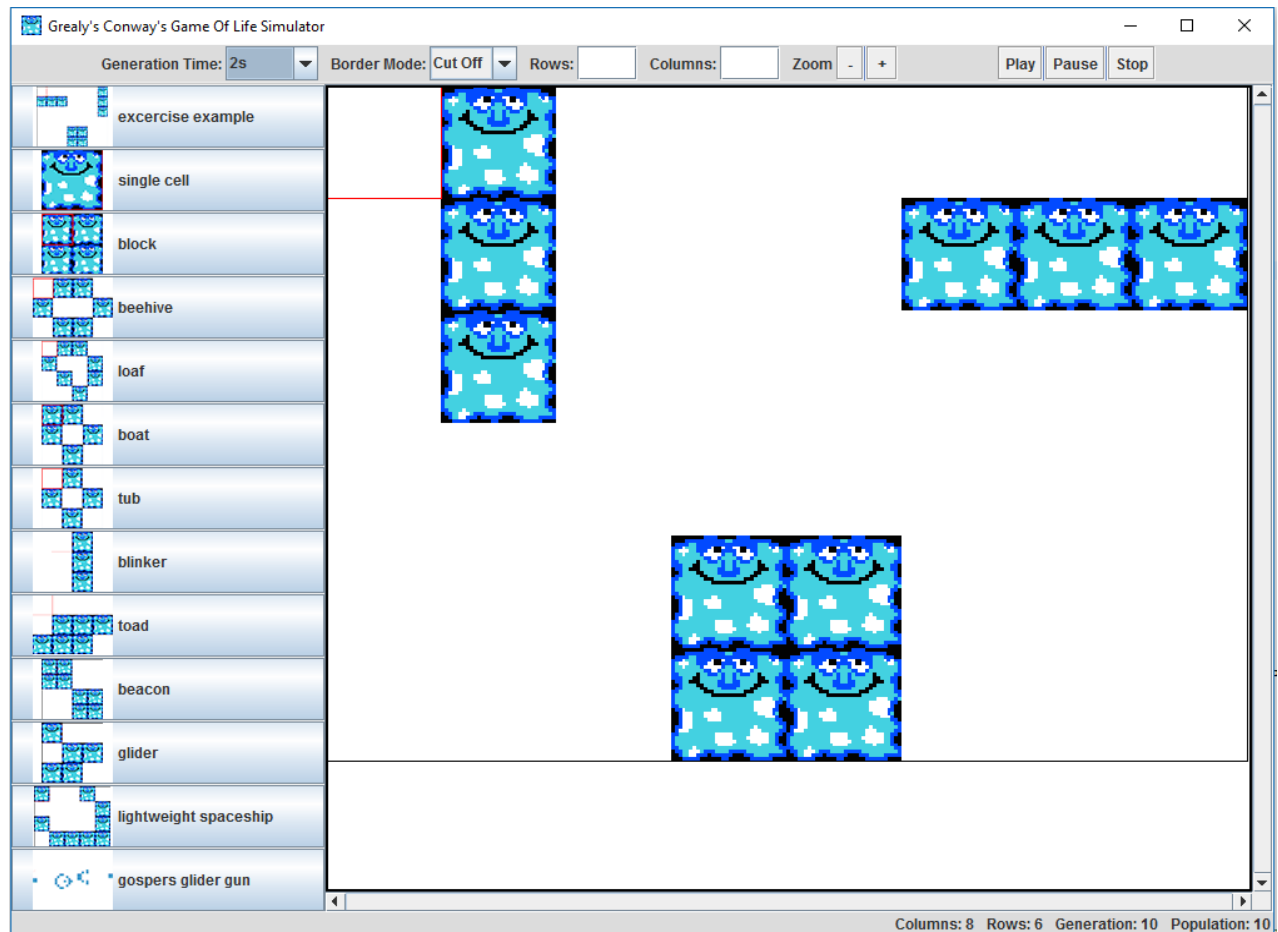


Grealy's Conway's Game Of Life Simulator

Jacob Grealy

02/13/2017

Version 1.0

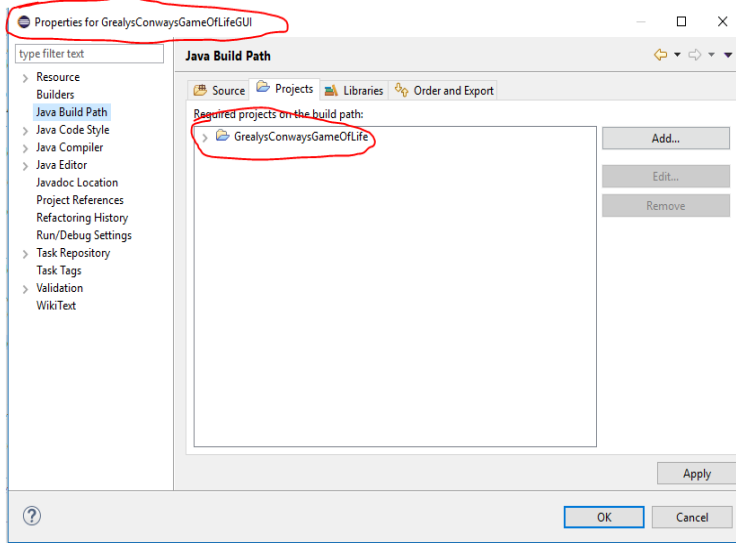


1.) Build:

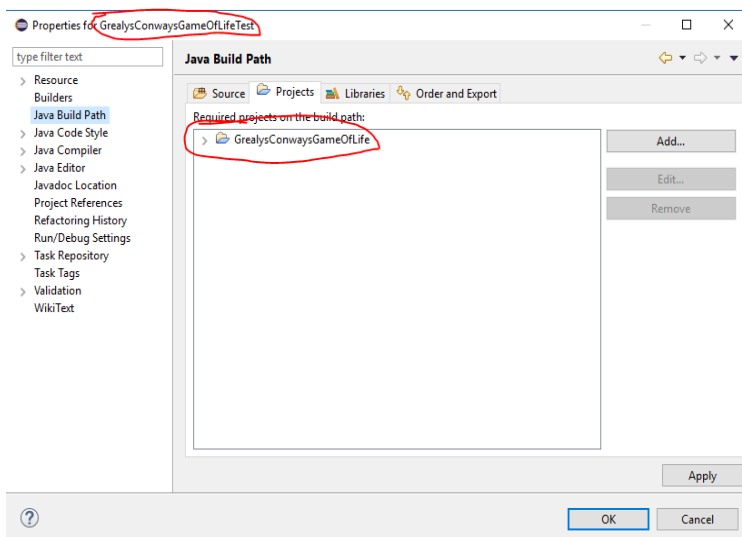
This was developed using eclipse Version: Neon.2 Release (4.6.2), and java version: jre 1.8.0_121.

I can not confirm whether it can be built using lower versions of java as I have not tried. I have not built outside of eclipse but I'm sure you could do that, just keep in mind the build requirements below. The easiest thing to do though is to just open a new workspace in eclipse and import all 3 projects. They should already be set up with the correct build paths that way and you can run the two runnable classes by right clicking them and then clicking run as Java Application.

