Plan of Attack CC3k+

Kabir Ashish Wahi

Thanh Le

Haichun Kan

Project Summary

The project will mainly revolve around game.cc and main.cc. The game class will render the map itself along with all the enemies, items, and the player. There will be a character abstract class which will be the superclass that all player characters and non-player characters will inherit. There will also be an items abstract class which the potions, gold, and barrier suit will inherit. To print and update the game board, the game class will have print and update functions.

Estimated Completion Dates

1. Creating UML – 15th July
2. Creating interfaces – 16th July
3. Implementation
   - 3.1. Creating game base and floor – 17th July
   - 3.2. Creating player character and implementing movement and actions – 19th July
   - 3.3. Creating enemies and implementing combat – 19th July
   - 3.4. Creating items along with implementing actions and effects – 19th July
   - 3.5. Implementing start, flow, and end game – 21st July
   - 3.6. Creating makefile – 22nd July
4. Testing and debugging – 24th July
5. Documentation, report, and final UML – 26th July

Responsibilities

The main core aspects of the project will be worked on together by all members, for example, game base (game.cc), character class, and items class. Once these are completed, the rest of the program will be worked on separately as follows.

Kabir Ashish Wahi	Player Characters
Thanh Le	Non-Player Characters
Haichun Kan	Items

Once these aspects of the project are completed, Individual portions would be tested, and correction would be made if required. There on, bonus features will be added if time permits. The debugging and testing will be done as and when one completes their portion of the project.

Q&A

Q. How could your design your system so that each race could be easily generated? Additionally, how difficult does such a solution make adding additional classes?

A. There would be an abstract base character class with player and enemy being subclasses. Adding additional races should not be too difficult as it would mean adding another subclass.

Q. How does your system handle generating different enemies? Is it different from how you generate the player character? Why or why not?

A. Enemies would be generated randomly in an chamber and on any tile. The generation of enemies would be somewhat similar as the generation process is the same except for spawn probabilities and the dragon as it only spawns with a dragon hoard or a barrier suit. The generation should not be too different as the only exception is that the player cannot spawn in the chamber with the staircase.

Q. How could you implement special abilities for different enemies. For example, gold stealing for goblins, health regeneration for trolls, health stealing for vampires, etc.?

A. Different attack methods could be implemented by overriding the attack function, which would be inherited from the character class. As for abilities not related to attacking, they would be implemented by adding additional functions to the characters class as that will be only applicable to them.

Q. What design pattern could you use to model the effects of temporary potions (Wound/Boost Atk/Def) so that you do not need to explicitly track which potions the player character has consumed on any particular floor?

A. We could use a decorator pattern to wrap the player with potion effects and generated a new player on every floor. This would allow us the remove and buff or debuff with ease. It would also enable us the apply effects on the player easily by wrapping the player character with the respective effect.

Q. How could you generate items so that the generation of Treasure, Potions, and major items reuses as much code as possible? That is for example, how would you structure your system so that the generation of a potion and then generation of treasure does not duplicate code? How could you reuse the code used to protect both dragon hordes and the Barrier Suit?

A. This would not too different from different race generation. All items come under the abstract item class and thus would be able to us the generate function from the superclass. However the compass is not treated as an item, but as a Boolean. This Boolean attribute is given to all enemies with one of the enemies on each floor having this as true while rest have this as false. To spawn the dragon with the dragon hoard or barrier suit, we would use a function in the game class to reduce repetition of code.