SWEN20003 Project 2B - Reflection

Overview of Changes

In my submission for Project 2A, I had originally intended to store the game history as an ArrayList of ArrayLists of sprites, where each full map was added to the outer ArrayList on every move made by the player. When undoing a move, the top sprite list was to be removed and the current sprite list would be reset. In my final solution, however, I opted to store individual sprite histories as a stack within the Sprite class. This meant pushing, popping and peeking the stack would make undoing much simpler and more flexible, while also providing cleaner code, as I avoided the necessity of performing deep copies.

There were also some changes to the classes used, with the Block, Tile, and Unit classes from Project 2A being removed, and the Coverable, Attacker and SpriteHistory classes being added. Coverable and Attacker were used instead of Block, Tile and Unit because I felt they suited the new structure better, and SpriteHistory was added due to the change in game history storage mentioned above.

There was some ambiguity in the interpretation of how the TNT block destroys the Cracked Wall, with the options being an explosion caused by the TNT being pushed directly into the Cracked Wall after already being in contact with it, and the TNT exploding immediately on contact. I chose to implement the explosion on contact because it made the gameplay feel smoother, and would also allow for more flexible potential level design (i.e. a level with a single-tile wide corridor and a cracked wall on one side).

Additionally, I decided to have blocks behave such that the blocks cannot run over the player. For example, I designed the ice block to stop should it run into the player, but continue to move after the player moves out of the way. On the discussion board, it was stated that the ice block should not stop when blocked by the player, with the reasoning that this behaviour would make the first ice puzzle trivial. I feel that my final implementation avoids making the puzzle trivial, since the ice block continues moving after the player stops blocking it, while also making the game feel more robust (e.g. on level 3, the rogue can push the stone against the player without having the stone and player occupy the same space, which wouldn't make much sense).

Difficulties Experienced

One difficulty I experienced was adhering to the UML diagram I made in Project 2A, which was a result of not spending enough time on fully designing how I wanted the final project to look. This meant that a significant portion of my time was spent changing the structure of the project and debugging the unforeseen effects of these changes.

These issues led to a lot of prototyping and a myriad of ad hoc additions made to the code with only the goal of functionality. This led to further issues nearer the end of the project, as I had to go back and overhaul several portions of code in order to complete other sections of the project. This issue

was compounded by poor concurrent commenting on one or two occasions, which only made the process more difficult and tedious. Therefore, the debugging process was certainly a difficulty.

Before I decided to change my method of storing game history, I found my original idea of storing the entire sprite ArrayList within a game history ArrayList to be difficult, and spent a lot of time on trying to get that method working. This also applies to other areas where I had difficulty producing functionality, due to initially not using object oriented concepts such as inheritance and access modifiers correctly.

Key Piece of Knowledge Gained

Due to my plan being constantly changed and redeveloped throughout this project, I feel that the importance of spending more time on design cannot be understated. This seems particularly important for projects such as this one, where there are numerous components interacting with one another.

Of course, since this subject has been my first experience with object oriented programming, I would say this reasonably involved project has also helped develop my understanding of object oriented concepts through actually having to implement them myself. These two pieces of knowledge also appear to go hand in hand, as I have found that software development with object oriented principles requires more rigorous planning than I had expected from my experiences with other languages.

Things I Would Do Differently

In more specific terms, if I were to do this project again, or a very similar one, I feel that my initial approach to defining the abstract classes would have to change, as the additional classes I used in the final project differed significantly from my original plan. For example, the Unit class I used in my plan had the Mage, Rogue, Skeleton and Player as subclasses, which didn't make sense in the implementation I wanted, and was an issue I should have resolved before starting to write the code.

Furthermore, while commenting was not a huge issue throughout this project, there were one or two times when poor commenting meant rewriting chunks of code, as mentioned earlier. I would say this is an area that could still use improvement for future projects.

Finally, as mentioned earlier, I felt that the biggest obstacle during this project was the lack of a single clear plan. Therefore, the aspect of development I would focus much more on in a future similar project would to be spend a lot more time making a robust plan and structure before beginning to write any code.

Word Count: 940