A.) “/GrealsConwaysGameOfLifeGUI/src/GrealsConwaysGameOfLifeGUIApplicationWindow.java” contains the “**public static void** main(String[] args)” method for the runnable gui application. In order to build this you must first build the “GrealsConwaysGameOfLife” project. In Eclipse this is accomplished by adding it to the java build path for the gui project as shown below:



B.) “/GrealsConwaysGameOfLifeTest/src/AllTestRunner.java” contains the “**public static void** main(String[] args)” method for the Junit Test Runner. In order to build this you must first build the “GrealsConwaysGameOfLife” project. In Eclipse this is accomplished by adding it to the java build path for the gui project as shown below:

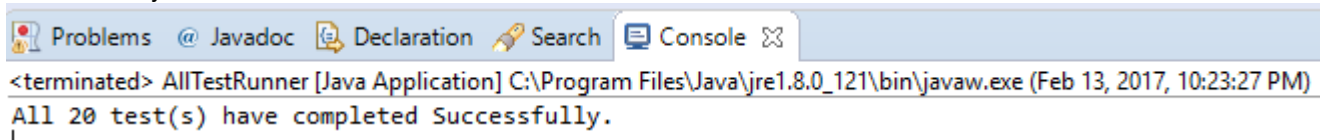


Essentially for both the test and gui projects, the main runnable classes are dependent on every file in their own project as well as every file in the GrealsConwaysGameOfLife project. These are the only dependencies; no external libraries were used. Neither class takes any parameters either.

