Bermuda Digital Entertainment (Team 7)

Product Requirements

Sebastian Sobczyk, Joseph Sisson, CJ Donoghue, David Ademola,
Prajwal Binnamangala, Aymeric Goransson

Requirements Report

We have received the initial set of requirements as a product brief from our customer – Prof. Dimitris Kolovos representing the University of York. Upon accepting the project, we were able to arrange a meeting with one of the stakeholders and elicit further requirements and specifications about the expected product. The requirements have been presented as such to comply with the stakeholder's aim for the product to be used during various promotional events such as open days, UCAS days etc. therefore requiring a specific set of qualities.
Upon discussing further details in a meeting, some requirements and preferences have come to light when proposing design ideas and potential solutions. This was also fuelled by the aim of the product and potential target audience defined by said aim.
Where any uncertainties arose, further questions were asked of the stakeholder. Conclusions drawn have helped to better understand the purpose of the product and identify potential risks. Solutions and alternatives to said risks were decided upon in a team effort.

Upon researching the methods of requirements presentation, we have concluded that while there is no concrete standard that is widely accepted, there are various considerations that must be taken into account when such documentation is constructed. These can be mostly condensed to: clarity, editability and accountability.

In order to keep in line with the aforementioned ideas, we have decided on a structure that is clear and easy to follow and understand with each requirement - or group of related requirements - having an index number assigned for ease of identification and referencing. This also helps to satisfy the idea of editability, as requirements can be easily added to the document following the same format, or even inserted in the middle of a group of requirements.

To make sure that the requirements can be accounted for, we have devised a test for each requirement that can be used to reach a binary judgment on whether the tested requirement has been met.

This approach leaves very little room for confusion and provides a clear set of pointers against which the project can be assessed to have been completed successfully.

Statement of Requirements

1.0 – Fully functional “York Pirates” game with predicted behaviour and short playthrough
- 1.1 Around 5-10 Minutes gameplay time
- 1.2 Small to medium map size
- 1.3 Meaningful Objective of the game
- 1.4 No Story
- 1.5 No progress saving

2.0 –Real-time style in game combat against colleges.
- 2.1 At least three colleges in the game, one building per college.
- 2.2 Being able to attack colleges
- 2.3 Colleges must fire back
- 2.4 being able to dodge cannonballs from enemies

3.0 – Meaningful point system

4.0 – Clearly defined objectives

5.0 – PG Rated Art Style that grabs attention and is visually attractive

6.0 – Scaling from 13 to 27 inch screens
- 6.1 Screen scrolling
- 6.2 Designed for PC/Laptops only.

7.0 – Easy to understand controls, either mouse and/or keyboard.

8.0 – Product to be submitted as a single .jar file.

Requirements Analysis

1.0

The overall aim needs to be considered together with the environment the product will be used in. Too simple and short of a game may make the product unappealing and have an opposite effect to the desired outcome. Too complicated of a game may make the target audience confused and disinterested. To strike a balance between the two the gameplay should be scripted so that all aspects of gameplay are well defined and controlled.

1.1

The expected length of 5-10 minutes spent on the product per user leaves little room for getting used to the controls and environment. This means that everything must be straight-forward and easy to follow and identify. Oversimplification, however, may result in the game being too short or too trivial, resulting in a negative response. An alternative is to focus on making the product slightly challenging while giving the user a quick explanation of the world and controls and making them comprehensive, yet also intuitive.

1.2

The map size is heavily connected to requirement **6.0** – scalability. Too large of a map may confuse the users and leave them lost in unpopulated terrain or struggling to find the objectives. A small map may make the game trivial, unchallenging, and disengaging. The map should therefore not be visible all at once, but it should also give pointers towards the next/main objective.

1.3

The objectives cannot be too simple as the gameplay time and engagement would be compromised, but they also should not be overly complex to keep the game short. A randomized objective list may result in different playthrough experiences leading to vastly varied outcomes. Linking back to the main requirement 1.0, the objectives should be scripted and either kept the same or have the same goal-weight with controlled randomness.

1.4

No risks associated.

1.5

As the game is to be short and a single-time playthrough experience for expositions, the lack of a save function is intuitive and does not negatively impact gameplay experience.

2.0

The combat is risk-prone due to two factors. Control implementation and vast skill differences between members of the target audience.

The controls must be intuitive and straightforward enough to be quickly mastered and will be addressed in requirement analysis **7.0**.

The skill and hand-eye coordination of the average user is expected to be that of an average 'non-techy' and non-gaming person. Therefore, the combat cannot be too complicated or too hard to succeed in. Otherwise, many users would be stuck fighting the first college/enemy and could quickly give up upon failing to win. The combat cannot be too easy, however, to not make it seem as a means of artificially extending the gameplay but rather as a core mechanic of the game.

