Testing Deliverable

<u>ToggleUpgradeMenuTest()</u>

- This tests whether the user is able to access and toggle the upgrade menu in order to upgrade, so this test would indicate if the user is able to upgrade in the first place. Our implementation has the user click on the Wizard sprite to toggle the upgrade menu.
- This tests the GameController and the Wizard sprites as the game controller allows the user to perform actions on the Wizard Sprite while the sprite and the class it came from has the functionality to upgrade.
- Logically, the upgrade functionality should lie with what should be upgraded - the Wizard class itself, and the only class that should be calling that upgrade method should be the Game Controller where the player controls what gets upgraded.

<u>UpgradeGameplayChangeTest()</u>

- This will test if there is a change in “gameplay” after a tower is upgraded.
- This will test the components of the Wizard class as this logically should hold the functionality that upgrades a wizard.
- If the code functions like it should, the numerical stats of the wizards should change which should translate to changes in damage output.

<u>FinalBossVisualDistinctionTest</u>

- This will test if there is a visual change between the final boss and the other monsters which there should be as the final boss is a rectangular dragon while the other monsters are smaller squares.
- This will test the components of the Dragon Class and the Goblin, Golem, Kobald, and Orc classes for comparison. Each of these Enemy subclasses should hold information about their visual aspects.
- If the code functions like it should, the Dragon should look like a rectangle which lies in its sprite aspects which is a larger rectangle.

<u>FinalBossDifficultyTest</u>

- This will test if there is a great increase in difficulty in beating the final boss.
- This will test the components of the Dragon Class as the Dragon class holds the stats for how difficult it will be to beat.
- If the code functions correctly, then the Dragon enemy should have much more health and more speed than the other enemies which is stated in the enemy stats within the Dragon class.

<u>WinScreenActiviationTest</u>

- This will test if defeating the boss will activate the win screen.
- This will test the components of the Dragon Class, Main Class, WinScreen class, and GameController class. The Dragon and its health status determines if the win screen will be activated. The game controller has the logic that will determine if the game is won

through its “won” variable. The main class will take that win status and initialize the win screen, and the win screen holds the display of what will be shown as a result of winning.
- If the code functions as expected, then when the dragon loses all of its health, the game controller will notify the main class to activate the win screen.

<u>DisplayStatsTestWinTest()</u>
- This will test if stats will be displayed when the player wins.
- This will test the components from Main class, Game Model class, and WinScreen class/GameOverScreen class. Game Model holds the variables that represent the 3 statistics. The Main class will activate the initialization of the win screen which would present the statistics.
- If the code functions as expected, then when the player wins, the main class will initWinScreen() which will launch the Win Screen which will access and present the statistics.

<u>displayStatsLoseTest()</u>
- This will test if stats will be displayed when the player loses.
- This will test the components from Main class, Game Model class, and the GameOverScreen class. Game Model holds the variables that represent the 3 statistics. The Main class will activate the initialization of the win screen which would present the statistics.
- If the code functions as expected, then when the player loses, the main class will initGameOverScreen() which will launch the Game Over Screen which will access and present the statistics.

<u>gameWonRestartTest()</u>
- This will test if the user can restart from the win screen.
- This will test the components from the Win Screen as the win screen class holds the logic for actions when you click the buttons located on the win screen including restart and quitting.
- If the code works correctly, if the user clicks restart, the game should restart.