Code Review of Escape the Bank

Group 5

By: Jad Alriyabi
Marco Lai

Date of Submission: 2022-04-11

Overview:

Initially, when working on the project with the entire team, we had many issues regarding scheduling and getting different people's solutions to the problems was quite tricky at first. Although by phase 3, our game was successful, and the code logic was easy to follow for all team members. Through further testing of our code, we realized that our game design consisted of having a game object superclass where all game components were initialized; This made the program modules have high coupling, which we then realized is not the best software design to implement. Furthermore, with constant code review of the game, we realized our code is free of bad coding practices such as confusing variables names, classes that do too much, dead code, and other bad coding practices, which we saw as a positive outcome of our project. This code review will showcase what we thought our program did well and where some edits were needed.

Classes Review:

Barriers/Bonus reward/Exit/Camera Class:

- Clean code, no code dumps or mismanagement of data
- Lack of documentation so some comments were added
- No refactoring was needed
- Applied buffered image object and sprite sheet to import image

BufferImageLoader Class:

- Very clean code, implemented declare the path to load the image from, applied try&catch to handle exceptions

GameObject Class:

- Clean code, no code dumps or mismanagement of data
- Lack of documentation so some comments were added
- No refactoring was needed
- Implemented Getter and Setter Methods

Code Review of Escape the Bank

Game Class:

- Some dead code was deleted.
- The render function was a bit too long so some edits were made
- Refactored proper game state variables name
- Overall has proper documentation, clean code and has good structure, good usage of data

Handler Class:

- Tick/Render loop through all game objects and updates
- add/remove functions
- We place the up down left right here to avoid other game objects except the player class sharing the unused functions.
- Overall has proper documentation, clean code and has good structure

Menu, MouseInput, Player, Punishment, RegularReward and Window Class:

- Clean code, no code dumps or mismanagement of data
- Lack of documentation so some comments were added
- No refactoring was needed

Sound Class:

- Proper Documentation, clean code, no code dumps or mismanagement of data
- Empty Exception in the ‘setFile’ method was implemented

SpriteSheet Class:

- Proper Documentation, clean code, no code dumps or mismanagement of data

Code Review Conclusion:

The Program did have some flaws when reviewing the code. We saw a need for some refactoring in the Program after deleting some dead code in our Game and GameObject Class, plus the addition of proper documentation in many of the Classes. Our investigation led us to believe that the Program was free of lousy programming structure such as bad programming structure, code duplication, unused variables, unnecessary use of unsafe or unsound constructs, methods that are too long and that could benefit from being refactored, classes that are too large and try to do too much, etc. Gameobject Class, the superclass for all objects in our Program, leads to high coupling in our Program in this section. However, the rest of the Program was relatively cohesive and was not dependent on other classes to be implemented. Finally, refactoring the Moving enemy tick method led to many unwanted errors and broke the game, so we decided that we would change the way the enemies in the game chased the player. This fix will be implemented in our final phase.