ENG1 - Assessment 2

Software Testing Report

Test2.pdf

Group 4

Mikaella Loppnow	ml2708
Tom Daly	td1026
Ethan Buss	eb2225
Dillon Pandya	dp1195
Ereife Odusi	ed781
Harriet Kirby	hk1114

Testing Methods and Approach

The purpose of the tests is to show our code's functional correctness, this is to complement our functional completeness as shown by our requirements. As such, we have built our tests to encompass our requirements.
By employing testing methods, we hope to prevent brittleness in our code and ensure clarity for ease of development and updates as our game progresses.
We decided to utilise a DAMP methodology (Descriptive And Meaningful Phrases) for the style of our tests - i.e, the tests should easily make sense to anyone when reading them.
Automated testing in IntelliJ.

- Automated testing is a key component of our testing because there are a lot of different scenarios that can happen in the game, and some rather complicated logic that makes it difficult to test manually (e.g., checking if the score is calculated correctly).
- It also generates a coverage report which is useful for quickly checking if our code is sufficiently covered by our tests.

Manual tests (for tests that are not suited to automated testing).

- Certain aspects of the game cannot be tested through automated testing - or cannot reasonably be tested through automated testing, such as the majority of the Screen classes, which serve to control the flow of the game. Simulating these would effectively require a full simulation of the game so instead these are tested by hand.
- Furthermore, many of these aspects are things that require human opinion to determine, rather than a precise examination of functionality; i.e., whether or not the character moves in a way that feels natural or if the map feels intuitive to navigate. Alternatively, end-to-end testing to evaluate an entire execution of the game does not warrant automated testing since there are too many combinations to cover, and the code warranted for this level of simulation would be extensive and excessive. It would also not be intuitive to understand, thus it would go against the DAMP methodology we are employing.
- Such cases are instead performed through manual testing, though we have attempted to minimise the need for this as it is time consuming in its own right.

Code inspection for certain classes:

- We have a significant number of classes that are not really suited to unit testing in their own regard: classes such as our constants, consisting of both container classes and enumerators, and that do not contain any methods of their own.
- As such, these classes do not benefit from being individually targeted for unit testing - thus, they are tested by manual code inspection to ensure that the constants are correct for the functionality that is required by our game.
- Furthermore, these classes are tested indirectly through the use of automated testing of the classes that utilise them - for example, ActivityType is tested through the testing of the GameState class.

The coverage report generated is not indicative of the actual coverage of our code since there are a lot of auto-generated methods (e.g., EndScreen.hide()) that do not require testing. Additionally, some large classes are not unit tested, and are instead end-to-end tested due to the sheer volume of classes they encompass, which are tested individually.

Testing Report

The testing section of our project was designated entirely to Harriet, and thus all the following tests were written by, executed by and analysed by her. Tests were carried out throughout the project, most significantly they were all carried out again prior to the declaration of completion and subsequent submission of the project.

Automated tests were carried out through IntelliJ JUnit testing, and manual tests were carried out by running the game.

Automated Tests:

Test	**Part of Project**	**Property**	**Description**	**Result**
AnimationComponentTest	Components	Correctness	Create a new animation component and ensure it is made correctly.	Pass (1/1)
CounterComponentTest	Components	Correctness	Create a new counter component and ensure it is made correctly.	Pass (1/1)
FixtureComponentTest	Components	Correctness	Create a new fixture component and ensure it is made correctly.	Pass (1/1)
HitboxComponentTest	Components	Correctness	Create a new hitbox component and ensure it is made correctly.	Pass (1/1)
InteractionComponentTest	Components	Correctness	Create a new interaction component and ensure it is made correctly.	Pass (1/1)
PlayerComponentTest	Components	Correctness	Create a new player component and ensure it is made correctly.	Pass (1/1)
PositionComponentTest	Components	Correctness	Create a new position component and ensure it is made correctly.	Pass (1/1)
TextureComponentTest	Components	Correctness	- Create a new texture component and ensure it is made correctly. - Test that texture is set to be invisible when hide is called. - Test that texture is set to be visible when show is called.	Pass (3/3)
ActivityTypeTest	Constants	Correctness	- Ensure that all intended activity types are in the enumerator ActivityType - Ensure that each activity type contains the correct activities.	Pass (2/2)
PlayerConstantsTest	Constants	Correctness	Ensure the starting location for the player is correct.	Pass (1/1)
GameStateTest	Models	Correctness	- Advance the day and ensure that the previous day is added to the list of days and that energy and hours are reset.	Pass (4/4)

