Design Document for: CS401
Authors: Tristan Brodrick, Chongyuan Chen, Ryan Johnson, Jordan White
Date: 02.18.2020
Status: Project Underway

Concept Summary: The app we are proposing is a dungeon crawler type game with JRPG style combat. The basic idea is that you have a party of four allies that work their way up a tower moving from one floor to the next. On each floor you could encounter a battle, a shop, a rest, or an event. In battles the computer determines a turn order randomly, then your allies take turns with the enemy making one action per turn. These actions include attacking an enemy, casting a spell, or using an item. Once all enemies are defeated, you may move on to the next floor, or if your allies are all defeated, then the game is over. When defeated, enemies will drop loot and gold, loot includes weapons and items to be used in battle, while gold may be spent at shops. Shops contain items that can be bought by the characters in exchange for gold. At a rest site your party may rest and heal themselves for the battles to come, although each may only be used once. Lastly you may encounter an event. Each event is different, but generally there will be some text explaining the situation, and a list of options as to how the player can react. Depending on how the player reacts to each event, there will be different benefits or consequences.

Audience/Customer: Our audience for this game would be the mobile gaming market. This is comprised mostly of teens and young adults. These games tend to use a more modular format, and we believe that our floor based system will be perfect for that, as you can easily beat one or two floors before putting the game down and coming back to it later.

Background: Our game is done in the style of a dungeon crawler, meaning that you move from one floor to the next going as far as you can to get the high score. The combat is more in the JRPG style, meaning it is 4 allies versus up to 4 enemies, each taking turns taking actions until one side is defeated.

Application Cost & Projected success: The application’s cost is planned to be $1, with there being monetization in periodic ads played every 10 floors of progress.

Design- Detailed Design /Dependencies: For our design we are going to make each run it’s own object, storing: an array of allies, the floor number, what's in the previous floor (so that the player may return to it). The game object will be able to save and load games by accessing cloud storage, and will also determine which floor will come next.

Each floor will be an object type of it’s own. Battle floors will determine turn orders, generate and track enemies, and manage attacking and item usage. Shop floors will generate an array of items for the players to buy, track whether a character can buy an item based on the character’s class. Rest floors will only heal the party, and thus will be able to access character’s stats. Event floors have a variety of possibilities, but will be self contained encounters that give or take items from the players, or modify the stats of the ally characters. The characters in the game both allies and enemies derive from the generic “character” class. The parent class stores gold and health. There’s also an array of items that serve as the character’s backpack, these items can be used if the character is an ally and will store the loot dropped by enemies. There is also an experience value tied to each character. This will be the amount needed to level up in allies, and the amount given to allies by enemies. The ally subclass introduces an array of stats including: strength (increases weapon damage), intelligence (increases spell damage and gives more mana), and constitution (increases health). A string representing the ally’s class, A level up function that increases the character’s stats based on its class, and a set of actions the character cna perform in combat. Enemies are much simpler only introducing am attack and defence stat. The final class is the Item class, this is used by the shop and by characters. The basic Item class has strings for the name and description of items, and int values for cost and rarity. There are three subclasses of items: weapons have an associated class and an attack value, armor has an associated class and defence value, and then other generic items have individual functions that make them function (e.g. healing potions increasing your health).

Related Work: The game was originally influenced by the game “Slay the Spire '', which has a similar structure to their floors, but without items. There are many other similar games currently available on the app store. One particularly interesting game that we found on the app store is called “Soda Dungeon”, which also involves very similar mechanics introduced in our game.

Frameworks/Services/Cloud/Backends: We aren’t planning on incorporating many Frameworks or backends. We only plan on using the cloud to backup saved games, so that the user can quit and come back to a run later.

Testing: Testing will be performed in a modular style as well as a whole; each aspect of the game will be tested individually, incorporated into the application, then tested as a part of the entire product.

Schedule: We plan to have the character and floor classes built by the end of February, with the children of the classes completed in the first week of March. By next week, both class testing and the item class should be finished. Next week, the user gui should be completed, but without functionality, which will be added in the coming weeks. All classes, except for main, should be completed and properly tested by the last week of March and the application should have some user functionality. In the first week of April, the application should be able to have cloud storage capabilities. In the second week, user gui functionality should be completed, leaving the last week to do a full application test. This project will be completed by April 23.

Dependencies: In this project, both our main, and characters depend on our floor class. These dependencies are based on how character actions are partially determined by the floor type and the main class requires all other classes to be properly functioning. In addition, all of the floor class' children, shop, battle, and rest, are also dependent on the floor class. Likewise, all of the character class' children, that being enemy, and player, are dependent on the character class.