Phase 2 Report

Approach

We tried to take an iterative design approach to the development of the maze game project. We began with the implementation of the playable entity and its movement controls. Once the playable entity was functional, we moved on to implementing the map and its collisions. After the map was implemented, we implemented entities and their pathfinding. Later, we implemented traps, eggs, and keys. Finally, we polished the game by adding animations to the entities.

UML Updates

-

Management Process

We used GitLab issues to distribute and maintain a to-do list of features that needed to be implemented and bugs that needed to be fixed. Each issue was assigned to an individual to prevent multiple people from working on the same feature. This approach allowed us to stay organized and focused throughout the development process, ensuring that all tasks were completed efficiently and effectively. The use of GitLab issues also enabled the group to track progress and resolve issues in a timely manner, leading to a successful project outcome.

External Libraries

For building the GUI, we used JFrame as an external library. We made this decision because one of our team members was already familiar with it. Using a familiar library allowed us to work more efficiently and effectively, as we were able to avoid spending time on learning a new library. JFrame also provided us with all the tools required to complete our project with minimal issues.

Code Quality

To maintain code quality, we used GitLab merge requests for code approvals. Before merging any code, at least one group member reviewed and approved the changes. This approach helped us to ensure that our code was consistent, maintainable, and free of errors. The use of merge requests also enabled us to verify if our teammates' implementation was adequate before merging it with the master branch. This reduced the risk of introducing bugs into the codebase and allowed us to provide feedback and suggestions to improve the quality of the code.

Challenges Faced

During the development process, we encountered several challenges that required us to adapt and find new solutions. One significant challenge we faced was when certain implementations turned out to

be ineffective for the tasks they were required to do. For example, when we changed the render logic, we needed to rewrite the collisions code. Similarly, we had to rewrite the collisions and positioning code to fix pathfinding with enemies. This was a challenge because replacing the existing code meant that anyone who was implementing another feature that required the ineffective code would have to rewrite their implementation as well.

To prevent these challenges from becoming a roadblock, we implemented two strategies. Firstly, instead of rewriting existing implementations, we tried to add more functions while simultaneously leaving the older ones working. This allowed us to implement new features without disrupting the functionality of existing code. Secondly, we had everyone implement code that did not have much association with the code other group members were working on. This allowed us to work on multiple features simultaneously, reducing the risk of disruptions caused by rewriting existing code.