Nov 6th, 2022

# CMPT 276 Project Phase 2
# Implementation: Report

Maze to the Queen Bee

Group 16

Marco Lanfranchi – 301433830
Sana Dallalzadeh Atoufi – 301399601
Satvik Garg – 301452798
Jinshuo Zhang – 301385111

# Contents

Implementation Report…..... 2
Management Process…..... 2
Sana's Report……..…..... 2
Stuart's Report..……..…..... 2
Satvik's Report.……..…..... 3
Marco's Report.……..…..... 4

# Implementation Report

## Management Process

To start phase 2, we had an initial meeting to plan everything and during this we decided to have at least one meeting per week to update each other about the progress and make sure everyone was on track. We also had meetings whenever an issue came up and we provided each other help whenever we could. We began by assigning the UI and Map to Satvik, the Bee to Marco, the Enemy to Jinshuo, and Rewards to Sana and decided to focus on implementing these first. We then started working on our parts and pushed sections regularly to Gitlab. A lot of communication was done over discord and online meetings and all group members communicated in these chats regularly. Although group members were responsible for their assigned parts of the game, we helped each other with the classes we made and made new classes for other features once these were done. In the next 4 paragraphs, we will share the individual reports from each group member speaking about their work over the span of phase 2 and then finish with a conclusion from the group.

## Sana's Report

My part was to work on the Rewards classes. There were two types of rewards, bonus, and regular. The regular rewards were in the shape of honey drops and they had to be collected by the bee in order to collect points. The bonus rewards were in the shape of honeypots, which would give the player more points than a regular reward if they got collected. The honey pot appears randomly in the game. Creating Rewards and making them appear at some point and randomly was a little bit challenging since I didn't really have any experience working with java. However, by looking at some codes and practicing I was able to work on that. I had to understand the requirements very well and understand what the code should do. However, it was a challenging and fun experience.

## Stuart's Report

In phase 2, my part was to create enemies and make them chase down the Bee. Creating enemies was easy, but I was confused on how to make them chase down the Bee. This was really hard so I needed to find an algorithm which can make enemies to find the shortest way between themselves and the Bee. I choose to use A* Path Finding Algorithm to help enemies to chase

down the player, but before applying this algorithm into our project, I had to understand how it works and if there is somewhere we need to change in our code. A* Path Finding Algorithm divides the entire map into multiple nodes, and measure both distance to the goal node (the Bee) and distance to the start node (where the enemy was born). It will label all nodes with values of these two distances and the sum of these two distances. Then always finding the node contains smaller values and sets this node as the alternative nodes and use this new node to find other nodes which are smaller than it. If there is a wall or cannot pass, then it will start another alternative node to find a new path. When the path between the Bee and the enemy is found, it will track back the path from the Bee to the enemy. If the Bee moves to a new direction, then the algorithm will find the path again to update the new position of the Bee. This algorithm causes me few days to finish and debug in our code. It is not hard to find if the next node is a wall, but hard to find which direction should be chosen when the enemy is standing in a cross, and if the direction can be used in the final path because of the collision between the enemy and the door. Because the enemy is not moving one tile at once, therefore sometimes their collision will happen and the enemy will get stuck.

## Satvik's Report

Throughout Phase 2, I was responsible for all the UI and map generation for the game while also helping my teammates with random reward generation. The biggest challenge I faced was with map generation and creating a map with hexagonal shapes as discussed in our Phase 1 paper. Initially our group had decided to create our map in the shape of a honeycomb with hexagons to fit the theme of the game. But as I did more research, I realized that creating hexagonal shapes would not only be difficult to create in terms of the player movement but also it would not look as clean as the blocks will be in a zig-zag shape. After I brought up these issues with the group, we decided that due to the time constraints for this phase, we would do a simple grid shaped map. This decision significantly reduced our workload and allowed us to focus on the other parts of the project such as player movement. After we decided on the shape of the map, my next challenge was creating a text map for the code to loop through and place the tiles within the GUI. This was especially difficult because I had to think of a way to create a map of size x and y with z number of rooms and doors in the middle of the walls. After multiple tries and research online, I was finally able to come up with a solution that was robust enough so that I could give it any number and it would give me the correct output. These two parts of this phase proved to be the most difficult for me.

