Test Manual

Game of Life Project

By: Aaron Zachariah

Table of Contents

Testing Scope

The tests for this application aim to test all functionality provided and ensure that the correct output is generated given some correct input. These tests must also show that the application is able to handle incorrect input and will fail with an appropriate error message or exception. Consequently, given some invalid input, the application must fail, and should not generate some invalid output.

Testing Strategy

There is a variety of user input and grid types that can either be accepted or should be throwing errors. To make sure that the application handles each case in the correct manner, there must be a concrete strategy on the various aspects of the program execution that must be tested

User Input

Reading input from the user is the first key feature that should be tested. The application prompts the user for various pieces of information – these are important features which effect the application's execution. Each aspect of input must be tested, first to ensure that the application reacts normally when given valid input, and also to ensure that an appropriate error is throws when given faulty input. See the section on "Test Scenarios" for more information

File Input

File input is the other key aspect of user input that must be tested thoroughly to ensure the applications functionality. The file is made up of two parts. The first part details the grid's dimensions, and the second part details the grid's contents. Naturally, the specified dimensions must match the actual dimensions of the grid's contents. Both the grid dimensions and contents must be formatted in the correct order. Therefore, a good testing strategy would be to ensure that valid files produce matching grids, and that invalid files produce an error.

Testing Scenarios

Below is a section detailing different types of testing scenarios, their input, and their expected output. These testing scenarios aim to test all the features of the Game of Life application, and test various types of valid and invalid input. When testing certain aspects of user input, make sure all other aspects of user input is correct, to prevent certain problems from covering other problems. These scenarios largely cover the UI, as the simulation engine is tested separately.

User/GUI Input

Input Dialog Boxes

Description: Test whether the application handles input validation correctly

Test Cases:

1. Test system behavior when given an input that is NaN
2. Test system behavior when given an input that is a negative value

Output File Pattern

Test Cases:

1. Test system behavior when given a typical output file pattern
2. Test system behavior when given atypical output file pattern (ex: pressing 'Enter' before typing the pattern)
3. Test system behavior when including the file extension

Button Presses

Test Cases:

1. Test system behavior when trying to simulate past the max generation
2. Test system behavior when trying to reset at the 0^{th} generation
3. Test system behavior when trying to configure before resetting
4. Test system behavior when trying to load a grid via the configuration option
5. Test system behavior when reset is clicked without inputting aa file.

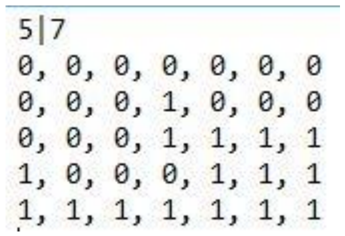
File Input

Grid Dimensions

Test Cases:

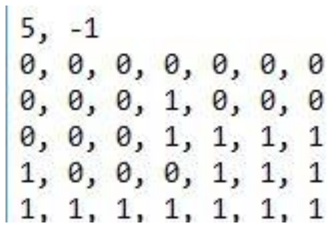
1. Test system behavior when grid dimensions are positive integers and delimited properly
2. Test system behavior when grid dimensions are not delimited properly
3. Test system behavior when grid dimensions contain nonnumeric values
4. Test system behavior when grid dimensions are 0
5. Test system behavior when grid dimensions are negative
6. Below is example file input for some of the above test cases:

Below is example file input for some of the above test cases:



```
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
```

Test Case #2



```
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
```

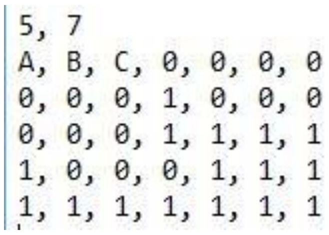
Test Case #5

File Contents

Test Cases:

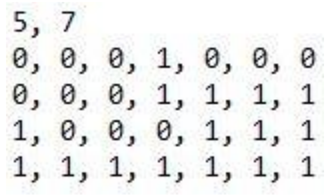
1. Test system behavior when contents conform to the specifications in the User Manual
2. Test system behavior when contents contain nonnumeric values
3. Test system behavior when the rows are not consistent
4. Test system behavior when the columns are not consistent
5. Test that integers other than 1 are treated as dead cells, as outlined in the User Manual
6. Test system behavior when the contents are not delimited properly
7. Test system behavior when non-integers are given in the contents

Below is example file input for some of the above test cases:



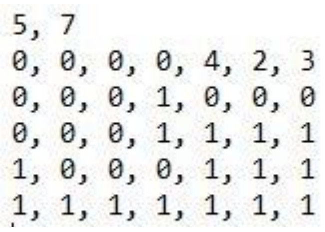
```
5, 7
A, B, C, 0, 0, 0, 0
0, 0, 0, 1, 0, 0, 0
0, 0, 0, 1, 1, 1, 1
1, 0, 0, 0, 1, 1, 1
1, 1, 1, 1, 1, 1, 1
```

Test Case #2



```
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
```

Test Case #3



```
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
```

Test Case #5