Requirements

TEAM 14 [ENG1]

1 Introduction - Part A

1.1 Project

This is a requirement specification document for a single player game based on the Dragon Boat Challenge. The player will participate in the Dragon Boat Challenge race as one of six teams, where they will compete for the title of the York Dragon Race and receive a gold medal. There will be four legs in total, with the fastest team in each leg going through to the next leg. In the final leg, the first-place team will receive a gold medal, the second-place team will receive a silver medal and the third-place team will receive a bronze medal. There will also be obstacles that the user will have to avoid that will slow down the users speed and the robustness of their boat.

1.2 Development

We began by discussing the primary requirements of the game and how we intend to implement certain features, for example, using the left and right arrow keys to control the direction of the boat. We expanded on these requirements and followed up on any queries about the design choices of the game during a Team-Customer meeting on the 21st October.

An important feature of the game is that over time, the paddlers in the team get tired, so speed, acceleration and manoeuvrability decrease progressively during every leg. Following this, we agreed that each leg should be around 1-2 minutes long, with the entire game lasting between 5-10 minutes, so that the user doesn't get bored during the legs and that tiredness can be recovered by stopping the boat in order to help the player.

Since every boat must remain in its lane for the duration of the race, we agreed with the client that leaving the lane may result in a penalty, which will consist of adding a fixed number of seconds to the players time proportionate to the number of seconds spent out of their lane. This was to keep the penalty consistent and proportionate for each player.

Another essential part of the game was that obstacles may be found in the river and that these obstacles should be both static and dynamic to provide variation and to increase difficulty when dodging objects. Following this, it was agreed that colliding against obstacles will progressively reduce the robustness of the boat until it breaks down and the game ends. After discussing with the client, it was decided that your boat and other boats can also be classed as obstacles, so if a player hits another team's boat then the other team's boat will also be damaged proportionately to the robustness of the players' boat.

During the meeting, further requirements were also identified, such as that the difficulty of each leg should increase progressively and should be set and independent of how well the user does in order to ensure that results are fair and consistent between different players. The first leg should be considered a 'practise' leg in order to ease the player into the game so that they can intuitively learn how the system works. This is so that the user is not immediately put off by any difficulty and can adjust to the controls without being at too much of a disadvantage.

A main menu screen will be available with only a 'Start' button. No record table shall be kept in order to preserve memory and for simplicity, due to the tight deadline.The layout of the game, during play, should be of seven lanes with the players boat in the middle lane. This is so that the player can see how well they are doing compared to other 'teams' and that the width of the lane isn't too wide as this would make it easier for the player to dodge obstacles

2 Requirements - Part B

2.1 Presentation

We will use three tables to present the list of requirements that we will need to consider: User Requirements and System Requirements. Each requirement has a unique ID, which will help us to identify and refer to different requirements, and a priority rating to help the Development Team understand which requirements are most important. We will also include any assumptions, risks and alternatives for each requirement.

2.2 User Requirements

Below is a table presenting the user requirements - what tasks the user should be able to carry out during the game process.

ID	Description	Priority	Assumptions, Risks and Alternatives
UR_Main_Menu	The user should be presented with a main menu, where from the user can start or exit the game	Shall	Assumed that the boat choice area will be done automatically after pressing start The start and exit buttons should be clear and obvious to the user
UR_Option	The user can pick a type of boat from a predetermined selection following from the main menu screen	Shall	Boats must have unique specs, such as: speed, acceleration, weight, strength and handling
UR_Damage	The system shall allow obstacles to damage the users boat and slow it down proportionately	Shall	If the boat's health decreases too far, the boat will break and the game will end
UR_Complete	Users will be allowed to win or lose the game after a total of 4 legs	Shall	Win should be where the users complete all four legs and have one of the top 3 times. Lose should be where the users did not complete the game, possibly by losing all of their boats health or being eliminated before the final round.
UR_Records	The users will be able to see the time it took them to complete the current leg Users will also be able to see their current stats	Shall	Users might feel more excited when they take a short time to finish the leg/game. Users need to see why their boats may be behaving worse than before or they'd get annoyed at the game
UR_Race_Track	The users can navigate their racetrack, as well as neighbouring tracks, by steering left and right	Shall	Assumed that the boats might be damaged and slowed down when they collide boundary of the racetracks
UR_Endurance	The speed, acceleration and maneuverability of the user will decrease progressively during every leg - this can be recovered by not moving the boat left or right	Shall	Assumed that the stats for the users boat will be reset at each leg Obstacles will be needed to prevent the user keeping the boat in one direction and not having their stats decrease