2.) Test

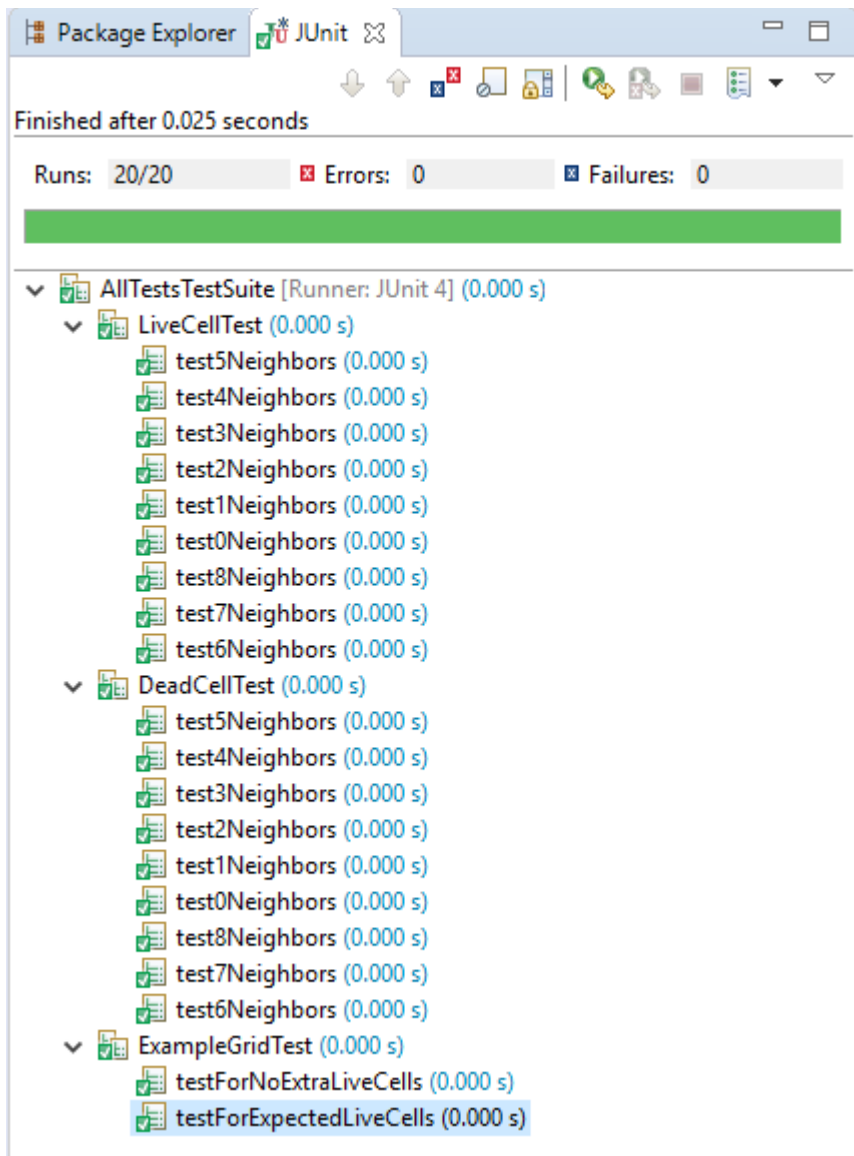
I have included a runnable class “/GrealysConwaysGameOfLifeTest/src/AllTestRunner.java”. This can be run like any other runnable java class. When running this class your view into the testing results is somewhat limited; because of this I prefer to run “/GrealysConwaysGameOfLifeTest/src/AllTestsTestSuite.java” as a JUnit Test in Eclipse. This gives you a nice JUnit view that allows you to dig into any errors as well as clearly see all the tests that were run.

