ASSIGNMENT 3

FALL 2022 CMPT276, GROUP 11

Cameron Lee
Jimmy Nishimwe
Eric Tran

1) All of the classes that subclass ‘Enemy’ have lines of code in their constructors related to instantiating their sprites and UI sprites that were repeated between all of them. So those repeated lines were pulled out and put into a method in the ‘Enemy’ class that each of the constructors now calls instead of them all having the same code.

2) Many classes across all parts had redundant methods that overrode their superclass’s methods, like update(), start() and delete(). Some classes make use of overriding those methods, but most just called super.update(), etc. in those overridden methods which had no purpose. All of those redundant method overrides were removed

3) The class ‘GameLogicDriver’ was a very big class that had lots of methods that basically tied together all of the game’s components and ran the game. This made it hard to go through the class and edit things, so we separated the storage and management of all the “entities” of the game into a new separate class.

4) The class ‘Map’ contained almost all of the code that was used for the pathfinder due to not being able to access the needed fields for the algorithm, so all of the pathfinding related code was moved back to its proper class and methods were added in ‘Map’ to allow pathfinder to access what it needs

5) After moving over the appropriate code to the ‘Pathfinder’ class, it was missing a lot of documentation, so that was filled in.

6) Also, one of the bad smells corrected in the Pathfinder class was long methods. For example, the method “getNextPosition” had some functionalities that could have been split into smaller methods. To remove this bad smell, wherever we needed to explain what the code was doing, we instead created a method named after the intention. For instance, we named the “tilemapConverter” the way it is because it refers to the code in the getNextPosition method that was used to convert the Dictionary tile map into an array List of nodes.

7) Moved point to screen transformation code from Camera class into the Renderer class. This code is lengthy because it describes the order of operation to transform a world point coordinate into a screen space coordinate. The Renderer class needs this code and the camera object to render the scene, but the code itself does not belong in the Camera because we don’t need every camera to have a copy of the point transformation code.

8) Abstracted Sprite class further to improve the usage outside of the game engine and inside the game world. Previously, in order to edit a Sprite, the user would have to access the Sprite Renderer of a Game Object then access the Sprite within, then if they wished to do more complex transformations they needed to access the Render Component within the Sprite class and manually set parameters such as texture coordinates. The first of the changes to this class is to start abstracting methods to help remove this low level such as adding a scale method to scale the Sprite size instead of having to manually change each texture coordinate.