2 Change Log

2a. Before beginning this assessment we a developed a plan to manage our changes to the previous teams deliverables (documentation, website and code) which focussed on three main goals:

- Identifying all necessary updates to the deliverables
- Prioritising these changes (in particular identifying which updates were most necessary in order to meet all deliverable + software requirements for the assessment)
- Tracking and recording changes made

When choosing which game to take over we discussed the merits not only of each team's game but also the accompanying documentation and website, as a result the game we chose to take over (team 8/delta ducks') had deliverables which we felt sufficiently covered the brief for assessment 1. Consequently, we were able to focus our time during assessment 2 mostly on changes to extend the game to meet the new requirements, and resulting changes to other deliverables, rather than spending time updating the deliverables to meet assessment 1 criteria.

When researching change management approaches one of our main sources was the change management section of 'Software Engineering' by Ian Sommerville. This source gave us valuable insight into the importance of change management and its different aspects; however, we felt the specific change management approach described was too formal and documentation heavy for a project of this scale and would have caused unnecessary delay/inefficiency. Instead we used this research to construct our own system for change management, retaining the concepts we felt would work for our team and project, whilst redesigning the elements we thought weren't suitable.

As described above our changes to other deliverables were largely influenced by our updates to the game itself. As a result, we focussed on identifying necessary software updates before identifying the necessary updates to other deliverables.

We identified necessary software updates similarly to how we elicited requirements during assessment 1. Namely, we discussed the given requirements before engaging in a client meeting in order to discuss ideas and ask clarifying questions. After this client meeting, we then proceeded to develop a formal list of updates we would make to the existing game.

Next we assessed the existing documentation and website in order to identify what aspects would be out of date or inaccurate in light of our updates, including updates to the requirements, architecture, method selection and planning and risk assessment documents which are detailed in the latter sections of this document:

Requirements

https://stanbsky.com/How-Hard-Can-It-Be-Redux/ReqTable.html

In all categories of requirements, we have removed individual requirements from the tables which the previous team felt were suitable. They were removed due to being either not completed by the previous team, or that we as a team decided not to include as they do not improve the quality of the game. These requirements are detailed below:

Functional Requirement Changes:

A large amount of additions needed to be made to the functional requirement section, due to the fact that we had to implement many new coding features. As each feature naturally creates several requirements each, there are many rows which were not there before.

Removed Functional Requirements:
“FR_ACCESS_OPTION”: We felt like the game was playable enough without having a switch to change the physical appearance of the game.
“FR_NEW_GAME”: Instead of automatically starting a new game, we instead decided to have the option to restart available to the user directly.

Non-Functional Requirement Changes:

The only requirement we added in this section was “NFR_ENTERTAINMENT” this was because, while the game had the game general goals. As it is now being presented to the customer as a final product, it should meet the customer’s needs more accurately.

Removed Non Functional Requirements:
“NFR_NO_INTERNET”: Didn’t feel this was relevant in an offline game environment.
“UR_ONE_SCREEN”: Should’ve been in User Requirements

User Requirement Changes:

The new User Requirements are all representative of the new features the customer wanted adding in part 2 of the assessment. These cover the wanted features, as well as the way we plan to implement them.

Removed User Requirements:
“UR_JAVA_GAME”: This requirement is not relevant for the user experience so it was removed from this section.
“UR_ONE_SCREEN”: As the game was already on the single screen, we thought it wasn’t necessary to create our game with this specifically in mind.
: “UR_COLLEGE_DFT”: Our customer meeting did not specify this as a requirement so we did not make this feature exist.

Architecture

https://stanbsky.com/How-Hard-Can-It-Be-Redux/Arcs.html

Due to the original diagrams being too small and too cluttered to be shown on the document, both abstract and concrete architecture have been recreated from the ground up to provide a clearer and more accurate presentation of the program's relationship between different classes. The diagram has been split into different sections for easier viewing, grouping classes with similar relationships into a diagram, combining the 'links' together and colour coding the lines for easier tracing between diagrams. The UML diagrams have included newly added classes and parameters to reflect the program's ability to fulfil new requirements such as saving/resuming game state, different levels of difficulty, powerups, testing (NFR_TESTING) and interface scaling (FR_SCALING) etc.