AllTestRunner.java:



```
<terminated> AllTestRunner [Java Application] C:\Program Files\Java\jre1.8.0_121\bin\javaw.exe (Feb 13, 2017, 10:23:27 PM)
All 20 test(s) have completed Successfully.
```

JUnit Test:



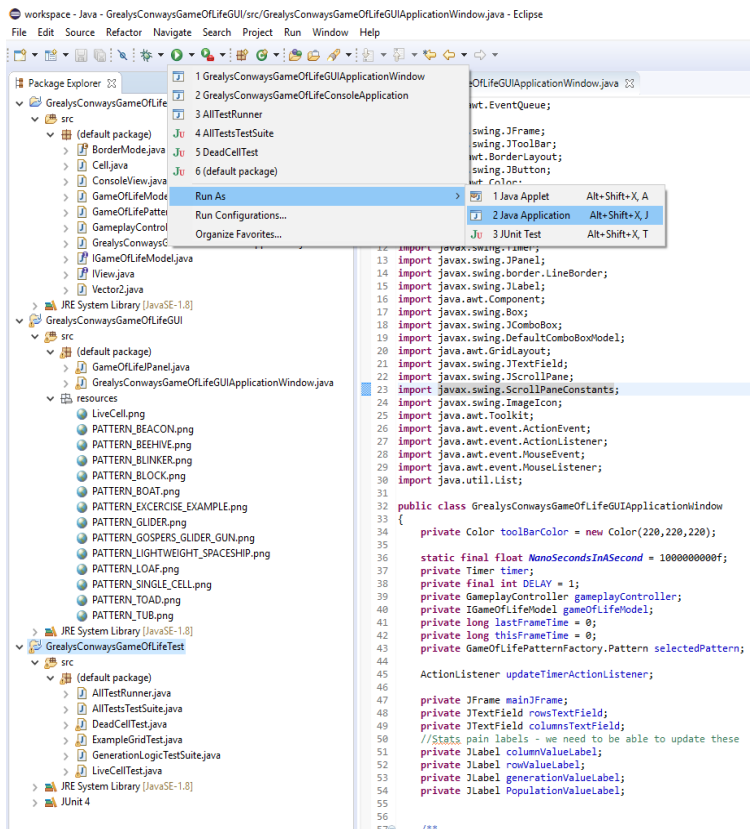
The tests have been written to confirm that the basic rules of ConwaysGameOfLife hold true after 1 generation. Specifically we test a live cell with 0-8 neighbors(LiveCellTest) and a dead cell with 0-8 neighbors(DeadCellTest). For good measure we also ensure that the example given in the exercise write up matches the expected output after 1 generation(ExampleGridTest) .

3.) Run

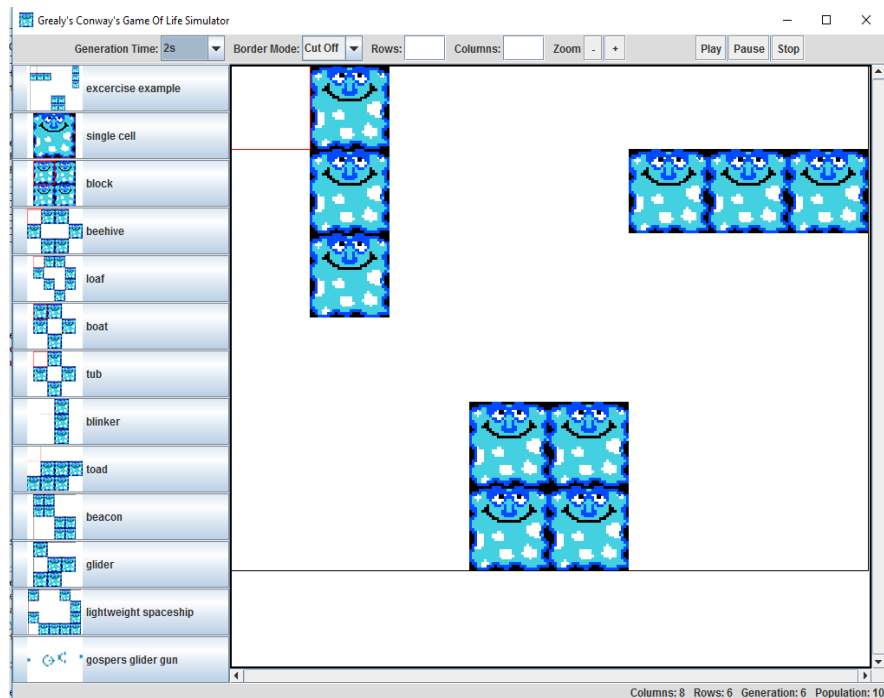
The main class for the gui application is

