Phase 4 Report

THE GAME:

Our game is a 2D RPG where you play as the hero and collect keys to unlock the exit and escape the pursuing enemies. Below are some key features of our game:

- Control your hero with the arrow keys to move across the map. Press 'p' to pause the game.
- Collect all the keys to unlock the endpoint to beat the game
- Avoid pursuing enemies which deplete your health bar
- Avoid red and black bombs which reduce your health and total score respectively
- Collect bonus rewards to increase your score and help you along the way:
    - Hearts will restore your health bar
    - Ice will freeze enemies in place for a short duration

FAITHFULTHNESS TO ORIGINAL PLAN:

In terms of game design and functionality, we've remained faithful to our original plan from phase one. Our game is still composed of the components we outlined in the plan from Phase 1: There's a single hero, a dynamic enemy which chases the hero around, two varying static hazards, a bonus ice reward, etc...

The gameplay elements all function as we imagined they would to begin with as well: The hero and enemies cannot pass through barriers, the hero takes damage on hit with an enemy or a health-bomb, the bonus reward freezes all enemies in place for a given time but will disappear if not collected within a certain amount of time, you can pause and end the game, the game can be restarted at the end, etc...

Our game is displayed as a board containing all the interactable components, as well as a display for the players current score, heart and key count. The board is divided into individual tiles which house the separate components of the game such as barriers, rewards or hazards.

Overall, our game has remained faithful to our original plan at a higher level of design. However, underneath the surface level of the game design and functionality, we've had to make quite a few changes to our actual code including adding additional helper classes and adapting our design to work with different frameworks and toolkits.

DEVIATIONS FROM ORIGINAL PLAN:

The biggest changes we implemented in our project was without a doubt within the code and our class structure. To begin with we had to add in several new classes that would serve as helper classes to implement features such as game textures, GUI, different menus and music. The main reason for these changes was to accommodate our choice of GUI using Java's Swing libraries and toolset.

One of the more impactful changes we made was creating a Main class which took over our original board class as the driver class which would run the games main logic as well as contain all the necessary instances of each game component. Doing this allowed us to easily propagate updates of the games current state throughout our code using the Main class's update function which would in turn update each individual component's own state depending on user input and component interactions.

Our Player and Enemy classes received several additions which were necessary in order to flesh out their movement and collision logic. This required finer tuning than anticipated in our original plan because we decided to have our game move smoothly and update more frequently rather than our original plan of only allowing actors to move one tile per clock cycle.

Features we added in our final version which were not in our concrete plan for phase one include, multiple level selection and music/sound effects. To facilitate loading in different levels we added a Map class which would map the elements of our game board to integers which we could then easily read in and correspondingly populate that instance of the game with. We also added in background music as well as sound effects for picking up keys, rewards or hazards, and two tracks that would play when the player won/lost.

Some notable features we decided to remove for our project were a restart function, which allowed the user to restart the main game, and a high score tracker, which would maintain the users highest score even after closing the game. Regrettably, we had to make these choices due in part to time constraints. As our project got closer to completion and the deadline approached, we had to decide to implement only the core features we wanted and needed for our game to function as we envisioned.

LESSONS LEARNED:

Throughout the course of our project we came to understand exactly how important the planning/design/modelling phase truly can be and how having a well thought out and thorough roadmap can significantly facilitate the implementation process. Although it is important to make sure you have a good design in place, we also discovered that it is often very difficult to develop an all-encompassing plan that accounts for everything.

More often than not, we had to revise our plan, not by choice, but out of necessity of the tools and frameworks we were working with and also sometimes due to real-world time constraints. This project was presented in a vague manner that left most of the implementation choices up to our group, for example we had complete freedom to choose how we would develop our GUI and what external libraries we would like to use. Despite choosing a firmly established and well documented library (Swing), it still took a lot of reworking of our initial design to implement our GUI as we learnt how to use Swing. On top of having to create the GUI we were also left with no restrictions on any sort of bonus content we wanted to add, e.g. extra hazard or reward types.

In conclusion, we found that the design process is just as, if not, more important than the actual coding process in software engineering. However, even with a well thought out design, there's often unforeseeable circumstances (e.g. client requests, time constraint) that require revisions to the initial plan in order to proceed, or even sometimes a more efficient implementation comes to light.

TUTORIAL:

As stated at the beginning of this report, our game is a pretty simple 2D RPG where the user takes control of the hero character with the goal of collecting all the keys on the map to unlock the exit point. Upon start up, the user is prompted with a choice between two separate level designs, which then loads in the appropriate map. To play the user inputs directional arrows (up, down, left and right keys) to move their hero across the board. The player should aim to avoid moving enemies as well as all static hazards which reduce the players score and health if they happen to collide with one. To help along the way, the player can opt to collect bonus hearts that are scattered along the map to increase their health count or an ice reward which will freeze enemy units in place for a short period of time. However, if the ice reward isn't collected within a certain amount of time it will disappear from the map, leaving the hero without any extra help. If at any point the user presses 'p' the game will pause, and a menu will appear allowing the user to resume the game or exit if they so wish. The game ends when the user exits (via pause menu), collects all keys and makes it to the end point, or runs out of hearts due to enemy or hazard damage.

VIDEO TUTORIAL LINK:
https://vimeo.com/408712691?fbclid=IwAR3qDD7wGLnzpigiy82DPTWqjJMJoqnjkij8jbERg4gvE46f9FBhPyB6-ug