UR_Penalty	If the users enter a neighbouring racetrack, they will receive a proportionate penalty	Shall	Risk that user may not know which is their lane and constantly receive a penalty - need to make it clear which lane is users
UR_Medal_ Screen	If the user has one of the top 3 times in the final leg, a medal screen will show with either a gold or silver or bronze medal Otherwise, a medal screen will show but with no medals	Should	Alternatively the medal could be presented over the top of the gameplay screen

2.3 System Requirements - Functional Requirements

ID	**Description**	**User Requirement**	**Assumptions, Risks and Alternatives**
FR_Boat_Health	The system will keep a health variable for each boat that may have different values depending on the type of boat selected	UR_Damage	Alternative would be a symbol of health to show how many times they can collide with an obstacle before the boat breaks
FR_Damage	The system will reduce the health of the boat when they collide with an obstacle	UR_Damage	Assumed that the amount of health deducted will be set according to the obstacle type
FR_Boat_Broken	The system will allow the boats to be broken when they take too much damage	UR_Complete	Assumed that the boat will be destroyed when the health reaches zero.
FR_Boat_Move	The system will allow the boats to move left or right to a avoid the damage from the traps by using the left and right arrow keys only	UR_Race_Trac k	Assumed that the boat will be slowed down when they collide edge of racetrack or traps Alternative would be the A or D keys
FR_Boat_Slowed _Down	The system shall reduce the velocity of the boats when they collide with any obstacles	UR_Race_Trac k UR_Damage	Risk: The system might not detect the collision
FR_Complete	The system shall allow other boats to win the game and receive a medal	UR_Complete	Assume that there can be three other boats that have the potential to get the three fastest times Risk: the system-controlled boats may always perfectly dodge obstacles so always have fastest time
FR_Selection	Each boat type will have unique, predetermined stats that are set within the class	UR_Option	Risk: Stats could get muddled up, they need to fit the boat description i.e. a fast boat can’t have a low speed

FR_Compare	The system must compare the users time with the generated times of the other boats	UR_Complete UR_Medal_ Screen	Risk: The generated times of the other boats need to vary between worst-case, average-case and best-case to prevent them constantly winning
FR_Time	The system must keep track of the users time during the current leg	UR_Complete UR_Records UR_Penalty	Risk: The penalty time needs to be consistent throughout gameplays
FR_Stats	The system must keep track of the preset stats of each boat and then how they decline/increase throughout the game	UR_Records UR_Endurance	Assumed that there will be no giant leaps in increase/decrease

2.4 System Requirements - Non-Functional Requirements

Non-Functional Requirements are qualities that a system must have

ID	**Description**	**User requirements**	**Fit Criteria**	**Assumptions, Risks and Alternatives**
NFR_ Response _Time	The system will be respond quickly to the user, i.e. when they use the arrow keys to move left and right	UR_Race_ Track	Respond in <milisecond	Assume similar for clicking the ‘Start’ button or selecting a boat, etc.
NFR_ Control	The system must allow the boat to be easily controlled by the user	UR_Race_ Track		Risk: The boat may be too slow and hard to avoid traps when handling. This could lead to the player getting frustrated and finding it unfair.
NFR_ Availability	The game must be accessible across different platforms i.e. Windows, Linux...	UR_Main_ Menu	Game should load and be fully functional (all options working in reference to NFR_Response_Time) across tested platforms	Risk: May not have time/ability to test across all potential platforms