Megan Fanning
CS162

Conway's Game of Life

Design

The grid will be displayed on a two nested vectors. There will be a current board and a future state board, the births and deaths will be recorded in the future board. Once all the births and deaths have been tallied then the future board and current board pointers will be swapped and the curent board will be printed to the screen by the printBoard function which will check the cell's boolian value and if true print a “8” to the board (the boolian conversion to char will be contained in private static constant of the board class.

The cell life cycle will be managed by a function which checkes all adjacent cells to determine the status of the cells.

I considered a few options to allow a infinite grid, stiching the top and the bottom of the grid together, trying to identify when a group of cells had left the 'live' grid etc. However I found the simplest solution was to kill cells once they were 5 past the visible grid.

Features List

Functions for board
-print board
-board array

--Lifecylce
--living array
--future state array
--clear array function
----
menu
-welcome screen
-user seed function accepts spawn point and number of rounds
-stop button

Lessons learned

One of my biggest debugging issues came from using rows and columns as my variables instead of x and y initially which resulted in me swapping the coordinate and getting bad data in my print outs.

One of the best debugging methods I found was to print the number of neighbors each cell had instead of just printing a '8'. This allowed let me check for errors in my algorithm for checking the number of neighbors.

I ended up scrapping the stop button feature because I felt making the user hit enter each lifecycle of the board was clunky and instead I allowed the user the choice of how many life cycles to display. I chose to limit how the user could give imput on shifting the design to ensure they would not accidentally shift something outside of the visible grid.

I found I didn't need a clear array function because I could swap the future and current arrays and directly overwrite the array without clearing it.

When asking the user how many spaces they wanted to shift the design if they replay with something other than a number it will create a infinite loop, however if they reply with a incorrect number it will properly cycle.

I was hoping to do more input validation however since it wasn't included the requirements I opted to impliment it last so a number of my functions would break if the user put in bad input.