“/GrealySConwaysGameOfLifeGUI/src/GrealySConwaysGameOfLifeGUIApplicationWindow.java”. Again this class is dependent on everything in “/GrealySConwaysGameOfLifeGUI/src/” and “/GrealySConwaysGameOfLife/src”.

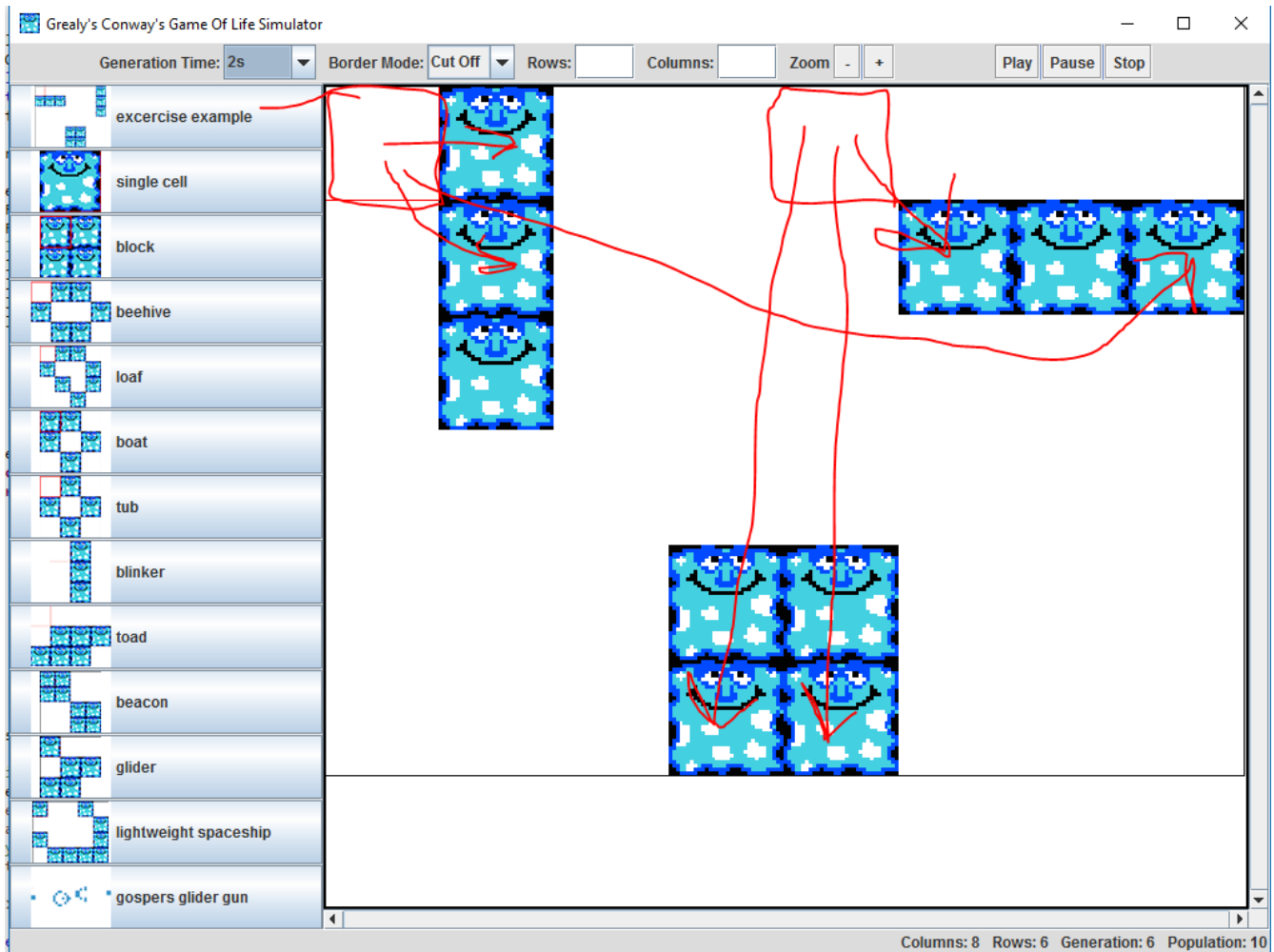
You should be able to run it as a java application, it doesn't take any parameters.



