Software Testing Report

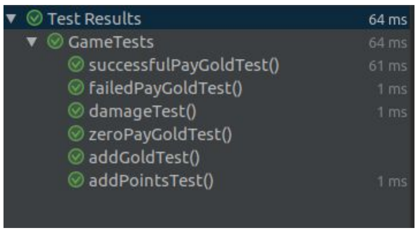
Testing Methods

We decided to use a variety of testing approaches to identify as wide a range of issues with our software as possible. To accomplish this we decided that we would use both white-box and black-box testing.
We used JUnit to perform some unit testing on some of the different methods in our code [1]. However, we found that JUnit was not compatible with libgdx (a Java library we used to handle the graphics of our game) by default, and we had to carefully construct our testing environment to not include anything which utilised libgdx. Specifically, we had to create a new Ship constructor which did not initialize any textures, as this was causing errors with our unit tests. This led to us not being able to test much of our game's code using JUnit, meaning we had to focus more on black box testing.

The Scrum framework utilised by the team relies on dividing development up into short increments which lead to white box testing (detailed below) of each section of the code as soon as it's finished, which is useful for ensuring confidence in every aspect of the code as well as being in line with our methodology.

Results

Unit Testing

Name	Description	P/F	Evidence
addGoldTest	Testing if addGold method in player class updates players gold variable correctly.	P	
successfulPayGold Test	Testing if method payGold(amt) returns true and changes players gold variables correctly, test ensures player has enough gold for upgrade.	P	
failedPayGoldTest	Testing if method payGold(amt) returns false if player can't afford the upgrade.	P	
zeroPayGoldTest	Testing if method payGold(amt) returns true if player's gold is equal to the cost. This tests an edge case.	P	
addPointsTest	Testing if method addPoints in player class updates players points correctly.	P	
damageTest	Generate a dummy ship, and simulate damage(amt) method. Test if method correctly changes Ship health.	P	

```
▼ ✓ Test Results                    64 ms
  ▼ ✓ GameTests                     64 ms
      ✓ successfulPayGoldTest()     61 ms
      ✓ failedPayGoldTest()          1 ms
      ✓ damageTest()                 1 ms
      ✓ zeroPayGoldTest()
      ✓ addGoldTest()
      ✓ addPointsTest()              1 ms
```

All of our unit tests passed. However due to the limitations of unit testing within libgdx, we can't claim that these tests are complete and correct as these tests are simulated without the use of a graphical end (we had to use a different Ship constructor which doesn't contain a texture file) however we have good reason to assume these methods are correct as these successful unit tests are backed up by play through testing of the game itself.

References

[1] SEPR “GameTests.java” Rear Admirals [Online] Available https://therandomnessguy.github.io/SEPR/Assessment/2/GameTests.java [Accessed: Jan. 20 2019]

[2] SEPR “Assessment 2 Testing Documentation” Rear Admirals [Online] Available https://therandomnessguy.github.io/SEPR/Assessment/2/TestingDocs.pdf [Accessed: Jan. 20 2019]