Introduction

Since finishing Assessment 1 we have gone through the requirements and made some changes. To identify which requirements needed changing/adding/removing we went through each requirement individually to make sure that it is a requirement rather than something that would be decided at the design stage. This resulted in some requirements being removed from the list. However we decided some of these requirements were legitimate requirements and that we had just put too much detail into and had elaborated too much on how these requirements would be achieved, therefore we decided to remove the extra information and keep a changed version of the requirement.

After this process we then went through the requirement list again making sure that it would be possible to accurately assess whether we have met a certain requirement or whether we have failed to meet it. This process resulted in several requirements being reworded and changed slightly to make this possible for all requirements.

Finally after adding, changing and removing requirements we did a last check that none of the requirements were contradictory or redundant. To make sure we had no redundant/overlapping requirements we went through each requirement and asking ourselves whether the requirement seems to be very similar to another. If so we compared how we would test to assess our completion of each requirement. If the test for both were the same i.e. if we prove we have met one requirement then we have also completed another then we could remove one of the requirements from the list. We also looked for any requirements which were contradictory. This was a similar process to redundant requirements. However if there were any requirements that when tested would automatically fail another requirement, then we would have to reword or maybe take out/replace that requirement.

Changes Summary (Changes also underlined)

Requirement 1.2 (“Ranged attack uses ammunition”) has been removed because it describes a design element rather than a requirement.

Requirement 2.1 and 2.2 (“The player may manually trigger a save, depending on the difficulty”) again tries to decide how requirement 2 as a whole will be implemented. Therefore, we have decided to remove 2.2 and reword 2.1 so that it is more of a requirement than a design decision.

Again we felt that requirement 10 (“The game must end when every area of the university has been conquered, or when the user’s health and lives are both 0”) was worded in such a way it sounded like a design decision so we reworded the requirement as above.

We decided that in requirement 12 (“The gameplay should be compartmentalised in such a way that single sections can be played in 5-10 minute chunks for use in UCAS and open days, but the game can still be played from start to finish enjoyably”), the final sentence being “but the game can still be played enjoyably” will be hard to assess so we reworded it as the above. We also decided that the sentence “for use in UCAS open days” was unnecessary as again it was justification for the requirement.

On doing the final run through we decided that the last part of requirement 11 was not necessary and so we removed it.

15.1 (“The controls must be similar to other top down games, for familiarity.”) again seemed more like a design decision rather than a requirement because the requirement just states that the user will use a mouse and keyboard and again this sub-objective is saying how we would implement that therefore we removed it.

For requirement 18 (“The user must understand what most of the dangers to their character are.”) The phrase “understand most of the dangers” seemed vague and hence difficult to assess whether we have achieved the goal, therefore we reworded it to the above requirement.

Requirements

System Requirements

Functional Requirements

1. The game must feature the player character, which must be a duck.
   1.1. The player character will have a ranged attack and a melee attack.

2. The game must be able to save its state and load it at a later point in time, resuming from where it was saved.
   2.1. The game will save after every level.

3. The game must take place on the University Of York campus.
   3.1. The game must contain at least 8 different campus locations that the player can move through.

4. The game must have a GUI.
   4.1. The GUI must show the location of the player within the University.
   4.2. The GUI must show the part of the University in which gameplay occurs.
   4.3. The GUI must show the amount of points the user has.
   4.4. The GUI must show the location of any obstacles.

5. The game must present at least 5 obstacles, which may be physical obstacles or enemies.
   5.1. The game must contain at least one random obstacle.
   5.2. The game must contain at least one objective-specific obstacle.
   5.3. Enemies will be either melee or ranged.
   5.4. If melee enemies touch the player character, the player character will receive damage based on the type of enemy.
   5.5. Ranged enemies may move, and will shoot projectiles at the player character.
   5.6. If a player character touches a projectile shot by an enemy, the player character will receive damage.

6. The game must contain at least 8 different objectives for the player to complete.
   6.1. Completion of an objective must score points for the user.
   6.2. An objective must be completed by acquiring resources and special powers.
   6.3. Objectives may be “quest” objectives or “survival” objectives.

7. The game must have rounds.
   7.1. The start of a round must present an objective that a player must complete to pass the round. The objective may be randomly allocated, or the user may be permitted to choose from a set of predefined objectives.
   7.2. A new round must generate at least one type of random obstacle.
   7.3. A new round must generate at least one objective specific obstacle.

8. The player must have several special abilities.

8.1. The player must be able to fly, walk and swim.
8.2. The player must have 3 other abilities which can be obtained during gameplay.

9. The game must contain resources which the player can collect to assist them in completing the objectives.

10. The game must end when every area of the university has been conquered or the player has run out of health.

11. The game should have different levels of difficulty.

12. The gameplay should be compartmentalised in such a way that single sections can be played in 5-10 minute chunks

Non-Functional Requirements

Constraints

13. The game must be able to run on any standard University of York computer.

14. The game must not require any significant financial cost to create.

User Requirements

15. The user must be able to control the player character with a mouse and keyboard.
15.1. The numpad should not be necessary (not all keyboards have them).

16. The user must be able to pause the game.
16.1. The user must be able to quit from the pause menu.

17. The user must be able to view their inventory.

18. The game must make all dangers visible.

19. The user must understand what the overall objective of the game is.

20. The user must be able to monitor the progress of their current objective(s).

Risks

3. It is possible that the University will not consent to the use of the campus in the game, although this is unlikely since this is an educational task and the University of York Communications Office is one of our customers. If this happens we will hold a meeting with our customer(s) to discuss any possible alternative depictions or locations.

3.1 A risk here is that many locations within the University have a very similar style and appearance, so it may not appear as though we have 8 different locations. We will have to choose our locations carefully to keep a good level of variety in the game. This risk can also be managed with the use of level specific obstacles, scenery and resources.

4.1/ 4.2 Referring to the above, lack of differentiation between locations may make the map unclear and the user may be unsure of where their duck is. We can minimise this risk with the above measures, and by making the GUI simple and clear.

4.4 An unclear GUI may mean the user incorrectly perceives some obstacles which may lead to an unenjoyable or frustrating experience. By making sure our GUI is clear we can eliminate this risk.

9. The risk here is adhering to the correct level of realism desired by the customer, who initially stated in the brief that the movements have to be fairly realistic, citing that 'it would be ridiculous to think a duck could open doors', but later talking about the use of special powers to enhance the ducks ability. We minimised this risk by asking our customer for clarification and were told that movement does not have to be highly realistic, however they would like us to consult them about any ideas we have for the game that may impact the realism of the movement.

20. Making the danger of obstacles obvious to the user might detract from the realism of the game.