The first thing you should see looks like this:

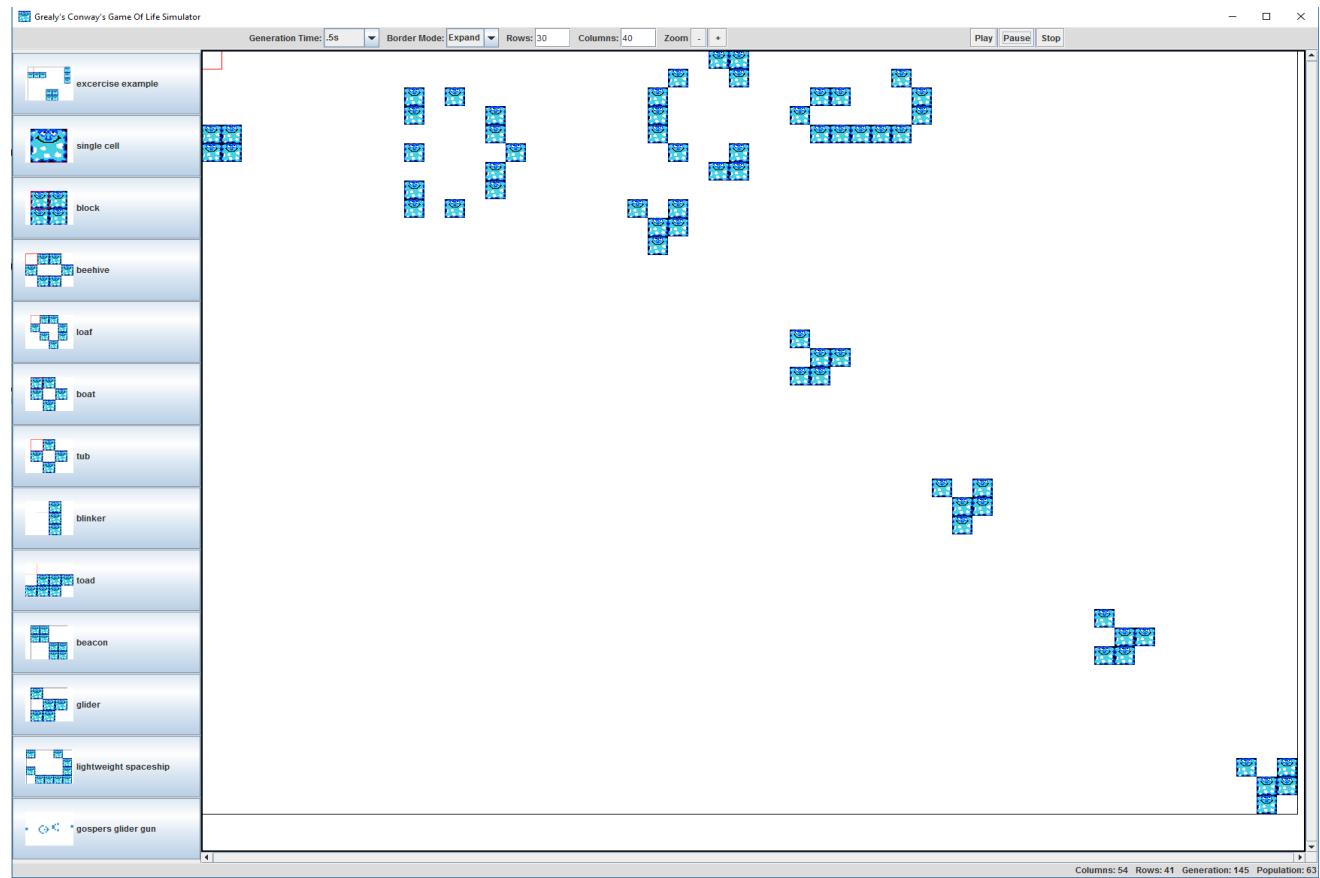


When the application starts it will default to an 8x6 grid with the exercise example loaded, running at 1 generation per 2 seconds, border mode of "Cut Off" and default zoom of 1, window size 1024x768. The window will not go smaller than 1024x768, but you can make it larger if you like. It will continue to switch between it's 2 states forever while the generation count continues to grow until you change something. The first thing I would recommend trying is changing the border mode from cut off to Wrap. This will cause the exercise cells to destabilize as a lot of dead cells now have 3 neighbors as shown below due to the outer blocks wrapping around. Eventually it will stabilize to a vertical beehive in one of the two bottom corners depending on which state you changed the border mode.



You can press the stop button to reset the state of the grid, this will clear everything, reset all the initial properties(though it won't recreate the exercise cells), and put the gameState in a paused state. Now you can click in the game window to "draw" cells one at a time, or you can select patterns on the left tool bar to add them with a click as well. I recommend trying the Gosper's Glider Gun pattern; it was the first pattern to be found that grows it's population infinitely by constantly creating gliders that fly towards the bottom right corner of the grid forever. You will have to either increase the grid size by typing in new row and column numbers and hitting enter or change the border mode to expand for it to fit though. Once you are satisfied with your initial state, you can click the play button to start generating the next generations. See below for image of Gosper's Glider Gun. The next section will go over all the features one by one.

