Testing

Hannah Thompson
Kyla Kirilov
Ben Hayter-Dalgliesh
Matthew Graham
Callum MacDonald
Chak Chiu Tsang

1.1.1 Testing Methods and Approaches

Our testing approach ensures comprehensive coverage of all project requirements through automatic testing methods, which include:

- Unit Testing: Every unit of code is tested independently.
- Continuous Integration: Tests are triggered automatically whenever code is pushed to GitHub, utilising GitHub Actions.
- Reporting: Test results and coverage reports are generated and saved systematically for review.
- Assertions: Used wherever possible to validate expected outcomes.
- Interaction Testing: Applied when interaction with the LibGDX library is necessary and assertions are insufficient. For instance, creating an energy bar on the screen is verified through interaction testing.

These methods are appropriate because by focusing on assertions and minimising interaction testing, we avoid issues such as false positives in method interactions that do not perform the intended actions (e.g., failing to change a variable as expected).

Our approach to testing involves creating an instance of the class to be tested and using mocking sparingly and where necessary. Mocking is applied for interaction testing, handling errors caused by headless environments such as NullPointerExceptions and shader compilation errors. When facing errors with LibGDX classes, mocks are created and injected into the class. If mocking is insufficient, we stub the method outputs of mocks to bypass issues.This approach effectively uses test doubles to allow for extensive testing capabilities while minimising the risk of classes passing tests when they shouldn't. By using minimal test doubles, we reduce false positives, ensuring that the code is genuinely functional.

1.1.2 Approaches for Choosing Input

Boundary testing was used to ensure that the software behaves correctly under extreme or unexpected conditions:

- For Animation - getFrame(), ensure the returned texture region has the correct width, and other elements remain unchanged.
- For LightCycle - render() method ensures proper rendering under various conditions.

Although it can often result in heavy test duplication, it was used sparingly to prevent errors in these cases.

Input space partitioning was used for GameClock - update() utilises boundary testing result in heavy test duplication.

Manual testing is reserved for classes that are not directly related to requirements, such as those responsible for creating minigames. This ensures that essential functionalities are automatically tested while non-critical components are verified through manual inspection, maintaining overall project quality without unnecessary automation overhead.

The combination of automatic and manual testing methods ensures comprehensive coverage and reliability. Automated tests provide consistent and repeatable validation of critical functionalities, while manual tests offer flexibility for less critical components this approach ensures robust and reliable software performance, addressing potential issues early and verifying that all requirements are met.

2.1 Report on Tests

The test environment for this project is configured using JUnit as the primary testing framework, with Mockito employed for creating mock objects to facilitate unit testing. The tests are executed using the GdxTestRunner, which is specifically designed to handle LibGDX game development environments. This setup ensures that the various components of the game are tested efficiently and effectively, enabling robust validation of functionality and performance.

87 automated tests (19 Test Class) were created and no tests failed. These tests have 61% instructions coverage in Utils packages and 75% instructions coverage in Objects packages. They also cover 84% instructions in Game Screen. Most of the classes in Objects and Utils are tested automatically. The remaining classes in these two packages are related to minigames and they are not directly related to requirements stated in the requirements document. Manual testing is performed to ensure all minigames work as expected.

On average, these tests cover 60% of possible branches. We ensure important classes such as GameClock and Collision Detector have at least 90% branch coverage to ensure they work as expected in different scnarios. Some tests have a low branch coverage as these branches were created to handle different users keyboard and mouse inputs. We were unable to simulate keyboard and mouse inputs in the test environment so these instructions/branches are not covered by our test. However, most of the branches related to numerical value are tested. In some cases, we choose inputs at the boundary of a branch for testing to further ensure no error will occur.

Manual testing was performed on every Screens in the Screen packages, including GameScreen which is covered by automated testing. Manual testing was chosen for Screens as Screens are often updated and it is difficult to keep interaction testing of Screen up to date. GameScreen is also tested manually as it’s a complex class and automated testing can not cover everything.

