Phase 2 Report

For the second phase of the project, we started with researching on how to implement several aspects of our game. Namely, setting up Maven build, how to create a GUI, and finally how to combine that with the core logic of the game. We started with dividing each feature of the project into a task and further divided it into sub tasks so that more people could get involved within each feature. This was crucial for implementing the GUI as we lack experience in that field. Along with that, we created other tasks for implementing core logic such as implementing pathfinding for enemies, listening for keyboard input detecting collision, updating score, etc. We were not able to fully implement the game, but we were able to understand the route needed to complete the project and create proper tasks accordingly.

Setting up Maven for the project was relatively simple, but implementation of the GUI proved quite challenging and complicated the workflow. Several changes had to be made to our initial design, which consisted of changing our existing classes along with the addition of several new entities. We adjusted our initial entity classes such as Player, Enemy and Collectibles to record position on the map as well as methods for loading and rendering the images for displaying on the GUI. We had to add three new classes. The Panel class controls the entirety of the application such as the GUI, game loop and manages several properties such as the dimensions, game assets, frames, and setup. The KeyListen class is used for capturing the logic around keyboard events. The Background Controller is responsible for managing the environment and the map of the game. Finally, the CollectablePlacer for setting up the rewards on the map. So far these are the changes that had to be made and there is room when we are adding in the rest of the features for our game.

As a group, we separated the game into multiple tasks. We separated tasks based on requirements. The first few tasks, such as setting up the GUI and implementing the game loop, had to be done sequentially, to set up a base for everyone to work on. After that, we tried to split tasks in ways that would allow multiple people to work on different tasks at the same time whenever possible. For example, we had the Player class worked on concurrently along with the Reward class. We would work on tasks in our own personal branches, and then send a merge request when we completed the task. Another team member would then read over the code and make sure everything worked before merging it into the master branch.

For external libraries, we used a couple from JavaX. We used Swing for the GUI. We chose Swing because it was the most commonly used, so there was a lot of documentation and tutorials about how to use it. We also chose it because one of our members had previous experience in using it. Swing let us make a panel where we could draw objects onto the screen. We also used the imageio library in order to read some images and use them in our game. As part of the GUI and inorder to receive keyboard input, we used Awt. Awt let us implement KeyListeners that would trigger code when certain keys were pressed.

We took some measures to enhance the quality of our code by following an easy-to-understand, consistent style. To help accomplish this, we followed lowerCamelCase for naming variables and UpperCamelCase for naming classes. We also made sure to define properties at the beginning of each class followed by the methods. Each class was in separate files, and similar classes were grouped together in one directory for better navigation. Additionally, we added JavaDoc comments for functions and general comments in the code to improve readability. Lastly, each of

us worked on separate branches which were later merged to the master branch after a code review was done. It also helped to mitigate conflicts, retain commit history and work with ease without interfering with each other's work.

During this phase, we dealt with several challenges in each phase of our work. The first challenge we faced was getting started with initializing Maven build for our project as we were unfamiliar with it. Some group members also faced challenges with using git when it came to resolving merge conflicts due to inexperience. Dividing up the task to fit each of our busy schedules was far more difficult than anticipated given our varying degree of experience, interest, and knowledge. However, the biggest difficulty we faced was during implementing the GUI. Only one member had experience in Java GUI development but in a different scenario. It introduced several flaws in our initial design and demanded several adjustments that we had not anticipated as we assumed we could keep the GUI separated from the core logic. Figuring out how to implement a game loop in which the entities (player, enemies, collectables) are updated regularly along with keeping track of keyboard events was unexpectedly hard to figure out. Another issue was some members of the group were unfamiliar with java so they had to learn and get familiar with it which made this phase take longer.