Testing

Due to the differences in the nature of this assessment and the previous one, our testing methods this time differed from the last assessment. Whereas the previous assessment had more of an emphasis on continuing the development of a partially complete game, and involved much refactoring and altering of the existing codebase, this assessment focused on extending the existing code with specific new functionality, following additional requirements.

For this assessment, we inherited Team Smew‘s code. This included their JUnit testing suite, which we found to be quite comprehensive. Due to the nature of this assessment the only changes we made were additive, as we were adding functionality to an existing, working codebase. As we didn’t alter any of the underlying code, we were able to reuse the previous team’s testing suite for our regression testing. Even though we didn’t alter any of the code, we carried out regression testing to make sure that our additions to other areas of the codebase didn’t interfere with the functionality of the existing code. This differed from assessment 3, where we had to write our own tests for regression testing, as we didn’t inherit any from the previous group.

Team Smew also used playtesting to test elements of the game that were too complex to test using unit testing. LibGDX gives obscure errors on some unit tests due to issues with the external libraries Smew used, so we also playtested these elements instead of unit testing. We retested all of their playtests as well as playing several full runs through the game, to ensure that any major bugs would be encountered. Any bugs discovered were rectified and the appropriate unit was retested.

Unfortunately, we were unable to reuse any of our tests from the previous assessment because we were working on a different game during that assessment. We used both unit testing and playtesting because these were the most efficient and logical methods for testing their respective functionalities, and also to maintain consistency with the previous group’s testing methods.

When writing tests, we first went through the new requirements and wrote tests to ensure that the requirements were satisfied. These tests were mainly done via playtesting, as the requirements are quite far abstracted from the actual implementation, and appropriate unit tests would have been very cumbersome and overly complex. We then moved on to more implementation specific tests. These tests ensured the functionality of the code supporting the actual implementation of the requirements.

Testing Report

Below follows a report of our testing for Assessment 4:

Unit Tests:

We inherited Smew's unit tests, which we feel comprehensively covered the elements we wanted to unit test. This is regression testing, as we did not alter any of this code, but wanted to ensure that our new additions didn't break any of the old code. The results of the testing can be found here:

https://docs.google.com/spreadsheets/d/1nQSEIE9ZNLqVrscaI6OQb_h1gb_05XGFxdjt4-fLh0E/pubhtml

The code for the tests can be found on our Github page:

https://github.com/teal-duck/mallard/tree/master/core/test/com/superduckinvaders/game

Although none of the test we ran failed, 11 of the tests were skipped. They were skipped because LibGDX, one of the libraries we used, gave strange errors under the test conditions that do not occur under real conditions. We believe this is due to Box2D, another library used in the project, requiring a game window to function, and some of the LibGDX components were dependant on Box2D. Extensive playtesting reveals no issues in any of the areas we had to skip unit testing for.

System Tests:

Smew extensively used playtests to ensure their system met all requirements and to ensure correct functionality of gameplay. We reused their system tests as part of our regression testing, to ensure we had not broken any functionality with our new additions. The results of their playtests can be found here, along with evidence for them:

https://docs.google.com/spreadsheets/d/1hRdf2yZJAAcsbRJbNXJtGofGPcVN7H8m9L_nrYjqISk/pubhtml

As displayed in the document, all the tests were passed, and no new further tests needed to be written due to how comprehensive these were, and because we didn't change any of the core functionality that these test.

Testing new Assessment 4 Requirements:

The following tests are all new tests to ensure the new functionality we added meets the requirements, and works as intended:

Testing Implementation of Requirements for Assessment 4

G14.1: “Demented waterfowl behaviours are picked at random from:

- Stand still
- Walk north
- Attack nearest enemy
- Run away from player”

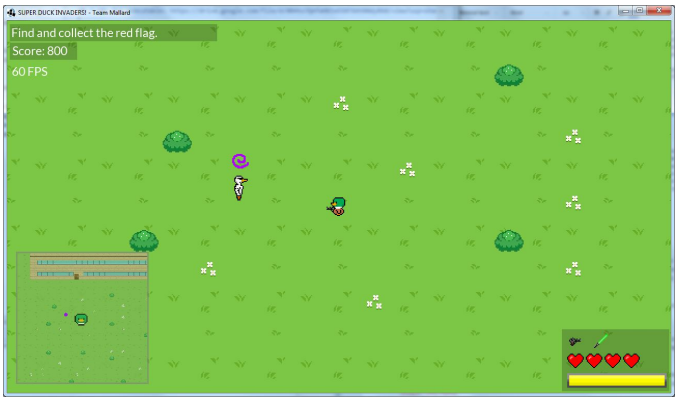
Description: Tests if the demented waterfowl behave correctly from the multiple behaviour options.

