Requirements

Group 26 - Spice Traders (prior team 22)

James McNair
Dan Wade
Alice Cui
Charlie Crosley
Rob Murphy
Marc Perales Salomo

2.a Requirement Elicitation

- Requirements were elicited from the product brief provided to all students at the start of the module.
- In order to supplement the provided requirements an interview was arranged with our client (Tommy Yuan) to clarify what we were required to produce.
- This interview took the form of a closed interview, where the stakeholder answered a set of questions that we had come up with beforehand.
- Our requirements are drawn from our analysis of the product brief and meeting notes.
- These requirements were made as top level requirements before being decomposed into smaller chunks.

Requirements Specification and Presentation Method

- Our requirements are presented to show what we have done in a more formal and practical format than the list we used when producing our project.
- In order to formalise our requirement's presentation we selected the requirements engineering method detailed in [1], to determine how our project can meet the needs of the user.
- This method splits the requirements into user requirements and system requirements (functional and non-functional).

- The distinction between user and system is needed as the user requirements can be described non-technically, which means stakeholders and non-technical people can be involved in the requirements process.
- The system requirements are a description of how the system will deliver upon the user requirements, which may use technical language (functionality, services and constraints).
- These system requirements are more useful to developers who are implementing the system described in the requirements.

- System requirements are split into functional requirements and non-functional requirements.
- Functional requirements are services that the system should provide, how the system should react to particular inputs, and how it should behave in particular situations [2]. Functional requirements can be written at different levels of detail.
- Non-functional requirements are constraints on the services or functions offered by the system. [2]

[1] I. Somerville, "Requirements Engineering," in *Software Engineering*, 10th ed: Pearson, 2015, ch. 9, pp. 102-137
[2] I. Somerville, "Requirements Engineering," in *Software Engineering*, 10th ed: Pearson, 2015, ch. 9, pp. 105

2.b Requirements Presentation - SSON

“The system should allow the user to play a pirate themed game based around a version of the University of York which is located on a lake, battling ships and plundering colleges.”

User Requirements

Requirements ID	Description	Priority
UR_PLAYER_MOVEMENT	Player shall be able to move around the map	High
UR_PLAYER_SHOOTING	Player shall be able to shoot projectiles	Medium
UR_PLAYER_PLUNDER	Player shall be able to collect and spend coins/plunder	Medium
UR_PLAYER_EXP	Player shall be able to gain experience	Medium
UR_PLAYER_UPGRADE	Player shall be able to upgrade using a skill tree	Medium
UR_ENTITY_GENERATION	Entities shall be spawnable	High
UR_MAP_GENERATION	A map shall be generated	High
UR_UI_INTERFACE	A UI overlay shall be generated	High
UR_UI_MAINMENU	A main menu should exist and be the first screen	Medium
UR_UI_PAUSE	A pause screen/other menus may be accessible	Low
UR_PLAYER_POWERUP	Player should be able to pick up five unique power ups	High
UR_DIFFICULTY	User should be able to select different levels of difficulty	High
UR_SAVE	User should be able to save the game at any time	Medium
UR_OBSTACLES	Player should encounter obstacles/enemy ships throughout the game	Medium
UR_FINISH	The game should end either on a game over or victory	High

System Requirements - Functional Requirements

Requirements ID	Description	User Requirements

FR_PLAYER_MOVEMENT	Movement is done using WASD or the arrow keys	**UR_PLAYER_MOVEMENT**
FR_PLAYER_PLUNDER	Collision of player with coin hitbox should despawn coin and increase money.	**UR_PLAYER_PLUNDER**
FR_PLAYER_SHOP	Players should be able to spend their plunder.	**UR_PLAYER_PLUNDER**
FR_PLAYER_EXP_TIME	Players should gain experience over time.	**UR_PLAYER_EXP**
FR_PLAYER_EXP_COLLEGE	Upon defeating a college, exp increments.	**UR_PLAYER_EXP**
FR_PLAYER_UPGRADE	Upon completing an upgrade, the relevant stats should increment.	**UR_PLAYER_UPGRADE**
FR_SHOOTING	Upon a trigger, shooting should commence. Releasing the trigger deactivates shooting (trigger may be assigned to various keys).	**UR_PLAYER_SHOOTING**
FR_ENTITY_GENERATION	Entities should be able to be generated as a batch, and be able to be assigned their own unique data that may be overridden by another class.	**UR_ENTITY_GENERATION**
FR_SHIP_GENERATION	Enemy ships should spawn randomly across the map.	**UR_ENTITY_GENERATION**
FR_COLLEGE_GENERATION	Enemy colleges should spawn in fixed positions across the map.	**UR_ENTITY_GENERATION**
FR_MAP_GENERATE	A map should be generated prior to the game starting.	**UR_MAP_GENERATION**
FR_MAP_COLLISION	Borders of the map should provide collision to prevent the player from exiting.	**UR_MAP_GENERATION**
FR_UI_SOUNDTOGGLE	Music and sound effects should be toggleable.	**UR_UI_INTERFACE**
FR_UI_STATISTICS	UI displays health, coins, experience, etc.	**UR_UI_INTERFACE**
FR_MAINMENU_START	Main menu allows player to start the game	**UR_UI_MAINMENU**
FR_MAINMENU_EXIT	Main menu allows player to exit/close the game	**UR_UI_MAINMENU**
FR_PAUSE_HALT	Pause menu stops gameticks (pauses the game)	**UR_UI_PAUSE**
FR_PAUSE_RESUME	Leaving pause menu resumes game from when it was paused	**UR_UI_PAUSE**
FR_DIFFICULTY	Users should be able to select from easy, medium and hard.	**UR_DIFFICULTY**

FR_SAVE	A save file should be generated as an .xml file.	**UR_SAVE**
FR_OBSTACLES	Players should encounter random obstacles which makes gameplay more difficult.	**UR_OBSTACLES**
FR_ENEMY_SHIPS	Enemy ships should attack when in range of the player.	**UR_OBSTACLES**
FR_PLAYER_POWERUP	5 unique power ups should spawn across the map that can be collected	**UR_PLAYER_POWERUP**
FR_PLAYER_WIN	When a player defeats all enemy colleges they should win the game	**UR_FINISH**
FR_PLAYER_DEFEAT	When the player runs out of health or their own college dies, they should lose	**UR_FINISH**

Non-functional Requirements

ID	Description	Fit Criteria
NFR_TIMING	Spawned entities (e.g. coins) have a finite lifetime.	All entities with a lifespan must despawn at the end of their lifetime.
NFR_PRECISION_CONSTRAINT	Entities shall not occupy the same position on the map.	No tiles overlap the same x, y coordinate.
NFR_RESILIENCE	Running the game on different hardware that meets the minimum requirements should not result in errors.	Running the system on alternative hardware that meets the minimum system specifications does not produce any errors.
NFR_INTEGRATABILITY	The system can draw upon a list of predetermined assets.	All used assets are accessessed from a predetermined list hardcoded into the system.
NFR_OPERATABILITY	The game should be easy to understand.	We have taken every possible method (within reason) to produce a comprehensible product.
NFR_ACCESSIBILITY	The system should be operable by players with impaired vision.	All text and values are large enough to be seen with impaired vision.
NFR_USABILITY	All on-screen information is comprehensible and in plain English.	All written information is displayed in English and does not contain technical jargon.