Module Code
COM00008I

BSc, BEng and MEng Degree Examinations 2019–20

DEPARTMENT OF COMPUTER SCIENCE

Software Engineering Project (SEPR)

Open Group Assessment

Module	Software Engineering Project (SEPR)
Year	2019/20
Assessment	2
Team	The Dicy Cat
Members	Michele Imbriani Daniel Yates Luke Taylor Isaac Albiston Martha Cartwright Riju De Sean Corrigan
Deliverable	Testing Methods and Approaches

TESTING JUSTIFICATION

Traceability Matrix: https://drive.google.com/file/d/1hVEtNUmXRG6uwWm0Vq5bAwHViH_d-t31/view

The team acknowledges the failure in generating robust unit testing material to support the game's architecture. The reasons behind this are multiple: miscalculation of time and effort allocation in primis, followed by an insufficient preventive research of the necessary tools and technologies needed for developing such tests. The Project Manager takes full ownership of the responsibilities of the former, while sharing with the Tech Lead the liabilities of the latter. This is something which no team member takes lightly, and everyone will reflect on this experience, treasuring it as a lesson for the incoming assessments.

Despite a thorough understanding of JUnit tests and the differences and applications of white- and black-box testing from every member in the team, what refrained us from achieving the results expected was the presence of too many dependencies between classes and methods which could not be broken with external tools and libraries such as Mockito. For example, creating a new instance of the game Kroy would require the instantiation of a new MenuScreen and GameScreen, which would return return a NullPointerException that traces back to the SpriteBatch of the instantiated screen being null, therefore making it impossible to test any method or function.

Nonetheless, we employed multiple methods to test our game, we took wide reaching methods to testing including user play-testing and debug features. We also decided, for the sake of consistency throughout the assessments, to stick with the ISO/IEC/IEEE 29119 standards [1][2] for software testing design report and traceability matrix (URL). A debug feature was implemented, which can be controlled directly from the main menu, by setting the 'showDebug' variable to true, which draws on the screen various important aspects of the player's view, such as the hitboxes. This allowed both us and any further developers to ensure that these usually unseen objects are aligned, located and are moving correctly. It was also created as a separate class in order to facilitate the addition and remotion of further elements to draw.

Finally, we also used play-testing. While this is not the most robust testing method, it allowed us to emulate behaviour similar to our players. This showed us issues we had not expected, nor had been able to test for - such as an infinitely scrolling map past the boundaries, and sprites overlapping other sprites that should be further in the foreground. This was particularly important since we were using external libraries which we couldn't unit test ourselves - by using this playtesting we were able to recognise some of the quirks of how our code worked with these libraries.

In retrospect, we believe we could have begun the testing process earlier in the development cycle than we did. While we had a rigid architecture set out, it would have allowed us to cement that architecture, as well as resolve certain issues much quicker than we did, and also allow us to control the amount of time we spent reworking sections of the code.

References
[1] Software Testing Standard Website [Online] Available: http://softwaretestingstandard.org/
[2] Summary of IEEE Software Testing Standard [Online] Available http://www.cs.otago.ac.nz/cosc345/lecs/lec22/testplan.htm