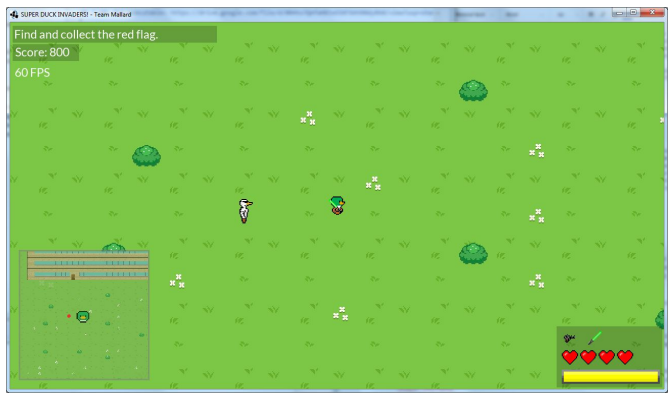
Test: Observe multiple demented waterfowl and how they behave in-game. Check for all four possible behaviours.

Status: PASS

G14.2: “Demented waterfowl become cured after 20 seconds of being demented”

Description: Tests that after a 20 second period, the demented waterfowl is no longer demented.



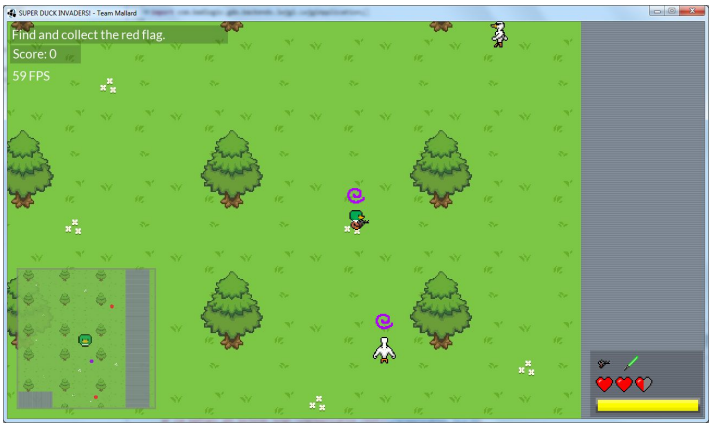


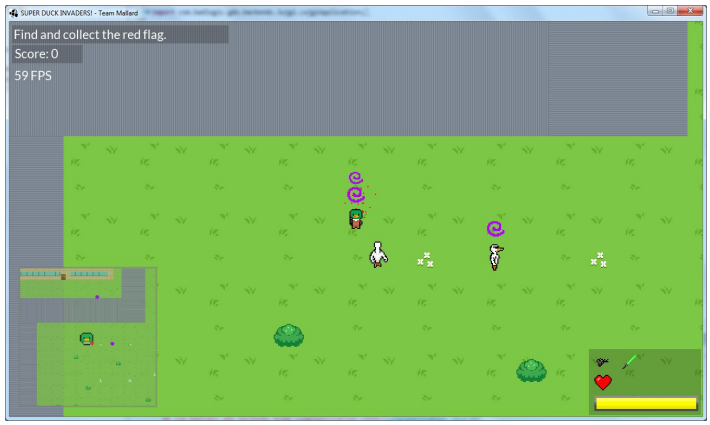
Test: In-game, observe a demented waterfowl and time how long it is demented for.

Status: PASS

C6: “Demented Player mode: continually and after a period of time the player will contradict a user input when this mode is activated”.

Description: Tests that when the player comes in contact with a demented goose, the player becomes demented (i.e. randomly inverts walk controls).



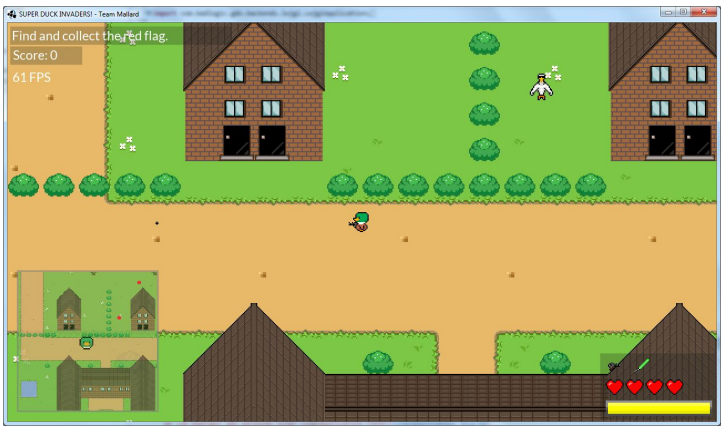


Test: Guide the player to contact a demented goose and walk around to see if the player’s moment changes randomly.

