Specifications and Design Document

Basic RPG Systems

Cristian Ponce

Software Engineering Methods and Projects 1

Objective

```
Design and implement an application of your choice. The a
pplication needs to be
fairly complex – in terms of the functionality, use cases
, number of modules/parts.
Please note that your design should include at least 1 de
sign pattern from each of
the categories Creational, Behavioral and Structural patt
erns.
```

Proposal

```
Design a set of systems commonly found in RPG (Role Playi
ng Games) type games.
```

Those commonly found systems consist of:

`modules`

1. Stat System
2. Inventory System
3. Character Creation System
4. Item Creation
5. Item Database
6. Weapon System
7. Consumable Item System
8. Upgrade System [for both characters and a items]
9. The list goes on...

But mainly I'm going to try and focus in on the `Inventory System` using `Creational` and `Structural` design patterns for the Items. After that I'll move onto adding `stats` to all `Items` through the use of a robust `Stat System`. Following with the Creation of the `Character Creation System` to then add stats to the player himself that change depending on the players level. Depending on the player's stats determines how much the player can actually carry in his inventory. Which then brings in the upgrade system that allows the player to upgrade his level and increase the amount of Items (weight) that he can possibly carry.

Functionalities

The Character(Player) can carry an inventory of a certain weight depending on the character's level stat

The Inventory holds a list of items. It handles the addition and removal of items with respect to total and current weight. The Inventory also deals with handeling the player’s currently equipped items. Depending on the game, the player is allowed anywhere from 1 to 7 slots of equippable items.

Items are split into three different categories - each having there own specific objectives.

1. Weapons - Objects that the character uses to defend or attack other enemies. [Ex: Sword]
2. Consumables - Objects that the character uses to get special abilities or restore stat points [Ex: Potion]
3. Quest Objects - Objects that the character holds to boosts stats or use for a puzzle in the game. [Ex: A Key]

Use cases

The stat system is usable in really any game. Any application that requires a set of numbers correlation with a single entity.

The inventory system is pretty much like a database of items in a sense, except the items don’t have all there default stats and numbers, they are changed by the player with use over time. The item database holds all types of items for use (copy) into the inventory with all there default levels and numbers.

The character creation system allows you to create players of different types that allow them to use different abilities or start off with different base stats.

There are many use cases for all of these systems, they all build up to work together and form the commonly used rpg systems.

Design patterns

The design patterns I’m planning to use could change by the time I start implementing the patterns into actual code to see what works and what doesn’t.

1. Charcter System - Private Class Data Design Pattern [https://sourcemaking.com/design_patterns/private_class_data]
2. Weapons - Decorator Design Pattern
3. Consumables - Decorator Design Pattern
4. Quest Objects - Decorator Design Pattern
5. Inventory - Composite Design Pattern
6. Item Database - Singelton Design Pattern
7. Stat System - Decorator Design Pattern

Rational and Trade Offs

Character System can be made using the private class data design pattern so we can split off all of the player’s specific variables and set them in a seprate class.

Item classes [Weapons, Consumables, Quest Objects] can be made using the Decorator Pattern becasue they can all be upgraded and as they’re upgraded they can change into different stronger objects or add different abilities by adding different accesories (“decorators”) onto them.

The Inventory has to hold objects that might have both simple and composite structures.

The Item Database has to be able to be accessible from anywhere and there can only be a single list to access. So it has to use the singleton design pattern.