6.170 Project 1 Phase 2 Design Document
Michael Wee – September 15, 2013

There were several problems to be solved in this phase. Because we were working exclusively with the DOM, there are several problems to be solved there. The first is that we need to be able to initialize conditions in the DOM by clicking on the cells. There are two main ways of solving this: one is creating HTML elements for each Game of Life cell and the other is tracking the mouse position on the page and calculating from that which square to trigger. I chose to create divs with IDs where we have an event listener that triggers a clicked function that activates the logic for the given cell.

Another problem to solve is to be able to start and stop the game and edit the board state in between. To be able to start and stop the game, I switched from the setInterval function to a recursive call of the setTimeout function. Stopping the game I added an extra feature that allows the user to change the speed of the gameplay of the board. The switch to setTimeout was crucial to allow me to change the milliseconds parameter from within the loop as the user changes the time while the simulation plays. I decided to allow the user to make changes to the board while it is running to allow for a broader range of experiments, and for the user to exploit the ability to slow the game down.

The final problem to solve was the coupling between the game logic and the DOM, and the requirement that states propagate between the two. Because of my design to decouple the graphics engine from the logic in phase 1, I was able to keep the game of life logic almost completely intact. The only change I needed to make was to add a set_board function that would take the current board state as seen from the DOM and update the internal state. The dom_graphics file contains all of the DOM manipulation. It creates the DOM elements, and updates and modifies them as needed by the game logic with the help of jQuery functions.

In terms of testing, I set up other initial configurations that are standard Game of Life patterns to test the logic of the board. These also verify the proper positioning of cells on the board. I included one with still lifes such as the block, beehive, and loaf. I also included the oscillating beacon and glider. These tests can be activated by selecting the corresponding initialize board function in the Board class. These tests are the most important as they are a way of verifying behaviour in the game logic that we know and can clearly define. These tests also verify very important parts of the game: that cells are correctly initialized and that the mapping between the board array and the DOM board are in sync. For this relatively simple system, it makes sure the few moving parts work as desired and are in sync: the game logic, the graphics engine, and the interface between the game representation and the graphics engine. The other necessary inspections are all observed graphically such as the proper location of the grid lines, the DOM, and the visual style. I also conducted visual tests by initializing the board with different times and patterns.