Engineering 1

Group Assessment 2

Software Testing Report

Cohort 2

Team 11

Testing Methods and Approaches

Our chosen testing methodology is a combination of dynamic testing and analysis using JUnit5 as a testing framework, and some static testing. As we have full access to the source code, this is the most appropriate testing methodology as it provides as much coverage as required, including for niche internal cases and cases only encountered during gameplay.

Using JUnit 5, we have created a number of unit tests for a majority of the game's classes and functions. These allow for simple testing, such as checking that a value can't exceed its maximum or minimum, as well as more complex testing such as two objects colliding which invokes multiple functions where all must pass in order for the test to succeed. This allows us to ensure that the quality of code is high across the whole codebase, and that adding a new feature won't break some core functionality by accident. Additionally, it allows us to detect errors in the code that may not be apparent in a quick gameplay test, but would become much more apparent given further testing.

As well as unit testing to test our algorithms and functionality, we also use gray-box testing wherein a developer or someone with some knowledge of the source code and internal workings of the game plays it to ensure that the actual gameplay experience is what we expect. This can pick up errors that unit testing cannot detect as whilst the algorithms may all execute properly, there may be a logical error somewhere in the code or some sort of numerical error that grows exponentially as it's used in the code. Furthermore, it allows us to tweak the gameplay to ensure it is as engaging and enjoyable as possible for the end user.

Test Results

Unit Tests

Below is a brief explanation of the purpose of each of the unit tests and their results when run including which parts passed and where there were errors.

<u>These tests relate to functions and mechanics of the game</u>

TestAI

This test has five components which each test different parts of the AI's capabilities. The tests cover the AI's ability to locate other AI boats involved in the race using ray casting, to locate obstacles on the race course using ray casting, to locate powerups on the race course using ray casting, that the AI boats can be hidden when set ho hidden and that the AI boat is set shown when set to shown. The results of these tests are listed below:

AI boat locates AI Boat with ray cast: Passed in 0.001 seconds
AI boat locates obstacle with ray cast: Passed in 0.002 seconds

AI boat locates powerup with ray cast: Passed in 0.002 seconds
AI boat is hidden when set to hidden: Passed in 0.001 seconds
AI boat is shown when set to shown: Passed in 0.017 seconds
All tests passed in 0.025 seconds.

TestBoat
This test has nine components to test that the player's boat is working correctly. The tests cover that the players boats durability cannot exceed a maximum value of 1.0 and cannot fall below a minimum value of 0.0, the boats stamina cannot exceed a maximum value of 1.0 and cannot fall below a minimum value of 0.0. The tests also tests the player's boats collision with other AI boats, obstacles on the track and the boats collision with the powerups available. The results of these tests are listed below:

Boat health has a maximum value of 1.0f: Passed in 0.003 seconds
Boat health has a minimum value of 0.0f: Passed in 0.002 seconds
Boat stamina has a maximum value of 1.0f: Passed in 0.002 seconds
Boat stamina has a minimum value of 0.0f: Passed in 0.001 seconds
Boat collides with other boats: Passed in 0.016 seconds
Boat collides with obstacles: Passed in 0.003 seconds
Boat collides with boost powerup: Passed in 0.001 seconds
Boat collides with stamina powerup: Passed in 0.002 seconds
Boat collides with repair powerup: Passed in 0.002 seconds
All tests passed in 0.033 seconds.

TestCollisionBounds
This test has three components to test the collision mechanic on items in the game. The three things tested in this unit test are that basic rectangles can collide, that collision of multiple rectangles together functions properly and that collision of rotated rectangles functions as intended. The results of the tests are listed below:

AABB collision of rects: Passed in 0.002 seconds
AABB collision of multiple rects per collider: Passed in 0.006 seconds
AABB collision of rotated collider: Passed in 0.023 seconds
All tests passed in 0.031 seconds.

TestDifficulty
This test has five components to test that the difficulty settings are functioning correctly. These tests are that the target speed of AI boats is set correctly on all three difficulties, that the obstacle count is set correctly on all three difficulties and that the powerup count is set correctly on all three difficulties. The other two tests check that the difficulty can not be incremented past hard difficulty and can not be decremented below easy difficulty. The results of the tests are listed below:

Target speed returns properly: Passed in 0.000 seconds
Obstacle count returns properly: Passed in 0.001 seconds
Power up count returns properly: Passed in 0.006 seconds
Difficulty will not increment past hard: Passed in 0.010 seconds
Difficulty will not decrement below easy: Passed in 0.001 seconds

All tests passed in 0.018 seconds.

TestGameObject
This test has two components to test that objects in the game can be hidden and shown when required. The test results are listed below:

GameObject hidden when requested: Passed in 0.015 seconds
GameObject shown when requested: Passed in 0.001 seconds
All tests passed in 0.017 seconds.

TestMain
This test simulates a headless libGDX environment so that textures can be made without actually starting the application which allows us to run tests on these objects without the game running.

TestMovableObject
This test has four components to test that the movement of objects in the game functions correctly. These tests check that objects cannot surpass their maximum speed by normal acceleration and that they cannot surpass their maximum speed by using code to manually increase the speed, it is also tested that an object will move in a straight line at the correct speed when using set values and that an object moves at an angle at the correct speed with set values and a rotation. The results of the tests are listed below:

Object cannot naturally accelerate faster than max speed: Passed in 0.001 seconds
Object cannot programmatically accelerate faster than max speed: Passed in 0.002 seconds
Object moves in a straight line at the right speed: Passed in 0.015 seconds
Object moves at an angle at the right speed: Passed in 0.001 seconds
All tests passed in 0.020 seconds.

These tests relate to the UI of the game

TestButton
This test has one component that checks that the button can detect when a mouse is hovering over it. The result of the test is listed below:

Button hover state works when mouse is over button: Passed in 0.015 seconds
Test passed in 0.015 seconds.

TestSwitch
This test has one component that checks that the switch can detect when a mouse is hovering over it. The result of this test is listed below:

Switch hover state works when mouse is over button: Passed in 0.014 seconds

Test passed in 0.014 seconds.

Test Completeness and Correctness

All of the unit tests for the game cover mechanics of the game that can be objectively tested with a known result beforehand. This was done as this is the best way to test that a function/component of the game is working as intended on a mechanical level whilst taking out any human error and keeping it as unbiased as possible. It is also impossible to test how the game feels as that requires a player to interact with the game and form an opinion based on their experience.

Test Report can be found on our website:
https://eng1-team-11.github.io/ENG1-Team-12/Documentation2/test/index.html

Gray-Box Testing

In addition to automated tests we performed a number of manual tests to further assess the fulfilment of requirements that can only be evaluated by a player, for example, NFR_SATISFACTION, as well as other aspects of the game such as aesthetic appeal of the game graphics and other UI related aspects like presence of the tutorial, info display etc. These features cannot be tested by unit testing, since there is no specific input or expected output that can confirm the correctness and realization of given requirements.

Manual testing results can be found on our website:
https://eng1-team-11.github.io/ENG1-Team-12/Documentation2/TestingMat.pdf