Phase 2 Report

GitHub Ids:

- Ryan = rat2
- Jason = xgill15x / jga132
- Luan = luan_nguyen_3@sfu.ca
- Zihao = zxa43

1. Overview

In phase 2, our group was asked to implement the game we had designed in phase 1. Since most of us are in our second year, there was a lot of uncertainty about how to even begin on a project of this magnitude. There was concern about the level of experience our group had collectively, and we knew we had a lot of work ahead of us. Implementing this game was not like working on any regular ol' assignment. There were many real-world concepts that were crucial in making our game function the way we wanted it to. In addition to having to learn proven game development practices, this was the first time most of us worked on a coding assessment in a large group environment. Unlike when you work on an assignment by yourself, you don't always have the liberty to work on your own time. There were sacrifices and commitments made by everyone in the implementation process to ensure that our group was being considerate of everyone's effort and time. We soon came to realize how important communication would be in order to get this project done efficiently and we tried to make it a priority for ourselves. The overall plan to implement our game was to first research the common ideologies that we'd all need to understand in order to put the work ahead of us in perspective. This included concepts like creating a central game loop, setting up inputs from the user, and having our game acknowledge those inputs along with other events and output the appropriate response. Once we all had a mutual understanding of how to proceed with implementing the design we had planned for our game, we moved on to learning how to manage our unified progress as a group. Since we'd be working independently from each other for the majority of the project, we all caught up on our understanding of version control and made sure we used Git to its fullest extent as it would be our friend throughout the duration of the project. With that done, all that was left was to determine who would be responsible for which parts and getting the project done in a timely manner.

2. Adjustments to the initial design

Overall, phase 2 closely follows the initial design of our game that was mocked up in phase 1. The UML diagram did not need to be altered for this phase as there were no problems implementing the classes and relations featured in it. The few adjustments that we had to make had to do with some of our use cases along with the original plan & description. With our use cases, we stated in most of them that the user starts off by clicking a start button on a start screen to enter the game. This is something that we didn't quite get to create because we underestimated the time it would take to implement the key features of the game. Currently, the user is thrown into the maze and it's game time right away. We determined that it was more important to get the game itself working correctly before we moved on to the 'nonmain' parts such as the start screen or an optional exit game feature. This is something that we plan to add on in our phase 3 along with testing to make our game have a fuller feel. The other small changes we made had to do with how the score would be updated depending on which sprite you interact with. Essentially we just altered the amount the score would increment/decrement when it interacts with another sprite.

3. Management process and division of roles

The management process and division of roles for this assignment went fairly smoothly. Regarding the management process, we tried to ensure that no group member had too much work on their shoulders at a time, and the work was distributed evenly among group members. We had a few forms of communication that we utilized to enforce these constraints, the first of which was holding group meetings once a week depending on everybody's schedule. These group meetings allowed us to concretely discuss our ideas and accurately determine the progress of our work, along with what was remaining. It was also a time when we could hold each other accountable for what we were responsible for, which helped keep the project on track. The second form of communication we used was Discord. We used this platform as a place to get across quick thoughts and messages, which would later be addressed in our weekly meetings. This combination of long-term and short-term communication ensured that we were all synchronized whilst implementing the game and the workload on each member was just about equal. As for the division of roles, we started off by assigning at least one large independent task for each member of the group to complete. After all the large tasks were completed, it was a matter of picking up leftover parts when a group member finished the one they were assigned. Some of the larger parts we tackled first were implementing the game loop, handling user input, creating the map logic, and handling collision logic between different entities. This gave us a solid foundation to implement the other behaviours of our game such as the monster pursuit logic, disappearing bonus rewards, and score updating.

Some specific examples of division of roles:

- Ryan - Gameloop, AI Monster behaviour, Collision checker, Projectiles

- Jason - Diamond spawning, Collision checker, Game state, Map tiles, Report
- Luan - Object creation, Hitboxes, Collision checker
- Zihao - Game HUD, Map tiles, JavaDocs

4. External libraries

There were not many external libraries used in our projects aside from ones that dealt with the graphical user interface. Since the project was done in Java, which is a relatively developed language in terms of libraries, we decided to use JFrame & JPanel objects from the Java Swing library to contain our application. In addition, we used the Java Abstract Window Toolkit library to take care of the graphics that needed to be displayed on the window. These libraries were not initially our first choice. Our group considered using frameworks and libraries such as LibGDX and JMonkey, respectively, which arguably have better game development tools; however, we came to the consensus that the built-in Java GUI libraries would suffice for the task at hand. After researching how to implement our game design, we determined that anything more than the built-in java libraries would be overkill as we most likely wouldn't even use all the excess tools provided by them for a simple 2D maze game.

5. Measures taken to enhance code quality

There were several measures we took to enhance the quality of our code whilst working on this project. The first of which was using proven methods to implement the core concepts of our game. We researched many ways to implement concepts such as the passage of time and multiple game threads, etc. and chose the approaches we felt best fit our project. Using these conventional methods allowed us to avoid rookie game-developing mistakes and ensured robustness within our development. Furthermore, we applied the concepts that were taught early on in the semester. The various classes in our game were carefully mapped out and designed with the relationships between classes in mind. For example, sprites on the map that exhibit some "living" behaviour all extend an Entity class, whilst inanimate objects extended a GameObject class. This hierarchy between classes is a logically organized structure showcasing the extent of object-oriented programming. Another measure we took to ensure the quality of our code was to make sure it was readable. In addition to utilizing Javadocs, all members of our group took the time to make comments on potentially confusing logic for each part they were assigned. Although these comments may seem redundant because we added Javadoc comments, we kept them in order to explain the specifics in our code. To summarize, to enhance our code we researched and implemented proven game development techniques, applied java concepts that were taught in class, and ensured readability with Javadocs and other comments.

6. Biggest challenges faced

As we mentioned before, some of the biggest challenges we faced had to do with our inexperience with game development. One of the challenges we faced was learning the many new concepts introduced to us whilst implementing the game. It was a little overwhelming to do all of this given a deadline. Nevertheless, we rolled up our sleeves and did the best job we could. Another challenge we faced was finding time to get together to work on and discuss our project. All of the members in our group have drastically different schedules which made meeting up in person incredibly difficult. We found ways to bypass this with online meetings but those too had their limitations. Some challenges that we faced relating to the actual implementation of the game were fixing buggy collision detection, understanding how to implement the monsters' dormant and active behaviours, and implementing the spawning and despawning behaviour of the diamonds (our bonus reward). The technical issues we faced along the way definitely had a steep learning curve but ultimately gave us a better understanding of how to finish our game.