Evaluation and Testing Report

Testing

We utilised the same test suite format detailed in Assessment 2. We felt this suite format was comprehensive for the project and made the link between implementation and testing explicitly clear.

Functional

Due to the small scale of the project Unit testing and integration testing have been listed together under the heading of unit testing. Some unit tests do require multiple parts of the system to run, but each test is designed to specifically test one single part of the system. As these integration test variants are quite small (two or three classes at most), debugging with other relevant unit tests should still be straightforward to discover the exact cause of any issues. Explicit integration testing was again deemed unnecessary due to the size and format of the system. In particular functionality of the system is not intended to exist separately so integration testing (outside of that required to ensure all functions had been tested) does not in fact provide a reasonable indicator of correctness in respect to the requirements.

We proceeded directly to system testing and performed these tests regularly. Whenever changes were made, we made it policy that changes should be deployed on the team member's machine in the context of the game. In addition to this, when faults were found we relied on system testing in debugging mode to isolate issues and ensure they were not impacting any other functionality. In our final stage of functional testing we acceptance tested both with the UI against the requirements ***link to test report*** and ensuring all other tests passed. This gauntlet of tests ensured that all tests passed and all requirements were met. Critical for this stage of development.

Our indication of appropriate quality for the code was met when:

- All tests passed
- All code was commented clearly
- All requirements were met

These indicators are absolute at this point of the project, as this is the final stage of the game development. There's no reason requirements shouldn't be met.

Non-Functional

Performance testing has been informally addressed throughout the project, only catching issues when they were the likes of memory leaks. To prevent issues arising the system has been deployed on the client's intended use-case computers throughout the testing. All GUI and playtesting has been performed on the intended use-case computers. Due to the low spec requirements of the game, this was not hard number testing. But ensuring timers behaved correctly and transitions were timely.

Compatibility throughout development, the majority of the team develops and tests on Windows, one member develops and tests on macOS and Travis CI deploys the tests in a linux environment. This has ensured full compatibility. However as we are using Java this was unlikely to be an issue provided the system being utilised has the correct version of the JVM.

Usability was qualitatively tested by fresh users at the end of the development cycle. Mainly to ensure the system was usable with only the materials provided to the client (the user manual).

Evaluation

To determine the correctness of our code in relation to the brief we went through our requirements and compared them to tests we ran by default.

Subjective requirements were to be decided on a vote based system, requiring 100% support of team members for a requirement to be considered accepted **results here.**

Alternatively, other requirements that could only be determined by end users (such as ease of use) were checked with the participants of the usability testing. For checks that failed, efforts were made to correct them as shown in the Usability Testing report (see *Usability* below).

Functional Test Report

Unit & Integration

See the results here
We ran 30 individual tests throughout the development cycle. Each corresponding to a function in the codebase. As mentioned above 100% pass rate was required for these tests. We reached this requirement. If it had not been reached we would have ensured all failing tests were written correctly and if they were we would have modified code to perform correctly.

System

Conducted implicitly with performance and Requirements acceptance testing. The system performed as expected and allowed all requirements to pass with the resources the client wished it to use.

Requirements Acceptance

We ran these tests as a qualitative extension of the system testing. This allowed us to explicitly state all requirements that hadn't been implicitly met from the initial Unit, Integration and System tests. The report can be found here. It was key that all requirements were met at this point. Any other situation would be classified a failure.

Non-Functional Test Report

Performance, running the game on the client's intended system works perfectly. Meaning, timers perform as expected and transitions do not hang. AI calculations and processing times are far below the cutoff listed in requirements.

Compatibility, as mentioned above the system is compatible with all modern operating systems. We tested macOS, Linux Ubuntu and Windows 10(The intended and required OS). The game ran as expected on all with no differences.

Usability

Usability was tested qualitatively; done in order to see how real users would react to the game, whether it was easy to understand and whether it was fun. The participants were given the user manual to read through and asked to play one whole game; they were then asked to make comments about the user manual and the game. Any issues that were raised, we attempted to correct. A second round of testing was then conducted to assure that the changes were sufficient. The results of the usability testing can be found here; the document contains a list of consequent changes and rejected changes at the bottom.

Changes to Assessment 3 testing

The previous test suite has been expanded to include the new requirements from Assessment 4. We have also returned to the format of our tests from Assessment 2. In execution, plan and result format.

Requirement meeting

Final Requirements
Requirement Acceptance Testing

We conducted qualitative requirements acceptance requiring 100% pass rate to conclude the tests as complete. We took each member of the team and asked them to play a full game (This was combined with usability testing). If they saw a requirement be fulfilled, they commented on it and marked it passed. This has produced a readable and clearly complete report. This maps clearly to the requirements and details how they are met within the game.

UI Requirement testing

We have tested acceptance of all requirements and since all of those requirements are met, we can state that requirements, related to UI are met as well (list of UI related requirements is here).