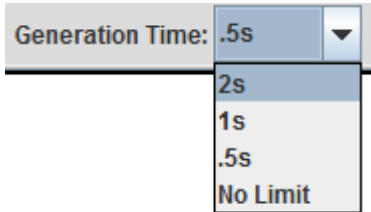
Gosper's Glider Gun:



4.) Additional Features

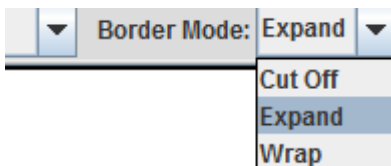
A.) Generation time: the time the gameplay controller will wait before generating the next generation.

- generation time can be changed to 2s, 1s, .5s, or No Limit.
- No Limit will result in the generations being generated as fast as the cpu can handle (or close to it in most cases).



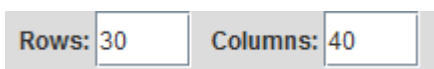
B.) Border Mode: determines how cells on the edge of the play area are treated.

- three border modes are supported:
 - Cut Off: the default mode, cells outside of the play area are considered dead and can not become live cells regardless of neighbor count.
 - Expand: The border will always expand to contain all cells. Dead cells on the outside of the border will be considered for becoming live cells, and if they do the border will expand to contain them as well.
 - Wrap: Live Cells outside of the border will be destroyed at the start of generation. However, when looking for neighbors cells on the border will consider cells on the opposite border to be neighbors, this counts for both live cells and dead cells. This results in “Pacman” like behavior for moving cells such as gliders and spaceships.



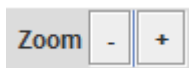
C.) Rows and Columns: the number of rows and columns can be altered by entering a number into the text fields shown below and pressing “enter” key. (nothing will happen if you don't press the enter key, consider yourself warned!).

