Testing methods

With our project being built using Unity, it only made sense to make use of Unity's built in test tools for testing our project wherever possible. Tests have been written in separate testing classes, related to classes used in the game, e.g. Unit and UnitTest classes. Unity handles testing within the editor, making it very easy for us to quickly test whether or not a feature is operating correctly at the click of a button, along with useful error messages should it not.

All of these Unity tests made use of assertion tests, where the actual outcome of the testing script was compared with the expected outcome. A test would pass if the two were the same. Should a test fail, we would go back and first inspect whether the test is operating correctly (the test was calling the correct methods at the correct time with the correct parameters). If the test was correct, we would then use any error messages to try resolve the issue and only pushing the new code to the master branch once we were satisfied everything was as it should be. Code that has passed testing is then refactored and tested again to make sure the new, cleaner version still works as it should. Tests were also refactored when necessary. All tests were written in a consistent style to keep the code easier to read and understand. We structured our tests names like *[UnitOfWork_StateUnderTest_ExpectedBehavior]* [1] to make it immediately obvious what the test was for and what was supposed to happen, streamlining our development.

Unit tests were written to test the back end aspects of the game. We had a list of core tasks that the game must perform to meet Assessment 2 deliverables which we followed when writing and testing the game as development progressed *(see Assessment 2 test designs).* By following this easy to read and understand checklist, it was clear what was being coded and tested. Each method found in the game scripts on the whole had at least one test to to ensure that part of the game functioned as expected, allowing us to gain a more complete understanding on how complete our project was. The actual tests can be found in the 'Assets, Unit Tests' within the Unity project (or on the website [link]) and must be run from within Unity's test runner.

We also needed to test the UI element of our game; these tests were carried out differently to the unit tests. Unity's testing features would have been ideal testing the UI, however many parts could not be tested in this way, due to them needing a human's opinion, to make sure the game is displaying information correctly e.g. information is displayed where it should be on the display. These tests were undertaken by hand and a similar table for tests was completed to keep a record on how the UI was progressing during development. The actual tests can be found in the 'GUI Test Runs' spreadsheet here: [link]

Testing report

Unit tests

After testing each method in each class for the game using Unity's built in testing tools, we found that all 43 unit tests passed (Fig. 1). Any tests that previously did not pass were corrected - either the test itself if it was not operating correctly or the method as development continued. We feel our unit tests were complete as we have tested each method from the game with at least one test, multiple in many cases. This was done to ensure methods act as expected normally and do not allow errors to occur which would compromise the game's integrity, resulting in a flawed product. Unit tests for Assessment 2 can be found at: [link]

Figure 1: A screenshot of the successful test results of all 43 unit tests in Unity's test runner

GUI tests

The GUI tests were conducted manually to confirm information was being displayed correctly, and all 6 GUI tests passed. Due to our approach of development and testing simultaneously, our GUI tests cover features that are currently implemented, but do not test for features that may be implemented in the future (i.e. a pause menu, a title screen before the game starts, etc.). While not as time-efficient as automated testing, manual testing is as reliable and more intuitive in the case of testing graphical systems like the GUI. The GUI tests for Assessment 2 can be found at: [link]

References:

[1] R. Osherove, "Naming standards for unit tests," Roy Osherove, 3 4 2005. [Online]. Available: http://osherove.com/blog/2005/4/3/naming-standards-for-unit-tests.html. [Accessed 9 12 2016].