Nick Weinert
Project 5 Program Design
CS 162–400 11/18/2018

**Program Design/Modularization**

This game will be a text-based escape game with 3 levels: a dungeon, house, and property. To get out you have to collect a bunch of metal shards in the house while you defeat common horror film enemies. Outside, there is one last boss (a fire breathing dragon) to fight. If you defeat him, his dying breath melts the metal shards and forges a key and then you escape.

To implement this game there will be 5 required classes. Dungeon, House, and Property that all inherit from Space will all be needed in order to implement the space and layout for the game. There will also need to be a character class to implement the players. The menu class should also be included.

The Spaces function will be the one that contains most of the variables. There will be variables in order to link the spaces together, as well as Boolean variables to track whether or not the space has been cleared (monster defeated). There will also be a name variable for the space and a name variable for the monster. There will be an attack function in order to fight the monster as soon as the character moves to the space. Dungeon, House, and Property will all override this attack function.

I do not plan to have the character contain much aside from a pointer to space to track where the character is, and a name.

**Test Plan:**

| Test Case | Input Values | Driver Functions | Expected Outcomes | Observed Outcomes |
|---|---|---|---|---|
| **Check that the pointers traverse through spaces properly** | **Set the up, down, left and right pointers.** | **setSpace(Space *)<br>getUp()<br>getLeft()<br>getRight()<br>getDown()** | **Current spaces should change according to which space is told to change to. For example, setSpace(getUp()) should change the current space to the space above the old space** | **All spaces move properly** |
| **Character dies** | **Fight a monster with no attack a bunch of times** | **doAttack()<br>setHealth()<br>getHealth()<br>getDead()** | **When player runs out of health they should be dead and the game should end.** | **Player dies and game ends appropriately** |
| **Character fights monster** | **In each space class, fight monster** | **doAttack()** | **Should loop, causing damage to monster health and player health until either player is dead or player is dead. Weapons should have** | **doAttack works as expected** |

| | | | | |
|---|---|---|---|---|
| | | | **random effectiveness** | |
| **Search space** | **3 tests: Take health potions, take metal, Keep searching** | **searchSpace( )** | **Take health should add to characters health, take metal should add 1 to satchel size, and keep searching should randomly swith between these options until user takes something or quits** | **All three tests work as described** |
| **Test print functions** | **Print the things that might need to be printed** | **printSatchel( ) printMap() printHealth() OptionsMen u()** | **printSatchel() should print size of satchel to indicate number of metal shards printMap should just print a static map printHealth should give characters remaining health OptionsMenu should loop until the player moves, allowing printHealth, printSatchel, printMap and searchSpace to be executed repeatedly. SearchSpace should not happen more than once, with a message displaying that space hs already been searched** | **Everything prints the appropriate statements, with the options menu having the requisite gurads against performing functions more than once** |
| **Entire game** | **Test the whole game a few times to make sure everything works together in unison** | **Entire game** | **Game should be tested trying to win, trying to die, trying to leave house early, etc** | **Possible to die. Possible to win. Possible to defeat final boss and lose, not possible to leave house early.** |

| | | | | |
|---|---|---|---|---|
| | | | | **All do attack functions work properly, things are labeled properly, entire game seems to work together well. No memory leaks or segfaults** |

**Reflection**

First and foremost, you'll notice that the character class contained significantly more information that just a space pointer and the name. I also put menu functions in order to choose what to do after each space has been cleared, an attack function, health functions and values and a container to carry metal shards to make a key. I had originally planned to make the game run primarily inside of spaces, but quickly changed my decision to have the character run most of the game.

I also made the decision to not let the character out of the house until every space had been searched. This was for two reasons. First, I wanted the player to be forced to fight every monster, and second I wanted to give the user the maximum number of chances to collect metal shards. There is no explicit clue that 4 shards are needed in order to win the game, so I wanted to give the most opportunity.

Other than these changes I did not need to make any major changes, nor did I run into any big problems with anything. This project went smoothly after I decided on a theme.

As a quick tutorial that may not be readily apparent at the start of the game:

- The name of the weapon doesn't matter and attack effectiveness is random. Attack effectiveness does not change unless you change weapons. If you missed and don't change, you will continue to miss. Effectiveness changes even if you select the same weapon.
- You only need 4 pieces of strange metal to melt into a key. Any more is a waste and you should take the health potions instead. 3 or fewer strange metal and you can defeat the Dragon and still lose the game.
- If you hide in the house, you have a 25% chance of the monster going away and your weapon effectiveness is the same as if you had selected a weapon (you should always hide, unless you get a great attack and then keep attacking until it's dead).
- The Dragon has 750 health. Change attacks until you get a "Great Attack" or it will almost surely kill you.