- If the mode is set to Cut Off or Wrap, and the rows and columns are decreased in such a way that some live cells are now outside of the border, they will be destroyed. If the mode is expand, the border will immediately expand back out to include the cells.



D.) Zoom: controls how large the cells are drawn on the game panel.

- Can be controlled with the “-” and “+” buttons shown below.



E.) Play / Pause / Stop: The Gameplay controller can be controlled in the following ways:

- Play: Sets pause to untrue, making it so the gameplay controller will generate a new state after the specified generation time.
- Pause: Sets pause to true, while paused the gameplay controller will not generate new states, and the current state can be modified before resuming play.
- Stop: Sets pause to true and resets the gameOfLifeModel to initial settings: rows, columns, generation number, population, etc.



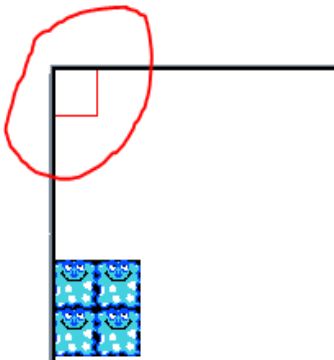
F.) Stats Display: Displays stats on the current generation.

- Columns: the current number of columns, ie the width of the play area.
- Rows: the current number of rows, ie the height of the play area.
- Generation: the generation number since the last time the model was stopped or initialized. (pause does not reset generation number)
- Population: The current number of live cells.

Columns: 54 Rows: 41 Generation: 145 Population: 99

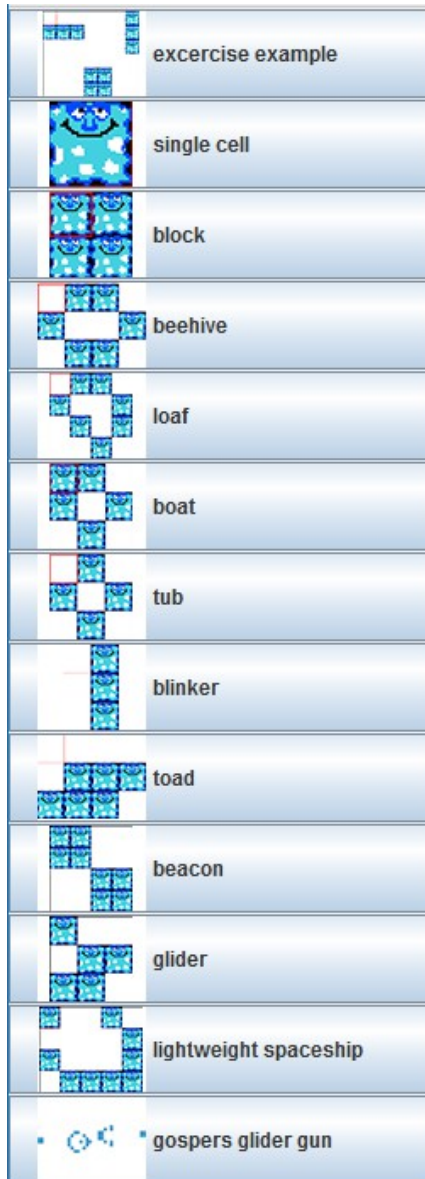
G.) Origin Indicator: red square drawn around the origin cell at position(0,0).

- The origin indicator can be useful when using the Expand border mode, as cell's can grow in negative coordinates which shifts the whole coordinate system.
- The origin indicator will always be drawn on position(0,0).



H.) Pattern Panel: Various patterns can be selected from a single cell to a Gosper's Glider Gun, and then be added to the game area.

- First click the pattern you would like to place into the grid, then click the top left point where you would like to place the pattern.
- Cells will not be placed over top of existing cells.
- The border will not expand to contain cells not placed with in the border unless the border mode is set to expand.
- Patterns obtained from https://en.wikipedia.org/wiki/Conway's_Game_of_Life



For inquiries on this software/documentation please reach out to the creator Jacob Grealy by email at JacobGrealy@gmail.com

For more by this creator, visit <http://www.JacobGrealy.com>