Phase 2 Report - Group 21

Describe your overall approach to implementing the game,

Our overall approach to the game was to break down the development into several tasks and implement the main use cases and classes we had developed in Phase 1 of the project. These tasks included the following:

- Map & Tiles System
- Player Class
- Reward & Punishment System
- Enemy Class
- User Interface

We planned to start off with first finding an external library for the GUI. After reviewing several options, we chose to use Java AWT/Swing. From there on we planned to implement the features in the same order as the list. First we made the map and tile system, which was followed by the player class and rewards, and finally the enemy class and the user interface. At the end we had to put the logic of the game together, such as what determined whether someone won or lost.

We distributed the tasks amongst ourselves and followed up on progress by reporting to each other through our group on Discord. Each time after working on the project, we planned on mentioning what we completed and stated what needed to be implemented next or any bugs that we noticed.

State and justify the adjustments and modifications to the initial design of the project (shown in class diagrams and use cases from Phase 1),

We had to make a lot of adjustments to our initial plan. As all of us were quite excited to attempt game development in Java, we had initially overestimated our ability to get all of the requirements done within the short time frame. In order to adjust for the deadline, we re-engineered our requirements. We removed some features, such as the fighting mechanics, final boss, and scoreboard, as these would take a lot of time to develop and were not required by the requirements given to us in Phase 1.

Apart from removing requirements, we also had to implement many new classes and operations that we did not initially think of. These were mainly for drawing and updating the objects, checking collision, and inserting assets to the screen. This resulted in us also having to update our UML diagrams and use case documents to reflect the changes we made.

Explain the management process of this phase and the division of roles and responsibilities,

We used a software called ClickUp to manage and distribute the work. Initially we broke down the work into separate classes, and then from these tasks, we broke them down into smaller subtasks. The following dashboard depicts our task list and distribution of work: https://app.clickup.com/14235132/v/li/198620771

Each of us volunteered for the different tasks by assigning ourselves work through ClickUp. Using git made distributing the work really easy as we could easily each work within our own branch and then push to the master branch when necessary. After completing a task, we updated each other through Discord on what we had accomplished and stated what needed to be worked on next or fixed.

The distribution of work ended up being roughly the following:

Jun: Player class, map and tile system, and reward and punishment system.
Fayez: Enemy class, and pathfinding algorithm.
Qijun: User interface, and sound.
Akki: Build automation, and unit testing.

List external libraries you used, for instance for the GUI and briefly justify the reason(s) for choosing the libraries,

The only libraries we ended up using were AWT/Swing. This was quite a simple library but also provided all the functionality we needed for our game. The other libraries which we looked at such as LWJGL and LibGDX were meant more for graphics intensive games.

Describe the measures you took to enhance the quality of your code,

To enhance the quality of our code, we tried to write good comments as we were coding so that we could understand each other's code. We also stayed persistent with having Javadocs for each class and each of its methods.

Furthermore, we made use of certain features which the Java language provides such as inheritance and abstraction. This limited the need to rewrite code and allowed for better organization. A CI / CD pipeline was added to automate the compilation process And check if the pushed code works or not. The pipeline is connected to the repository using A SSH token and is triggered on each build.

Finally, near the end of our development, we reviewed the code to clean it up. Removed any unnecessary variables and comments that were no longer needed and adjusted the access modifiers for the different variables.

And discuss the biggest challenges you faced during this phase.

We faced many challenges throughout this phase and learned a lot about software development and game development.

The first challenge we faced was selecting an external library. Initially we had explored many libraries such as LWJGL and LibGDX, but we found that these libraries offered a lot more than what we needed. They were more meant for graphics intensive games and would also require learning a lot of new apis such as GLFW, OpenGL, OpenCL, OpenAL. As we did not have that much time to develop the game, we wanted to use a much simpler library. In the end we decided to use AWT/Swing for our GUI.

After this we faced the challenge of distributing the work. We overcame this challenge by using a task management tool called ClickUp. We started by breaking down the game development into several of the main classes. We then broke these classes down into certain tasks or logic that needed to be implemented. Then by using the ClickUp tool, we were able to volunteer and assign ourselves for tasks and distribute the work amongst ourselves.

Another challenge we faced was the early deadline. A lot of the features we had anticipated to add to the game would have taken much longer to implement. As a result we had to remove a lot of the requirements we initially had. This included removing the attack system, boss battles, and multiple levels. Due to re-engineering the requirements we had to redesign our UML and rewrite our use cases.