Requirements - Req1

Dragonite
Team 21

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

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.htm

<u>User Requirements</u>

Importance Scale
Vital, High, Medium, Low, Unessential

Requirement	ID	Importance	Risk	Environment Assumptions
The game must be programmed in Java	UR01	Vital	N/a	Going to be running on a virtual machine, allows it to run on whatever hardware is planned to be used for the Open day
To build a simple racing game; the main goal is to have the fastest time available	UR02	Vital	N/a	Being used in an Open day and as such needs to be able to be picked up quickly by users
The game must be suitable for prospecting students to the University and their families	UR03	High	N/a	Usage by General Public so be suitable for a E rating
To have a single player experience in which the player compete against AI opponents, there will be 3-6 AI with a maximum of 7	UR04	High	R9	There will be only one screen, keyboard and mouse usable to users. It is a casual game so doesn't need to be too difficult The power of the hardware will be low so not too much processing should be done
The game should last between 3-7 minutes and contain 4 stages. (3 legs, 1 Final)s	UR05	High	R10	During an Open Day people may only be in a location for a short period of time; the game being too long would be lead to many users having to leave the game before it is finished
Each boat should have a different set up of 4 attributes (Speed, Acceleration, Maneuverability, Robustness) which are balanced, the player will be able to choose a boat with the non-chosen boats being used as AI opponents a) Speed is the max speed of the boat b) Acceleration is how quickly a boat gets to its max speed	UR06	High	N/a	These are abstractions of boats which represent different factors such as construction, material, design etc…

c) Maneuverability is the horizontal speed of the boat d) Robustness is a decrease in damage taken from obstacles				
The boats will be split up into lanes, if the player moves outside their lane then a time penalty may be incurred	UR07	High	N/a	Based on the lane system used in the actual dragon boat race
Obstacles will be in the boat's way, a boat colliding with an obstacle or another boat should decrease robustness based on the speed of the boat; upon being hit the obstacle will disappear.	UR08	High	N/a	I.e. rocks, ducks, geese, logs etc….
If the player's health is decreased to 0 then they lose	UR09	High	N/a	Same as the boat being destroyed
The player should get tired over time, decreasing speed, acceleration and maneuverability	UR10	Medium	R11	Same as real rowers getting tired from rowing
Controls will be given as images to the user. keyboard and mouse controls will be used.	UR11	Low	R12	Mouse and keyboard are mostly used already so users will be familiar already with the controller
The graphics will be clear and the player able to distinguish themselves from AI and the AI from each other	UR12	Low	N/a	N/a
Show times of each leg	UR13	Low	N/a	Based on the qualifying system in the Dragon boat race
Recovery of energy	UR14	Unessential	N/a	N/a
Animation to make the more interesting	UR15	Unessential	N/a	N/a
There should be variation between the different legs in the game.	UR16	Unessential	R13	N/a
Audio for the stages	UR17	Unessential	N/a	N/a
Demo's to allow the customer test the game	UR18	Unessential	N/a	N/a

System Requirements

Requirements	ID	Importance	Risk	Environment Assumptions	Functional / Non-functional
Programming language will be Java	SR01	Vital	N/a	Going to be running on a virtual machine, allows it to run on whatever hardware is planned to be used for the Open day and is strictly required in the assessment paper.	Functional
Timer to record leg times for the player and AI	SR02	Vital	N/a	Time is essential to calculate whether or not the player progresses to the next rounds.	Non-Functional
Control the boat using WASD / Arrow keys	SR03	Vital	N/a	There will be a keyboard and a mouse available to play the game with.	Non-Functional
Implement AI that can control multiple boats simultaneously	SR04	Vital	N/a	There will only be one set of input devices and the player(s) cannot control multiple boats hence AI opponents are needed to implement competitive racing.	Functional
A boat class to hold all the attributes and functions for the different varieties of boat	SR05	Vital	N/a		Functional
Assets and sprites to display lanes, boats and obstacles	SR06	Vital	R16		Functional
Multiple game states to display information to the player and interact with the product	SR07	Vital	R15		Functional
An Obstacle class to hold all the attributes and functions for the variety of obstacles.	SR08	Vital	N/a		Functional
An AI class that manages how the competing boats behave and adjust its difficulty given what leg of the race is in progress	SR09	Vital	N/a		Functional