Method Selection and Planning Change Log

Note: Due to the nature of this section, we have not updated the original files from Team 8, however this change log reports every difference between our teams for this category.

a) Team 8 before us chose agile software development as their engineering method for reasons outlined in their own report- but simply put, they frequently cycled between doing development then receiving customer feedback. We decided as a team that we would instead take a hybrid approach- for the design and testing stages we used a waterfall approach, but an agile approach was used for the refactoring stage. We did this because we felt that most sections of the project were heavily reliant on past stages, however the refactoring needed frequent collaboration and communication for those working on it.

Below is a table detailing the key differences and similarities in our methodologies (differences in orange, similarities in green):

Team 8	Team 3
Multiple iterative customer meetings	One customer meeting pre-design
Team ‘scrum’ meetings consistently held once a week	Regular team meetings scheduled when project situation calls for it
Teams often split to utilise each team members strengths and experience (e.g., software and documentation teams)	
Frequent delivery of partially completed product to the customer	Focussed on delivering a single complete product to the customer
Most important features priorities, to ensure their delivery	

We used several of the same tools as Team 8, however these are the new ones:

NEW Tiled: This was used to create what the map looked like. The previous map was considerably bigger than what would be comfortable to play, so a new map was made with this application.

NEW Trello: This tool was absolutely vital in keeping track of what progress had been made on each individual task. Team members were also assigned to tasks, making it easy to see who should be assigned to which tasks.

NEW GitHub Issues: During the bulky reactoring stage, the issue section on GitHub was extremely useful in keeping track of things that needed attention, while alerting others in the team who may have insight, and allowing the author to move on.

NEW paint.net , Gimp and Python Pillow: These were all used to create the art, including the water texture. These are simple applications to use which is why they were selected: the graphics of our game aren’t extremely taxing.

b) As a team we continued with the following team roles from assessment 1:

- Meeting Chair – Stan
- Secretary – Jarred
- Librarian/Report Editor – Alex

Task allocation took place throughout the assessment process as the workflow passed hands multiple times. As this was happening, we tried to keep these assignments in line with each person's particular skill set in order to ensure efficiency and comfort for everyone involved. In a similar way, the team leaders also changed a few times as different team members are natural leaders in different areas. For example, as the deadline got closer in the last few weeks, Joe was the most communicative to members, informing them of the remaining needs of the project. This is a similar approach to Team 8, because they also did not assign a set leader at the start and instead waited for a natural leader to emerge- they kept this leader as someone to lead discussion and oversee everything and this status was not reassigned.

c) The plan of action for our project was very adaptive and frequently changed, so we do not have an extremely detailed and structured plan like the previous team, however you can find the loose plan we all worked to below in the following link.

https://stanbsky.com/How-Hard-Can-It-Be-Redux/Gantt.html

Risk assessment and Mitigation Change Log

https://stanbsky.com/How-Hard-Can-It-Be-Redux/RiskTable.html

Approach

When considering the risks associated with this project after user safety, the next biggest priority was meeting requirements for the requested features.
This meant that we could go into modifying the existing code having considered the biggest risks and avoiding making our work more difficult before we got started.

Removed Risks

R2: This was simply removed as that break had passed, ready to be replaced by R23

R17: As the game was never planned to have any online functionality, it was no longer necessary to be looking out for the risk

These were the only two risks removed, as other risks were still at least partially relevant to what we were working on in part two. Removing them felt to be

Newly Added Risks

R23: Although R2 was no longer relevant, we wanted to still consider the risk of ‘getting out of touch’

R24: This was the largest worry when it came to the save function, as it would massively limit the games functionality

R25: When it came to adding new functionality, the main risk by and large with the code that we chose was that classes frequently were created without regard to good software engineering practices, making them difficult to extend and dangerous to modify