There were many changes in the Map package compared to Phase 1 and many of these changes came from our realization that the classes we made in the UML diagram weren't in fact needed. For instance, the Room and Barrier classes had no use in our game, as there were no specific Rooms in the game and the walls were spawned using the text map with no special function

other than preventing the user from going through. Additionally, after realizing that having a whole “room” as a PunishmentRoom would be too difficult for the player, thus that class was removed as well. The Map class was changed and named the TileManager and Tile classes which were responsible for the spawning of the tiles and creating the UI. Moreover, we had originally planned to also have locked doors, but due to time constraints and inexperience making games in Java we decided to stick with all unlocked doors, but to have more traps and enemies to keep the difficulty of the game at the same level. Another aspect that changed from the initial idea was the Environment package. As we finished the main parts of the game such as player movement and enemies, we realized the game was a bit too easy as the player could easily dodge the enemies and traps. So, to increase the difficulty level of the game, we decided to add a black filter to the game, so the player could only see a certain radius away from them. This made it so that the enemies and traps were not seen as easily, thus making it harder to avoid them. We also decided to add sound to the game as it added to the immersiveness to the game and made it fun to play overall.

## Marco’s Report

For phase 2, I was responsible for making the game’s main character, the bee, and implementing its controls. In addition, I made all the games pixelated images, helped my partners with their classes when needed, and made changes to mine and others’ code as the phase went along and the game began having more details. The group did not have many changes to our use cases from phase 1, however due to a design choice that was made at the start of phase 2, which was to not have any locked doors on the map, we did not implement use case 5, the case of the player trying to go through a locked door. Apart from 2 of the use cases which involved changing the current state of the application (starting a game and pausing a game), I found that I was able to implement all the rest of the use Cases through methods existing inside the classes I worked on in the project’s entity package. In our UML class diagram from phase 1, the “character” package represents what is now the “entity” package in our project and it contains all classes for the game’s moving characters. This package had a few changes from the initial design. The Entity class, an abstract class I made to be extended by the game’s different types of entities (bee and beekeeper enemies) remained as part of the package and did not see many changes itself. I used this class to hold the fields that these entities share such as speed and multiple images, write methods to avoid colliding with walls since neither entity can walk through a wall, and write methods for getting an entity's location on the map. Next, I began working on the game’s main character, the bee. From our initial UML diagram, the class called “Player” is now called “Bee” and is a subclass for the “Entity” class. This class represents the game’s main character, and it contains a field to hold all its rewards, methods to switch its images to illustrate the bee’s wings flapping, methods that implement the use cases mentioned above, and more. In order to control the bee with keyboard inputs from the player, I made the “KeyHandler” class found in the

“main” package which implements the Java AWT KeyListener interface. In this class, I implemented the methods to listen for direction keys being pressed and released, and Satvik implemented methods detecting keys pressed which change the game state such as pressing enter to start or esc to pause. One change in the entity package from the UML class diagram was the removal of the “EnemyGenerator” class. Since we decide enemies would all initialize with the same values for their fields other than a randomly generated starting point, this class was unneeded and instead our GamePanel class contains a list of enemies. Outside of the “entity” package, we decided to create a new package called “objects” which has a “SuperObject” class and many subclasses of this class. Along with an “ObjectManager” class used to organize a list of these objects used in the game, these objects are all simply classes that contain an image and dimensions to display some image on the screen with no movement required. Some difficulties came along in the later stage of the phase when it came to implementing collision checking for the bee with rewards and enemies. Since the game had a high amount of frames per second each at which the game’s objects were being updated, the bee was set to move much less than the length of one tile. When it came to rewards being collected, they were the size of one tile and there were not many issues, but when it came to not overlapping the enemy and player before colliding I found myself stuck checking when they collide. This brought me to talking with the group and coming to realize that some of the code needed to be enhanced such as the values set for fields as the distance entities moved were set at undeclared values instead of making use of the tile’s size. After making those adjustments and testing bee and enemies map locations with print statements the issue became much clearer and we were able to stop the two characters from going over each other. Towards the end of this phase, I added a “Sound” class which is found in the “environment” package. This class was used in the “GamePanel” class to play sounds when the bee goes over a trap tile, collects a reward, and loses or wins a game. I used relevant interfaces from the Javax Sound package to play and stop the .wav files. This was my first time working with sound in any type of coding and was very fun, however slightly challenging as I ran into issues playing music while at the game’s title state and therefore did not get to adding that.

## Conclusion

During the last few days of phase 2, the group worked together to point out any errors we noticed while playing and testing the game and we helped each other to get these fixed. We also made small changes such as the number of rewards in the game, the time limit a player has, the number of enemies in a game, and more changes in order to make the game what we felt was the perfect level of difficulty. Along with this, we all added JavaDocs comments for the Classes/methods we worked on and kept each other updated and aware of what was left to work on until the end of phase 2.