			- Attempt to perform an activity and ensure that it is only performed if there are sufficient resources; and that resources are not consumed if there are insufficient to perform the activity. - Check that the activity counters are updated correctly. - Simulate the performance of the same activity every day and check that the corresponding streak appears and that the other streaks do not.	
LeaderboardTest	Models	Correctness	- Create a new Entry instance and check that it is created correctly. - Create 10 new entries and insert them into the leaderboard, then test that a new score that is: A, higher, is added to the leaderboard; B, lower, is not added to the leaderboard. Test that the lowest entry is removed and the leaderboard never stores more than 10 items.	Pass (2/2)
PhysicsPolygonTest	Models	Correctness	Create a new instance of a physics polygon and check all attributes are created correctly: - Name - Type - Position - Vertices	Pass (4/4)
EndScreenTest	Screens	Correctness	- Check that the score for a single day is calculated correctly. - Check that the score for an entire playthrough is calculated correctly.	Pass (2/2)

Manual Test Charter:

Test Name	**Actions**	**Expected**	**Testing**	**Results**
Start Game	Begin the program, click the “Start Game” button.	Game begins from day 1 (Monday). Start button should be obvious.	Correctness, usability	**Pass:** As expected.
Quit Game	Begin program, click “Quit” button.	Program closes as normal. Quit button should be obvious.	Correctness, usability	**Pass:** As expected.
Restart Game	Finish the game and click the “Main Menu Button” then the “Start Game”	Sleeping on Sunday leads to the score screen - clicking on the Main Menu button then opens the main menu, click start game and game begins again. How to restart the	Correctness.	**Pass:** As expected.

	button.	game should be obvious.		
Play With Name	Type in a name and then start and finish the game.	Name should appear on the final screen alongside the score. The ability to enter a name should be obvious.	Correctness, usability	**Pass:** As expected.
4-Direction Move	Move using WASD and check that the character moves the correct direction for one key pressed.	W: UP A: LEFT S: DOWN D: RIGHT	Correctness, usability	**Pass:** As expected.
8-Direction Move	Move in 8 directions using WASD combinations.	Should move in the correct direction (combination of keys). Opposite key combinations (A&D and W&S) should cause no movement.	Correctness, usability	**Pass:** As expected.
Collision Test Internal	Walk into buildings and see if it is possible to pass through them.	Buildings (and other objects such as the lake) should prevent the character from moving onto / into them.	Correctness	**Pass:** As expected.
Collision Test Edges	Walk all 4 edges (and corners) as the map and check if the character can go off screen.	The character should be stopped when walking into the edge of the map, in the same way they would be stopped when walking into a building.	Correctness	**Pass:** As expected.
Interaction Test	Walk up to each interactable location so that the prompt appears then press the E button.	The loading bar for the corresponding action type should appear, once it is complete then the energy, time and counters should all update correctly. On sleeping, the time should reset to 07:00 and the daily counters to zero.	Correctness, usability	**Pass:** As expected
Animation Experience	Walk around the map	The character animation should feel natural.	Usability	**Pass**: As expected
Movement Experience	Move around using WASD	Movement experience should feel smooth and responsive.	Usability	**Pass:** As expected
Playthrough Test	Play through the full game, simulating the full user experience.	Game should be intuitive and should not crash or fail to respond.	Usability, correctness	**Pass:** As expected.

Overall, our code passed all of the tests we designated to it. The tests cover the full material of our code, as the manual tests cover all of the aspects we were unable to test through the automated testing. I believe our testing to be both complete and an accurate depiction of the reliability of our code.

Testing Documents' URLs:

- Manual Tests Document: https://eb2225.github.io/theemd-site2/subpages/documents/Manual-Tests.pdf
- Coverage Report: https://eb2225.github.io/theemd-site2/subpages/documents/coverage.html