All requirements have at least one automated test, which is shown in the traceability matrix on the next page. This kept development on track and ensured that the requirements were being met by our game. A traceability matrix was used in our project for:

1. Verifying that the requirements had been met - gave us an overview of whether the requirements had been met, and where the gaps were.
2. Tracking progress visually - we were able to see which requirements were yet to be met and use this to plan the next tests to be carried out.
3. Change management - as the requirements changed, we could update the matrix easily to add the new developments.
4. Stakeholder communication - a clear and organised way to present the outcome of our test suite to 3rd parties, including the project stakeholders.

2.2 Traceability Matrix

	UR_RUNTIME	UR_MOVEMENT	UR_INTERACT	UR_INTERFACE	UR_ENERGY	UR_REST	UR_MAP	UR_RECREATION	UR_TUTORIAL	UR_STUDY	UR_EAT	UR_DAYS	UR_GAME_END	UR_ACHIEVEMENTS	UR_SCORE_LEADERBOARD	UR_SOUND_EFFECTS	UR_GRAPHICS	UR_SCORE	UR_INTRO	UR_CLOCK	UR_DAY_NIGHT_CYCLE	UR_ANIMATIONS	UR_READABILITY	UR_DIFFICULTY	UR_DAILY_DIFFICULTY	UR_MINIGAMES
AchievementHandlerTest														x												
ActivityImageTest																	x				x	x	x			x
AnimationTest																						x				
BuildingTest							x										x						x			
CollisionDetectorTest		x	x			x	x	x		x	x															
EnergyBarTest				x	x												x						x			
EventManagerTest			x		x													x		x						
EventTest			x														x									x
GameClockTest	x											x	x							x						
GameMusicTest																x										
GameScreenTest		x	x	x													x									
LeaderboardTest															x											
LightCycleTest																	x				x				x	
NameTextField				x															x				x			
PlayerControllerTest		x	x			x	x	x		x	x													x		
ResourceManagerTest																x	x					x				
ViewportAdapterTest				x													x						x			
AchievementDisplayTest									x				x	x												
UiElementTest				x	x															x						

	FR_START_GAME	FR_LEADERBOARD	FR_ENTER_PLAYER_NAME	FR_AVATAR_MOVEMENT	FR_WORLD_MAP	FR_ENERGY_BAR	FR_INTERACTION	FR_HIGHSCORE_TABLE	FR_EATING	FR_RECREATION	FR_RESTING	FR_STUDYING	FR_SOUND_EFFECTS	FR_SCORE_COUNTER	FR_ACHIEVEMENT_TRACKER	FR_GAME_ENDING	FR_SCALABLE	NFR_JAVA_VERSION	NFR_TIME	NFR_UPTIME	NFR_COMPATIBILITY	NFR_PERFORMANCE	NFR_COMPLIANCE	NFR_EFFICIENCY	NFR_SIZE	NFR_BUILD_TIME	NFR_SIMPLICITY	NFR_FUN	NFR_DIFFERENT_STUDY	NFR_ENERGY_BAR
AchievementHandlerTest															x							x	x							
ActivityImageTest				x			x															x			x					
AnimationTest				x																		x		x						
BuildingTest					x		x																				x	x		
CollisionDetectorTest				x			x															x		x						
EnergyBarTest						x																								x
EventManagerTest						x	x		x	x	x	x		x															x	x
EventTest							x																x				x			
GameClockTest							x									x			x			x								
GameMusicTest													x											x				x		
GameScreenTest	x				x															x		x								
LeaderboardTest		x						x						x	x		x											x		
LightCycleTest					x																x						x			
NameTextFieldTest			x				x																				x			
PlayerControllerTest				x			x															x								
ResourceManagerTest																	x							x	x					
ViewportAdapterTest	x						x										x				x	x		x	x					
AchievementDisplayTest															x															
UiElementTest						x								x	x		x													x