Code Review
Jaime Villanueva & Linden Ostergaard

We chose to do our code review mainly on the UI class. This class was incredibly long and held an overwhelming amount of information that affected the group's understanding of the class' purpose. There were several issues with this class, such as the number of fields, length of methods, lack of documentation, and bad variable names. Overall, this class had too much and didn't have a specific function.

First, we tried to reformat the code by improving documentation and renaming variables to improve readability. However, we quickly realised that there were simply too many things that this class was trying to accomplish. So we categorised the fields and separated them based on how interactive they were with the player. This way, the UI class would be used specifically to draw the images onto the screen, and the new UI_Information class would be used to hold the game's information regarding the player. For example, the user's play time would be moved from the UI class to the UI_information class, so that the UI would continue to update and display the time, but the information would be held in the UI_Information class. Other methods that were moved and modified were the spawn location, time for bonus rewards, title screen substates, and the reset info method.

Other than the UI Information class, we quickly scanned the lines in all other classes. Comments were added to several methods and lines that help provide readability to other developers. Cleaning up dead code and other irrelevant comments came naturally, as we questioned why each line was in the code if its function was not obvious. This improved the understandability of the classes in our game. Some methods in the AssetSetter were simply copied and pasted, so we modified those to use a single method where possible. The KeyHandler class was a little bit hard to follow along, so comments were added to improve its clarity and make sure the developer knew what each key code would do. All of the classes only used public methods, so we also went through those and checked their call hierarchy. If the method was only used in its own class, we changed it from public to private. In addition to changing access modifiers, we also restructured the code so that the public methods appeared closer to the top of the file.

However, when running some tests and playing through the game again, we discovered several bugs with the bonus reward spawn time and location. Eventually, we fixed the bugs by moving both setSpawnTime and setSpawnLocation to the AssetSetter and GamePanel classes respectively. Most changes did not affect the actual function of the code, as we simply renamed and moved various pieces, or created new methods that already worked originally, before this code review.

We identified several bad code smells during this assignment, including:

- Bad/confusing variable names
- Methods that were too long and could be refactored
- Classes that were too large
- Lack of documentation
- Poorly structured code
- Dead code
- Code duplication

To address each issue, we used various refactoring methods, such as:

- Extract Method
- Move Method
- Move Field
- Extract Class
- Rename Method
- Simplify Method

To summarise our code review, we identified prominent issues in the UI class and used several refactoring methods to make the class' purpose more clear and understandable. At the same time, we simplified the amount of data being processed in that singular class and enhanced readability and understandability for all developers. Afterward, refactoring work was done on many other classes, providing more documentation and renaming variables and methods to better match their purpose. Access modifiers were changed to positively impact clarity and reduce the amount of coupling within each class. Lastly, the code was reorganised to further increase readability.