Phase II - GOL

Overview:
Since this phase was a continuation to the last phase, most of the design decisions and challenges came through in the last phase of the project. However, I add some changes to my last code in order to make it more robust for this phase of the project. The biggest change was to allow for DOM manipulation to display the output as opposed to the canvas. Moreover, it came different from the last phase in that now users could input their own initial conditions, grid size etc which in contrast were hard coded earlier. The main 'game' concept remained the same but the way the system worked changed. More on this is below.

Main focus and goal:

The main focus of this project was to incorporate the DOM to system. So now, I allowed users to input data including i.e. initial conditions and grid size graphically; this data was then translated into system code representation, the game code then worked on this input, and based on its function, generated an output. DOM simplified the input, translation, as well as outputting the data. Now the users could generate their own patters, stable or unstable ones. This could give them the extra motivation to try and make stable patterns out of their initial conditions.

System description - implementation :

My last code wasn't comparatively as robust. This time, I decoupled a lot of components. There's less game code in graphics and hence less interdependency in the code. Moreover, there's much less exposure. Improving the robustness in itself was a design challenge as I had to figure out the most optimal way to design my code such that there was no exposure or very little exposure. There's more decoupling of code and hence less interdependency.

F**eatures of minimum viable product and UI:**
Once again, I tried to keep the display simple. Main features include a
1) board that allows users to set initial alive state
2) run the state to see how it evolves
3) stop the simulation
4) increase or decrease the speed of simulation (also extra feature)

The UI is simple with not too many coloring and styling. However, it serves the purpose well.

Testing:
Unit tests suites to be supplied as array of :
arrays of state_change_fns with requisite args(as array) and
and
arrays of test_fns with requisite args(as array) and expected_result.
run_unit_tests applies all state_change_fns to game and then goes through all test_condns, and checks for expected results for each one of them.S
Please see sample tests in tests.js.

Extra Feature:
Evolution is a very erratic process. And it's hard to determine how gradual it is. So I thought why not allow the users to decide this for themselves? My extra feature allows users to control how fast or slow their species evolve!

Conclusion:
My code feels a lot more robust and well designed at this stage, esp due to the DOM manipulation as well as decoupling of code components. The approach is simple and the names are self-judicious in that it's easy to determine what each small snippet achieves. Overall, this was a decently challenging problem set but I hope to have attempted it wisely and in the scope of good software practice.
Thanks.

Isra Shabir