Design Plan - Dungeon Dudes

Jeremy Carter, Elizabeth Creek, James Viner

Project Summary

Dungeon Dudes is a simple role-playing game that generates rooms for the player to explore, monsters for the player to fight, and treasure for the player to collect.

Features Targeted

Attack Potion

Add a usable piece of loot, the 'attack potion', that the player can drink in order to roll an extra attack die during their next attack. This can be done by separating a parent class of treasure into usable and unusable categories, in which all forms of treasure contain descriptions, but usable treasure also provides the option to 'use' the item. The new 'use' option will be accessible via the menu.

Running Away

Add the option to run away during combat. This is doable by creating a room guaranteed to have no monsters and placing the user inside of it, giving them an opportunity to look over their available options before continuing.

User-Provided Monster File

Add support for a file that allows the monsters seen in-game to be customized. Allowable fields are name, starting health, and number of attack dice. This feature will involve deciding on a format for the input file to use, reading in and stripping off the relevant data from that file, and primarily error-handling in the case that the user places irrelevant or unusable data inside. A list of program-default monsters will be retained in the main file and used in the case that the monsters file was not valid.

Architecture

Classes

This program will require the use of multiple custom classes for the instantiation and use of like objects. Below are their names and attributes.

```
class Entity():
  max_hp
  curr_hp
  num_dice
  make_attack(self)
class Monster(Entity):
  name
  description
  has_treasure
class Room(object):
  description
  num_foes
  has_treasure
  num_paths
class Treasure(object):
  name
  description
class Consumable(Treasure)
  use_item(self)
```

Significant Functions

`generate_room(num_enemies)`

This function will be responsible for the generation of a room object with randomized characteristics. A number of enemies can be passed as an argument in order to explicitly set the number of enemies selected to be fought in a given room.

`resolve_combat(player, room)`

This function will be responsible for resolving combat with a given enemy, including temporarily changing available menu options, updating health for both parties, and printing appropriate messages for damage dealt and a player's defeat or victory. Accepts the player object and a room object as parameters for use during combat.

`populate_monster_list()`

This function will be responsible for reading from the user-editable `~/.dd_monsters` file and replacing the game's default monsters with the ones the user provides. Returns True if the replacement was successful or False if there was an error. If there was an error, an appropriate error message will print and the program will continue, defaulting to the program's pre-defined monster pool.

User Interface

The user will interface with the program by passing menu options that correspond to game actions.

General Approach

1. Creation of Menu class
2. Creation of Room class
3. Random generation of rooms and ability to explore
4. Creation of Entity class and Monster/Player child classes
5. Implementation of combat (including flee feature) and fail state
6. Creation and implementation of Treasure class
7. Implementation of Consumable class (attack potion)
8. Implementation of user-editable monster file
9. Implementation of unit testing