Game Content Design Documentation

Milestone 1 Design:

Characters

All Character classes for Milestone 1 were relatively untouched. The milestone 1 project outline defined a hierarchical abstract class structure so many of the functions could be split be inherited by their super classes. The reason for this was to make the creation of new characters incredibly simple. This also allowed definition for static values every instantiable class would need (initial stat values, leveling functions, etc).

Game Environment

Environment's design was to contain a set of tiles that could be built up into a game world by putting them in an array of some kind. The idea was to make it relatively low level, with each tile containing the information it would ever need. These tiles were then assembled into a two dimensional array to represent the world.

Items

Items were meant to have the same sort of hierarchical structure as the characters that way we could treat everything as one class and only the two exceptions if the swag was an instanceof weapon or armor. Inventory was meant to be self-sustainable, providing all the necessary methods to manage the inventory's contents and money.

Quest

A quest was meant to be a list of QuestEvents that would be polled every time a player performed an action. Since the events had to be done in order, I speculated we would use a Queue to manage it and just poll the top quest event whenever it matched up with the action performed.

Actual Design:

Characters

Characters kept their design with very few differences in design. Because of the server/client interaction and each responsible for maintaining their own states, all changes to any character had to be done through commands. That means some of the functionality I had planned on for the characters had to be removed and moved into a ComparableCommand to ensure game state synchronization.

Spell Inventories were changed to contain AbstractSpells that were capable of executing on the game state. The GUI's code was expecting each spell to be able to create itself without knowing anything about the state. Therefore the DummySpell setup (which just kept track of which spells the player could cast) had to be removed and replaced with a way to make it easier to instantiate the proper command. This forced the creation of a single CastCommand to be

instantiated with the AbstractSpell passed in as a parameter so it could access the correct information. While it generalized the CastCommand to the GUI, it made the CastCommand itself a little grungy since many of its validity tests depended on the semantics of the spell. But, provided you check for the specific requirements, the way it works now make it easy to cast a spell.

Despite the problems with casting using the GUI's "cast to move" mechanics, all spells do have a correct range as required by the specification.

One other major change to Characters came from the animations side of things. Ian implemented all of the changes associated with the graphics. But this is related to changes sent across the server, to make sure other clients view the character doing the correct animation.

Game Environment

The game environment was static after Milestone 2 when tile overlays were added. The only change to the way the game environment was interacted with was how the game map was serialized across the network. We did a tricky thing where we overrode the readObject and writeObject methods by adding methods with the correct method signature. Since Ian needed to add a TileRecord between two tiles in the world to calculate Djikstra's algorithm and that TileRecord contained a HashSet, we had to find a way around serializing the HashSet. Java doesn't serialize HashSets properly so when we overrode the read/write Object methods, we were able to send a string that could be reconstructed on the client's side.

Items

Items kept all design from Milestone 1

The only additional change came from when the game was initialized. Each time a server is created, it runs a game initializer which creates a finite amount of monsters that the game re-uses throughout the duration of the game. This meant we had to have some way of creating new loot whenever they died so they didn't drop one piece of swag the entire game. This problem lead to the reflective SwagFactory, which used reflection to create new instances of the items upon the death of a monster.

Quests

Quests kept all of the original design from Milestone 1, EXCEPT that it no longer used a Queue of Quest Events. Ian wanted to be able to draw all of the quest's events so he requested the change to an ArrayList. Since it wasn't a difficult change this was implemented and correct accessor methods were implemented as well.