2.1

Placing more than 3 colleges could help introduce some controlled variability into the game as the objectives would hold the gameplay weight (i.e. destroy College X) while randomising the exact building to be destroyed.

2.2 – 2.4

The college combat, as discussed in point **2.0** cannot be too complex nor too trivial. Introducing two-way combat with a dodging mechanic strikes the perfect balance of covering our bases. With intuitive controls the dodging will be easy enough to give the player an advantage and agency in the fight while not requiring to be skilful.

3.0

The point system of the game, as many other aspects, must comply with the short gameplay and not require too much time investment to be meaningful and useful. To tie-in the aim of meaningful combat addressed in **2.0**, upon destruction of each college enough points should be acquired to receive an upgrade for the user's boat. If too much focus is given to the point-system it could derail the main objectives of the game and prolong the gameplay.

4.0

As pointed out in **1.1**, **1.2** and **2.1**, the objectives must be concise and easy to understand. The clarity of the objective is one of the main defining factors of the quality of the gameplay and how smoothly it progresses. Making the objectives too complicated may result in confusion and inability to finish the game, while objectives such as "swim 100 metres" can make the game look infantile in its development. Combat-focused objectives solve both problems as addressed in **2.1** and **3.0**.

5.0

The art/graphics style must capture the attention of potential users while they are walking around the promotional areas. Therefore, the sprites must be clear and visible enough but cannot clutter the screen so that there is no risk in negatively impacting clarity while playing the game. Pixelated or too simple graphics will not attract any attention while overly complex and perfected graphics may look out of place in a simple and short game.

6.0

Screen scalability is also related to the earlier art style discussion in **5.0**. When scaling from a smaller screen to a larger one there is a risk of the sprites stretching and getting pixelated. Constant size of sprites may introduce clarity issues on smaller screens. Screen scrolling is the best tool to deal with this as illustrated in the next point

6.1

Scrolling of the screen is necessary to address the scalability and map risks. With a scrollable map and gameplay area we can make the world larger than the screen to address cluttering and do not have to confine the map to the visible area.

6.2

As presented on PCs and Laptops only during promotional events, designing for phone/tablets and different operating systems would waste time and could lead to an unfinished product.

7.0

As addressed earlier in **2.0** the controls must be intuitive. Like the combat implementation, controls may also negatively impact the gameplay when designed poorly or being too complex for so-called 'unskilled' users. The controls must also be responsive in an expected manner when it comes to scrolling the map and moving the boat around. Sticking to keyboard controls will help with not confusing inexperienced players who would need to use a trackpad on a laptop.

8.0

The deliverable must be easily transported across systems (Windows and Linux). This makes the .jar executable the best form of compilation. However, a small but non-negligible risk involved in this method is that some of the sprites, code and therefore functionality may be forgotten to be included. When compiling the game into the .jar executable we must take extra care to make sure that it is the complete game with all its contents to prevent any mishaps during presentations.

Requirements Testing

1.0 - Play-test the game and make sure no bugs are present, while also:

1.1 - Check how long the game takes to complete by someone who knows what to do as well as an uninformed person and compare the times.

1.2 - Judge whether uninformed players get lost while looking for targets and objectives.

1.3 - Check whether the Objective is layered and ask for feedback from a play tester

1.4 & 1.5 - Make sure no unnecessary features are added

2.0 - Check whether the projectiles and objects act and change in-real time as opposed to set long time intervals.

2.1 - Check the number of Colleges in the game and their structure

2.2 - Check whether the player can fire projectiles at the enemy structures and damage them

2.3 - Check whether the enemy structures fire at the player

2.4 - Check whether the enemy projectiles can be dodged and do not always hit

3.0 - Check whether the point system works and provides a metric to judge play-through quality.

4.0 - Check whether the objectives are easily understood by uninformed players and if the explanations provided are sufficient

5.0 - Check whether no appropriate graphics have been added anywhere in the game or asset repository. Test player feedback regarding the color scheme and graphics style.

6.0 - Test the behaviour of the application and quality of graphics on different machines and screen sizes. Test whether quality is preserved when the window is resized.

6.1 - Test the screen/camera scrolling mechanic and whether it behaves as expected.

6.2 - Check the application runs smoothly on desktop computers and laptops running windows and linux

7.0 - Test with uninformed players whether the controls were understood easily and whether the 'controls' screen is sufficient in explaining them.

8.0 - Check whether the project compiles to a single .jar file with all dependencies and runs smoothly.