simulation in one step. All output files will be written accordingly, and the only thing left to do would be to reset the simulation.



Configuration Window

Clicking the button “Configure” will open up another window, where the user can configure the initial state of the grid without having to create a text file. This feature can only be used at generation 0, so make sure to reset before using the Configure option. When the window opens, the user must first set the row and column count via the buttons on the bottom. After, a fully dead grid will appear, with the specified dimensions. The user may click on the various cells to set their initial state (alive or dead). Once the user is happy with their selection, simply click load grid data, and the grid in the main simulation will update. You can freely run the simulation from there.

Below is an example usage. Simply click “Load Grid Data” and the simulation will update.



Errors

There are various errors related to faulty user input that could occur. This section lists some error types and error messages you might see, and how to fix them

Illegal Argument Error

This is a type of miscellaneous error, which occurs when the user inputs some faulty input. For example, if the user inputs the lower bound of the tick range to a negative number, this will prompt an error, since a negative value simply doesn't make sense in the context of the simulation. The error message will let the user know what was wrong.



Number Format Error

This is an error when the application expects an integer, but the user does not type an integer. A prompt will appear, letting the user know that an integer is expected.



IO Error

This error will occur when there is a problem building the grid from the selected input file. This is most commonly caused by mistyping the file contents or simply selecting the wrong file.

Fix: Make sure the file formatting follows the standard outlined in the formatting section. Also confirm that the file is the correct input file.

