CMPT 276: Group Project

Spring 2020

Phase 2- Group #9

Overall:

At the very beginning, we think the most difficult part of this phase of the project was generally the question of how to start the project. Considering that many of the group members have never collaborated on such a large-scale project, we really struggled with the equal delegation of work among all the members. Eventually, we decided that the best way to distribute the work was looking at our UML diagram/use cases and then assigning a group of classes for an individual to work on and then hopefully we could combine everything together when all the classes were complete. We spent the initial few days and meetings together learning git and setting up the project in Maven, as it was the first time any of our group members have used Maven. After that, the most crucial question was how we wanted the game to play out, for example, did we want the game to be grid-based and only allow the players to move along a 2D array or did we want the movement to be more smooth and flexible. Upon determining the general requirements, it became fairly easier to stray off into our own individual tasks and also including additional features.

Differences from Phase 1:

Although we had a fairly straightforward UML diagram, it was far from perfect and it turns out we actually strayed off from the original visualization of our game quite a lot. For example, as stated before, we originally visualized our game to be a maze-like game that contains a 2D array with all the overlying objects, such as the walls, characters, enemies, etc. This would consequently result in all the players and enemies to become grid-like and resemble the movement of a typical 1980 Game & Watch game. However, we collectively decided that aesthetically, it would be much better to have a general wider map and using key listeners and update methods that would move the player and enemies at a constant and steady pace. This would consequently make the movement look infinitely smoother than the original idea. However, this came with some problems as the old UML idea heavily focused on the concept of the 2D array. A lot of the class's methods, objects, and interactions between classes had to be heavily modified to support a frequently updating thread-based game.

Another feature that we tweaked from our phase 1 write-up was that enemies move only when the player moves. We thought this would give another aspect of strategy and difficulty while the player is conceptualizing methods to run around the enemy to get to objectives. We decided to change the concept of traps a lot in our phase 2 implementation as well. Originally, a trap was supposed to be a tile causing a door to open a room and cause an enemy to grab the

player from the room. Instead, we decided to make rooms that would cause a movable enemy to spawn, if the player opened the door, giving the chance for the player to run away. The trap doors could also spawn a bonus reward however that could give the player extra points and also act as a checkpoint for the game. We decided to separately scatter traps around the map that the player must avoid. One extra feature that we did not decide to add was the way a trap forces the player to die and respawn. We thought it would make the game too difficult and would not make sense if a trap made the player suddenly start at the entrance point of the game. That is why we decided to make a trap that causes the player to lose a life but still allows the player to continue from that point. Similarly, when the player is caught or has hit a trap, we decrease the player's score but we do not respawn the items that have already been collected. That way, there is a sense of desire and angst for the player to collect everything without getting caught.

Another change is that we decided to choose a different theme and setting of our game environment. Our original idea was that the player was an office worker trying to poison his evil boss, but after analyzing how the enemy moves, we decided to change the story to an adventurist who is exploring an old haunted dungeon for treasure. Similar to how the ghosts from Super Mario Bros. only move when the character is looking away from them, we wanted to emulate that uneasy and spooky feeling of getting chased by a ghost, by having our monsters move only when the player is moving. This makes it seem that the enemies are unnoticeably getting closer and closer to the player.

Management process:

As stated before, we decided to follow the agile design method and a modified scrum process flow to develop our game. This means that we would meet around once or twice a week to discuss the requirements of the project in which we would develop a backlog of tasks to accomplish by the next meeting. These tasks would not be too long, for example, a task would be to add a feature in the game like allowing the player to pick up items. Then the tasks would be delegated to each member and every week, we would accomplish these goals or discover new tasks that needed to be accomplished. Our distribution of tasks was fairly simple, we would have the list of the things that needed to be done and whoever is more comfortable finishing a particular task would choose that task to complete. Whoever had the most confidence in their coding ability would take the remaining task on the table. That way, we would all have a task and accomplish an equal amount of work for the project.

After the meeting, we would go code together for several hours, so if we came across any problems, we could resolve them right away. After we went home and accomplished the given tasks and at the next meeting, we would report our achievements. Obviously, there would still be

communication outside of the meeting through our main means of communication, Discord. Moreover, most of our work would be done through our branches, and it is usually at our meetings where we would decide to merge the code and work through any merge conflicts. All in all, our communication and organization were fairly well-done, however, as it came closer to the due date and with the confusion of the current pandemic situation happening right now, the management of our project became a little frazzled. So in the future, we would want to be able to quickly adapt to sudden situations.

External Libraries:

Besides the basic java library functions that include Lists, ArrayLists, etc. we also included:

JFrame - used to build the window in which the game is used to take place.

JPanel - used to control the locations and the display of the map, player, enemies, etc. on the window frame

JComponent - allows the player, enemies, rewards, traps to be drawn on the map in which will be displayed on the window frame

KeyEvent - actively record the input of the user to constantly update the player

BufferedImage - used to handle the image to color the map

Timer/TimerTask - used to have a constant game timer

Math - used for math calculations between the enemy and player

Sprites were referenced from https://pixel-poem.itch.io/dungeon-assetpuck

Code Quality:

When designing our classes and the interactions between them, the main rule we tried to follow was simply the rule of low coupling and high cohesion. For the most part for a given new class, we would want the class to only have methods that are associated with only itself and we would avoid a situation where an object calls a method from an object that calls another method. Another code quality feature that we wanted to follow was to have good naming habits, so for

functions, we wanted to be able to write a piece of code that could be understood just by reading the names. However, as our code involves multiple complications of java JFrame libraries, the code naturally becomes a little messier. Therefore, at that point, we would provide very descriptive comments for the functions on the functionality of the code. Similarly, we provided descriptive JavaDoc comments for our classes to describe the functionality of each class.

Challenges:

Considering that for most of us, this was the first time we have all worked on such a large and complex project that requires a lot of planning and organization, it is only natural that we faced several challenges during the process. The first challenge being the first objective in the project, starting the assignment. First of all, none of us knew how to use maven so we spent an entire meeting setting up the project and getting everyone on the same track. But after that, we realized that we did not know how to start the code since there was no foundational code to work on, which meant we were not able to separately work on different classes. For example, one of us was not able to work on the player class because we could not test it without the implementation of the map. Another issue would definitely be the complications of working with Gitlab. Most of us never actually worked collaboratively on a project so most of us have never before worked with git. This caused massive confusion towards the end when we trying to merge everything together as we would encounter merge conflict after merge conflict, or accidentally merging or pushing to the master branch which continually causes branches to be outdated and other further complications. Lastly, our last challenge would definitely be this sudden given situation of the current outbreak that is occurring even now. Considering that, a lot of us are forced to practice social distancing, we were forced to do online meetings rather than physical meetings near the due date. This was a major challenge for us, considering a lot of problems with our code were solved by showing and talking through the problem. This was definitely much harder and more frustrating doing problem-solving through social media since it is harder for everyone to be on the same page. In the future, we hope to further choose different approaches when coming across the problem. For the first challenge, we just had one person start the project and then we built of it, but next time we are hoping to build and understand the foundation together in order to make it easier to implement things later. Hopefully, through practicing git more often through this phase, our understanding of git has improved. Lastly, due to the outbreak, we have discovered that perhaps we may need to improve our method of communication when we are not together. Overall, all our experiences that we have endured throughout this period will definitely be referenced and utilized for not only phases 3, 4, and 5 but for our future endeavors as well.