Testing Report

Tom Rosewell
Sam Redgewell
Tom Keeler
Jake Stogden
John Limb
Adham Noureldin

Description of Testing Plan and Methods

For our testing we decided to have 3 phases of testing. They would be split up into 1 initial test and 2 tests of our updated implementation.
This initial testing phase would allow us to collect information on what needed to be changed about the game, alongside the unimplemented requirements UR_COMBAT and FR_ENEMY. These would become new requirements labelled with a priority rating. Once this was implemented we could move onto phase 2. Phase 1 will be conducted by just inspection, as it is the fastest and simplest testing method, and would make any glaring issues or shortcomings apparent before development begins. We decided against any more complex testing methods both due to the short timeframe, and the fact any bugs that slip through can be picked up in later testing stages.
Phase 2 of testing would be the testing of our game that includes the new requirements iteratively. This would occur after the implementation. This would allow us to find bugs in our code and potentially overlooked changes to the gameplay. This phase was partially implemented by the team adding new code, by checking that the code ran before passing it over to the testing team, who performed a more in depth search for bugs, which would be added to the issues section of the GitHub repository and noted in the table below. Afterwards we would have a final set of changes to make to the game, leading us to phase 3.
In phase 3, multiple testing sessions would occur. It would be an iterative process where the final bugs and other small problems are coded out of the game and no new requirements we made. Iterative testing would be needed as the code would be updated alongside the testing, as this would be the most efficient way time wise.

The actual testing methods for each 3 phases will be for the most part exactly the same. We would apply both white box and black box testing (in that order). White box testing would allow our programmers to remove predictable bugs before they were discovered in black box testing, however only black box testing will be implemented in the first phase as we expect most pre-existing bugs to either be having a noticeable effect on gameplay, or to be discovered while implementing new features. This would be very useful for saving us time as time would not be spent on debugging the problem. For similar reasons the testing team decided to base some of their tests off a repository for Java Unit testing**[1]**, discovered by one of our testers. The 3 phase plan seemed optimal as it logically tackles the problem of already existing bugs, and bugs we will inevitably create ourselves. Removing the initial bugs first is a more suitable decision as if we implemented more features beforehand we may create co-dependent bugs that are harder to remove. Each problem will be given a priority of ‘shall’, ‘should’ and ‘may’ which indicate how urgent the problem is.
The testing we agreed to perform is alpha testing, but despite its benefits, beta testing will not be carried out. While beta testing would allow the discovery of more obscure bugs, it would require a team of players who were from our target audience and also uninvolved in the development of the product. To do this, participants would need an incentive to do this work for us, and considering our limited time scale and lack of budget, beta testing was concluded to be outside the scope of the development.

Testing for all phases was mostly carried out using manual testing methods due to the limitations of the LibGDX Headless functionality not allowing for spoofing of certain elements of the GUI within GameScreen E.G. LibGDX Shaders. Leading us to mostly Black box testing in this phase. All tests are recorded in the testing document Including the automated.

Initial Testing - Phase 1

Problem ID	Description	Priority	Final Build status
P1	The window of the running game is possible to scale, but assets inside the window are affected.	Should	Pass
P2	The game camera can allow you to see outside the map.	Should	Pass
P3	Being too close to the map edge will mean you don't get attacked by other colleges.	Should	Pass
P4	Ship controls can be frustrating (there is a delay of turning).	May	Pass
P5	Players are not informed of all controls (namely 1 and 2 being zoom, C locking the camera and B switching camera targets, and X being a reset).	May	Fail
P6	Enemy ships will not follow the player perfectly, and will go 360 degrees in the other direction at a certain angle.	May	Pass
P7	Options for music and sounds do not work properly, not turning off with the options.	May	Fail
P8	Enemy ships hitbox is not very accurate, and could do with shrinking down to the sprite used	May	Pass
P9	The player's spawn location is not always in the very centre, and with a changed window size a player can spawn right next to a college (or inside one).	May	Fail

Secondary Testing - Phase 2

Problem ID	Testing type	Description	Priority	Status
P7	White Box	The audio control problems stem from confusing object management.	Should	Incomplete
P11	Unit Tests	Testing Framework implemented	Should	Complete

