Requirements - Req1

Dragonite

Team 21

Omar Omar
Rhianna Edwards
Okan Deniz
Craig Smith
Omar Galvao Da Silva
Joel Wallis

Team 19

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

Requirements

SSON: The program shall allow users to take part in a dragon boat race game, competing against AI to achieve the fastest time in an enjoyable and easy to use manner.

Summary

The requirements were discussed during the team meetings. They were read individually by each team member from the product brief so that they could be decided upon and analysed in the following team meeting. Each member voted on the importance and risk of each requirement. Requirements were split according to whether they were user/system or functional/non-functional.

From the product brief, further requirements were devised. For example, the product brief mentioning that subsequent levels are required to increase in difficulty levels tells us that we would have to implement our AI in such a way that it would be able to avoid obstacles better and finish the races quicker in those subsequent levels.

Tables were used to represent the requirements and an importance scale was used to provide an easy way to summarise each requirement and prioritise ones with higher importance. Another reason why tables were used is to ensure that as requirements evolve, it makes it easier for the team to make visible/clear updates and changes. The importance scale was given 5 different scales which includes both must-have and unessential ends to it in order to be able to cover all possible cases of importance.

Agreements were made with the stakeholders during customer meetings so that expectations were aligned and everyone was clear on what requirements were to be prioritised. This way, requirements that were not essential were also established early so the team's focus could be shifted.

Throughout our requirement management process, we have referred to various requirement gathering resources[1] and especially made use of the requirement elicitation process. Our team-customer meetings with the stakeholders were vital and fell under the negotiation & discussion step of the requirement elicitation process diagram.

[1] https://www.tutorialspoint.com/software_engineering/software_requirements.html

Importance Scale
Vital, High, Medium, Low, Unessential

1. User requirements

ID	Description	Associated Risk	Priority
UR01	The game must be programmed in Java	Game cannot be played on Open day computers or machine that does not support Java language.	Vital
UR02	A racing game that is won by the boat with the fastest time	Players do not understand how to play the game and therefore it cannot be played or missing limbs	Vital
UR03	To have the player compete against 3-6 AI opponents	Too many opponents results in problems with processing	High
UR04	Each boat should have a different set of balanced attributes	If boats are not balanced, the game becomes too easy/hard	High
UR05	If the player leaves their lane then a time penalty will be incurred	Game becomes too easy and there are no consequences to cheating	High
UR06	Hitting obstacles will result in a decrease of robustness and disappearing of the obstacle	Game becomes boring and unplayable with no obstacles	High
UR07	If the players health decreases to 0, they will lose	The game gets boring and has no challenge	High
UR08	Player gets tired over time, therefore the speed, acceleration and maneuverability will decrease	The game will be too easy and not challenging enough	Medium
UR 09	The controls should be keyboard and mouse	There will be no way to play the game	Low

UR10	There should be good graphics, animation, audio to make the game interesting	The player will not be interested in playing the game because it looks unappealing	Low
UR11	Show time of each leg	The player won't know how long the race took them	Low
UR12	Recovering of energy when it's not used	It makes the game too hard and boring	Unessential
UR13	Implement different levels of difficulty in the game	The player will get bored over time and feel no satisfaction from playing	High
UR14	Create a save, load function	The players game might become interrupted and they have to start again	High
UR15	Implement five power-up packs which improve attributes	The game becomes too hard and boring with increased difficulties	High

2. **Functional requirements**

ID	Descriptions	User requirements
FR01	Implement AI that can control multiple boats simultaneously.	UR03
FR02	A boat class to hold all the attributes and functions for the different varieties of boat.	UR04
FR03	Boats are allowed to stay in another participant's lane but they will incur a time penalty.	UR05
FR04	Multiple game states to display information to the player and interact with the product.	UR09
FR06	An Obstacle class to hold all	UR04

	the attributes and functions for the variety of obstacles.	
FR07	An AI class that manages how the competing boats behave and adjust its difficulty given what leg of the race is in progress.	UR03
FR08	Assets and sprites to display lanes, boats and obstacles.	UR10
FR09	The game should not last longer than 3 minutes.	UR02
FR10	Control the boat using WASD / Arrow keys.	UR09

3. **Non-functional requirements**

ID	Descriptions	User requirements	Fit Criteria
NFR01	Timer to record leg times for the player and AI.	UR02	Show the recorded time of all players at the end of the race.
NFR02	The game should start immediately	UR01	the game start within 5 second
NFR03	The game should be suitable for everyone.	UR02	There is no age restriction to play the game.
NFR04	Scoreboard which displays the times of all the boats	UR02	show the order of in which the boats crossed the finish line at the end of the race
NFR05	Audio between stages.	UR10	There will be music output between stages.