University of York

Department of Computer Science

Requirements Deliverable

Team 19:
Julia Kunikowska
Jack Longmuir
Justin Mendon
Luke Roberts
Douglas Sword
Andrea Zhu

Deliverable submitted for Team assessment 1:
Engineering 1 - COM00019I

Module Leaders:
Prof. Dimitris Kolovos
Dr Javier Camara Moreno

Requirements

The requirements for The York Dragon Boat Race have first been developed through reading the product brief and carefully evaluating the specific features that are mentioned in them. Some of the requirements were explicitly stated like the attributes of the boats and increasing the difficulty, however there were also requirements that were too vague to leave on their own, like the obstacles on the river. After coming up with a list of requirements, together as a team we arranged a Team-Customer meeting to further discuss the requirements with our customer. The meeting with the customer occurred on the 19/10/2020 where we asked a series of prepared questions on the requirements to get more information. Some of the questions we included were:

1. “What is a “penalty”? Should a team be disqualified for breaking the rules? Time restrictions (for how long allowed outside of a lane)?”
2. “Do you have a preference for what the obstacles in the river should be?”
3. “How many teams should compete in each leg?”

After the meeting we came up with the user requirements, which we then divided into mandatory and additional to ensure that the team would stay on track. This was decided with the customer during the Customer Meeting by analysing the customers responses. After finalizing the user requirements, as a team we discussed the system requirements and divided it into function, non-functional and constraint requirements.

1. User requirements

ID	Description	Associated Risk	Priority
UR_CROSS_PLATFORM	The game should be accessible across all platforms i.e mobile, desktop	Game is unplayable for the user (R2 from risk table)	Shall
UR_BALANCED_BOATS	The boats in the game should be balanced	Game is unfair to play for the user	Shall
UR_INCREASED_DIFFICULTY	The difficulty will increase each completed leg	Game becomes too easy/ boring to play	Shall
UR_STAMINA_RESET	The boats stamina will reset each leg	Game becomes too hard to play	Shall
UR_OBSTACLES	There should be static and dynamic obstacles that boats will avoid or else the boats robustness will decrease	Game becomes boring and unplayable	Shall
UR_FIXED_PENALTY	The penalty for leaving the designated lane will be fixed each time	Game becomes too easy to cheat and gives no consequences for breaking the rules	Shall

UR_SPLASH_DIALOGUE	There will be splash dialogue explaining the controls	User does not know how to play the game and can become frustrated.	Should
UR_SMARTER_AI	The boats CPU will improve each leg to increase difficulty	There is no competition in the game for the user	Should
UR_CONTROL_BOAT	The user can control the boat using the arrow keys	The user should be able to control the boat or else they cannot win	Shall
UR_WIN_LEG	The winner of each leg will be determined by which boat crosses the finish line with the shortest time	There will be no point of playing the game if there is no way to win it	Shall
UR_GAME_OVER	When the robustness of the boat reaches 0 or all the legs are complete, the game is over	There are no consequences to the actions of the user and the game can become too easy	Shall
UR_GAME_IS_PLAYABLE	Makes sure that the player can play the game	The game cannot be played otherwise by the user	Shall

2. Functional requirements

ID	Descriptions	User requirements
FR_CONTROLS	The game will allow the user to move the boat	UR_CONTROL_BOAT
FR_TIMER_LEG	The game will show a timer displaying how long the leg is taking	UR_WIN_LEG
FR_OBSTACLE	The screen will display obstacles when they are meant to appear	UR_OBSTACLES
FR_HP	The robustness of the boat will decrease when an obstacle is hit	UR_OBSTACLES
FR_BOAT_PLAYER	The game will provide a boat that the player can control	UR_CONTROL_BOAT
FR_BOAT_CPU	The game will provide other boats that are controlled by cpu	UR_IMPROVE_AI

FR_NO_BACKWARDS	Boats are not be allowed to move backwards	UR_CONTROL_BOAT
FR_ATTRIBUTES	the boats have attributes for each boat (speed, acceleration, maneuverability, and robustness)	UR_BALANCED_BOATS
FR_DIFFICULTY	The game will progressively increase difficulty	UR_INCREASE_DIFFICULTY
FR_CONTROL_INPUT	The game should not allow wrong user input.	UR_CONTROL_BOAT
FR_GAME_OVER	The screen should display game over when robustness reaches 0	UR_GAME_OVER
FR_OWN_LANE	Boats are not to be allowed to stay in another participant's lane for the entire game	UR_FIXED_PENALTY
FR_TIMER	During the race there will be a time limit that will be displayed on top of the screen	UR_WIN_LEG
FR_ANTI_CHEAT	The game should not have a way of cheating	UR_CONTROL_BOAT
FR_OFFLINE	The game should be allowed to be played offline if downloaded	UR_CROSS_PLATFORM

3. Non-functional requirements

ID	Descriptions	User requirements	Fit Criteria
NFR_INIT	The game should be able to start instantly	UR_CROSS_PLATFORM	within 3 second after accessing
NFR_LOADING	Fast loading screen	UR_CROSS_PLATFORM	The loading screen will be around 8 seconds
NFR_EZ	The game does not require training to be	UR_SPLASH_DIALOGUE	the age restriction for the game to be played PEGI +3

	played		
NFR_AVAILABILITY	the game will be available all the time to be played	UR_CROSS_PLATFORM	There is no limitation on when to access if not on a hardware level so 99% cases
NFR_CRASH	The game will not affect the device negatively	UR_CROSS_PLATFORM	CPU usage percentage: 30%
NFR_ACHIEVEMENTS	The game should allow the player to see his game achievements	UR_SCOREBOARD	Shown at the record option menu
NFR_SCOREBOARD	The game should allow the player to check his statistics	UR_SCOREBOARD	Shown at the record option menu
NFR_FORCE_QUIT	If the game freezes it will close it self	UR_GAMEOVER	30 seconds of frozen screen will result a force quit
NFR_COLOURBLINDNESS	The game will use inclusive colours	UR_GAME_IS_PLAYABLE	Allows player to see the game and win a leg