Samuel Fiscus and Jason Richter

CS 341 – Data Structures

Final Project Proposal, revised

We’ve totally revamped our project proposal idea. Instead of a fully-fledged, text-based adventure game going through a pre-set adventure path, we have chosen to focus our project on a previous “A++” portion of the original game – an object-oriented combat system. This topic would better incorporate the themes of our Data Structures course, such as other-focused design and documentation, as well as replicability and compatibility between clients. Our documentation of our interfaces will be designed in such a way so that another team would be able to design their own “encounters” and “entities” that could be swapped into our client, and vice-versa.

Whatever the implementation, ­­­­­entities will be able to combat each other with a model similar to games such as Pokémon or the Final Fantasy Series. Each entity will have the basic choices of attacking, defending, or using special abilities. All implementation will be based off of formulaic output computed from entity attributes. For example, an attack method may output an integer based off of the home entity’s attributes, and this output will be passed to a defense method of another entity, through which the final damage will be computed to be subtracted from the other entity’s health attributes. Another attribute that may be further developed will be an integer representing the “team” that an entity is on. This could be used in order to prevent friendly fire or to focus enemy attacks if we decide to enforce a simple AI.

The main class describing the basic combat unit, “Encounter,” so far will aggregate two entity lists, one each for current and removed combatants. “Encounter” will also have a priority queue representing the current entities’ turn order, calculated from relevant attributes, that will be refreshed after every turn cycle. As combatants are whittled down, they will be removed from the current list and be transferred into the defeated list, to make combat easier and also to allow for a victory output and the option to pick up their swag. These may be future implementations, so we’re building the framework to more easily allow for this later.

What we just described is more of our C-grade interpretation. For our B grade, we want to incorporate a list of aggregated usable items that can execute a method on a target entity(ies). The implementation of items’ descriptions and methods will be up to the designer of the classes, but the framework will be built in order to develop item objects that can utilize single or multi-targeting methods to help or hinder entities. An example may be an entity having potion item that can target a single entity; this potion may have a method that calls a setter for the target entity’s health in order to add a set amount not greater than the target’s maximum health. Another example may be a grenade that can target multiple targets on a team, effectively placing an attack on them all.

Another implementation at this level would be equipable items that would have equip and unequip methods, calling setters on entity attributes to modify and restore original values, respectively. A prospective way of gaining these equipables may be from looting from removed entities during or after a combat encounter. Depending on our motivation, we may choose to incorporate a “slotted” equipment structure so that an entity may only have one helmet, for example. Otherwise, these equipable items may be in a list or only available as a single equip.

Our A-grade interpretation of this project will incorporate a status effect aggregate of the entity class. These status effects would call place and lift methods that would use setters to alter/restore target entity attributes. The status would have to utilize a turn counter to be lifted off of an entity, or perhaps have a connection to a certain item that could do so manually.