Project Write-up and Design Choices

Jason Richter and Sam Fiscus

The basic idea of the project is to have a “battle simulator” which could be used inside of a larger program as a module. The code features a turn based combat similar to other games such as Pokémon, in which turns are ordered by an initiative stat. The turns are organized and run by a class titled encounter, which also contains all necessary data for the combat encounter. Each encounter object is a self-contained combat encounter fully capable of coordinating its actions without a need for a client besides initialization. The encounter uses a binary search tree sorted by initiative to order the turn, then reloads and repeats using recursion. The encounter contains entities which will fight each other or the player using rng. The player object is a type of entity which is controlled by input as opposed to rng. The entities all contain a StatBlock object, which contains all the necessary stats for each entity such as health, initiative, or attack. They also contain Equipable objects of type armor and weapon which enhance their combat prowess. Each entity also contains Usable and SpecAttack objects which they can use a limited number of times in combat to gain an advantage. At this point in time only Players have the ability to use these, though all should be able to soon with minor changes.

Through the development, the hardest portion to conquer was the Encounter class controlling everything. Choosing the right data structures and method of looping through turns was crucial to get the correct result. Originally it was slated to use a PriorityQueue to order each turn. But the binarytreemap ended up being easier to work with and automatically sorted itself too. Sam figured out that using a recursive structure to actually run the combat made more sense than a more traditional loop, as the only way combat ended was in a base case, being that one side has to win or everyone is dead. Since each turn runs identically structure-wise, it can be recursively called.