Project 4- Integration Strategy

The Integration strategy we used was Bottom-Up Integration. We had an initial design that included the following classes:

* Game
* Battle
* Force
* Unit
* GameMap

The “Game” class acts as a controller, using the rest of the classes to manage the game. It first initializes a list of Force objects, which themselves consist of a “side” and several Unit objects, to represent the game map. Then, it allows the player to act on Forces by clicking them and then clicking adjacent regions. When two forces collide, the Game class creates a “Battle” using the opposing forces, and the Battle object applies damage (based on the units in each Force) at a fixed interval until one side has been reduced to 0. Attackers always have a chance of recouping some of their losses (for each force in the battle, a fraction of each unit in the force is restored based on RNG), but defenders only recoup losses if an adjacent region is friendly. The GameMap class is a static collection of methods, and controls all modifications of the SVG on the webpage (helping to isolate problems with logic from problems with graphical elements).

By programming the Unit class first, then Force, then Battle, and then Game, our development path was predictable and easy to follow. Bugs are easy to isolate to a specific class, and refactoring has been relatively easy (for example, moving from allowing 1 move per turn to 3, and making battles play out all-at-once).

While the bottom-up approach usually means that the lower-level modules got a lot more attention, we were able to thoroughly test the Battle class and have had few issues with the implementation of the Game class. We also found ourselves able to test partial code early on in both Project 3 and Project 4.