Status: PASS

G15.1: “Rapider fire: This cheat will make the standard fire rate as fast as rapid fire, and rapid fire’s fire rate faster”

Description: Tests that when the player enables the “Rapider Fire” cheat, the rate of fire of the weapon increases for the player.

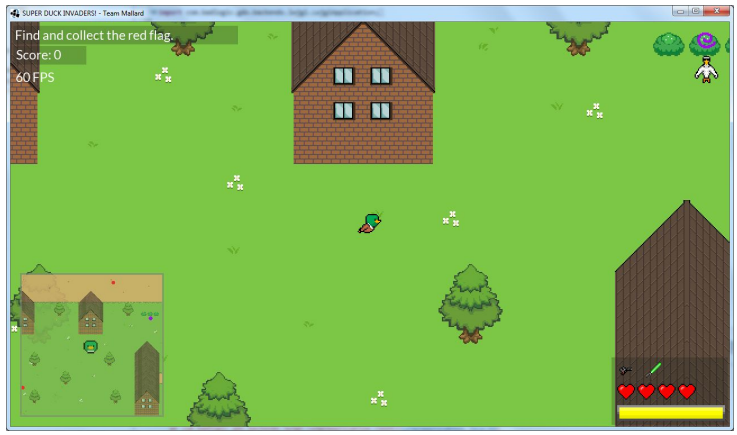


Test: Enable “Rapider Fire” through the cheat menu and compare the different rates of fire before and after enabling the cheat.

Status: PASS

G15.2: “Infinite flight: This cheat will make the player able to fly for an unlimited amount of time”

Description: Tests that when the player enables the “Infinite Flight” cheat, the player will not run out of their flight ability.

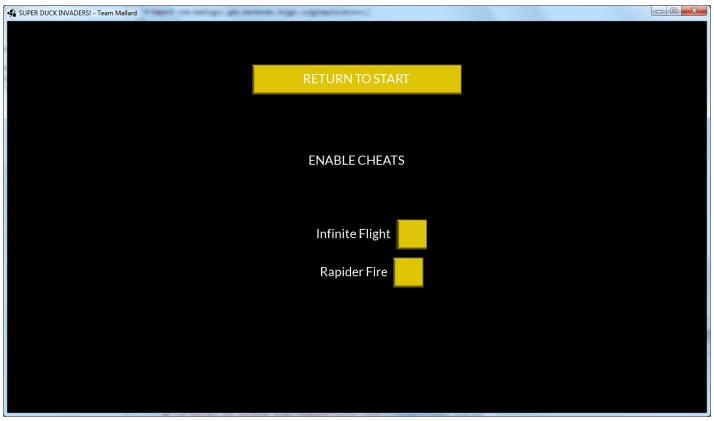


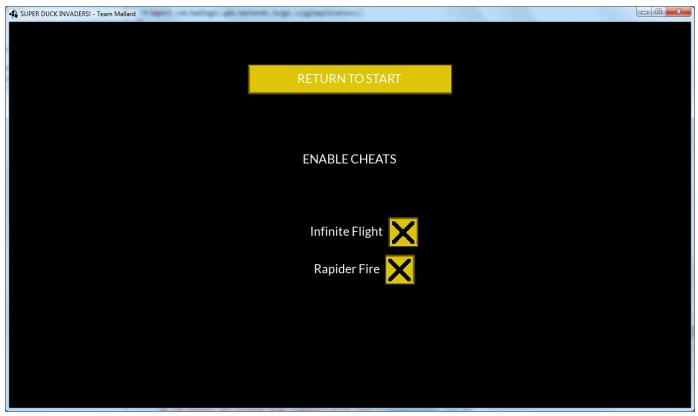
Test: Enable “Infinite Flight” through the cheat menu and use the flight in-game to see if it depletes.

Status: PASS

G15.3: “Cheats should be toggled on and off through a cheats screen”

Description: Tests that when the player is in the cheats menu, the cheats can visibly be toggled on and off and their states correspond to the gameplay.





Test: Go to the cheats menu and toggle the two cheats on and off and test them in-game.

Status: PASS