Tertiary Testing - Phase 3

Included here is only tests that failed and their IDs or were otherwise informative to the game design in the document below the table is included an explanation for the tests, A complete list of tests in this phase can be found at (Link) as well as a list of bugs that could not be

written out that could not be picked up by traditional scripted testing, but are noticeable from playing the game

Problem ID	Testing type	ID	Priority	Status
P12	Black Box	14	May	Incomplete
P13	Black Box	60	May	Incomplete
P14	Black Box	133	May	Incomplete
P15	Black Box	134	May	Incomplete
P16	Black Box	156	May	Incomplete
P17	Black Box	179	Should	Incomplete
P18	Black Box	180	Should	Incomplete
P19	Black Box	181	Should	Incomplete

Format for the testing document is as follows

Category	Test Name	Reference	Pass condition	Fail / Pass	Explanation

P12

Player Ship	Alternate fire	Ship	Creates cannonballs	Fail	Cannonballs being created but not moving out from ship

Alternate fire should send cannonballs shooting in 4 directions from the ship however the cannonballs seem to collide with the player ships hitbox causing the explosion animation. We couldn't find a solution for this problem as it is not consistent and the method of creating the cannonballs doesn't allow us to displace them enough to fix the bug.

P13

PowerUp	Shield Second Usage	Ship	Shield Works on second Usage	Fail	Bool ShieldStatus not set to false when it is destroyed

Shield status Is not updated, so the shield could only be used once This bug was not noticed until very late into development and as such did not get fixed by the coding team, however with looser time constraint could have been fixed by adding another query to the update

method that removes the shield if it hit the time limit of it existing, this would also act as an artificial cooldown to its usage.

P14

Music	Music Volume	SoundController	Music changes volume when controlled by Music volume	Fail	Music Volume does nothing to volume

The separate music controller slider cannot control the volume of music, this is due to the sound controller not separating them as concepts, we viewed this as a low priority issue as the master controller would still allow for changing of the volume, and the fixing of this would have taken some time, this was a bug that was inherited from the initial build and would have taken an entire overhaul of the soundController class.

P15

Music	Music Disable	SoundController	Music doesn't play when music isn't togged on	Fail	Music toggle doesn't turn music off

Similar to P14 to Fix the music toggle would require an overhaul of the soundController and separation of the sounds, as the toggle button effectively just sets volume to zero for the music, this was left as low priority and with more time would have got attention.

P16

SFX	Button Press Back	SoundController	Back button makes Sound	Fail	No back buttons make the SFX

This Bug was regarded as very low priority as it was more of a quality of life feature for the user, however the back button in all of its iterations would not play the associated sound, this could be fixed easily but also wasn't noticed until later into the development cycle and as such was put at the bottom of the stack of jobs as more important features needed to be fixed.

P17

Save System	Save Game Save Second attempt	GameScreen	Game is saved again	Fail	Button Cannot be pressed

This bug was built into the method that built would save the game, and we as a team are not aware of what is causing it, we believe it is to do with the nature of screens within the program however we are not sure, to remedy this we would have spent more time on our screen architecture as well as more time learning how to use screens within Java however due to time limitations this was not possible

P18

Save System	Save Game Load Second attempt	GameScreen	Game is Loaded again	Fail	Load game screen cannot be accessed as no different screens can be opened

This issue was caused by the above one, as the game cannot be loaded again if the save state cannot be initialised, the same solution would be applied here with better knowledge of LibGDX screens and more time spent understanding the functionality.

P19

Save System	Save game across plays of the game	-	Game can be opened again after closing the program	Fail	Game state cannot be saved across separate runnings of the program

The test is here more as a proof of lack of build, this feature was attempted but was not built due to it requiring a different saving method involving external files, and as we couldn't get the saving methods to work effectively (P:17,18) we moved this below that on the stack of jobs and as such it wasn't attempted, as above this could have been solved with more time on the project as the previous issues would have been resolved and this could have been attempted.

Limited Unit Testing functionality

Some unit testing was done however it could only be performed on aspects of the program that were abstracted from the main functionality of the program, therefore only asset tests were done, however the build for healless running was left in to demonstrate the possibility of being able to test if MainScreen were rewritten to separate the Logical and GUI aspects of the game allowing for LibGDX headless runner to be able to run, these are inside the tests folder at the top of the GitHub.

[1] https://github.com/TomGrill/gdx-testing