Requirements

Things needed in this document:

1. Introduction explaining how requirements were elicited and negotiated, and why they are presented as they are. Show evidence of research into requirements specification and presentation (1 Page, 5 Marks)

2. Give systematic and appropriately-formatted (e.g. tables as shown in lectures) statement of requirements, including notes of any relevant environmental assumptions, associated risks, or alternatives (3 Pages, 17 Marks)

We must use appropriate requirements referencing system (ID's for each requirement) which will allow us to easily modify these requirements later on in the process.

Requirements

Introduction

Before we started programming the game itself, setting up the requirements for our project is crucial. Having good requirements not only helps us understand what needed to be done for the game but also preventing us from working on unnecessary tasks. With that idea, we began deciding on our system requirements after reading through the project briefing document. This told us how the requirements should be laid out as well as how they should be divided between the user and the system - both functional and non-functional - as well as giving us some constraints to consider. After a brief brainstorming meeting, we put together a large list of possible requirements as well as a series of questions we had for our stakeholders (such as what art style would be best and whether a menu system would be necessary). This would help us finalise our list. Once we had a complete question list, we organised to meet with one of our stakeholders. During the 15 minute discussion, a member of our group acted as a scribe for each answer. Additionally, we recorded the conversation, with permission, in case any important details were missed as they answered our questions. This ended up being a good idea as it gave us concrete evidence of what we were told by the stakeholder without having to rely on hastily written notes from the meeting and memory.

Now that we had the input of one of our users, we could create a single statement of need and we could resolve our draft into a complete list of user requirements. Our SSON was “The game should allow a user to control fire engines and use these to defeat enemy fortresses”. From these, we derived both functional and non-functional system requirements to be used in our architecture plan.

As part of this section, we did some research into how best to display our requirements and what features needed to be included. According to the IEEE Guide for Developing System Requirement Specifications, we were given a detailed description of what should go into a well-developed requirement list as well as how we should identify the requirements necessary for our system. According to the paper, each requirement should be well detailed with risks and priorities just to name a few. It also gave advice on possible pitfalls that might arise while we produce our tables which were paramount in maintaining consistency and reliability. The document suggested that we should take advantage of several methods to research what the actual requirements should be. These included questionnaires and simulations to name a few. Luckily for us, however, we had the booked interview slot with one of our stakeholders that would go on to be one of the key sources of inspiration for our requirements. Due to time constraints, we were unable to develop more methods of research but we believe the interview sufficed for our system.

As a result of our extra research and the information presented in the lectures, we decided to present our requirements as a series of 4 tables. The first being user requirements, the second being functional system requirements, the third being non-functional system requirements and the final one being the constraints that had been put on our project and were non-negotiable . The reason we chose to do this is to allow us to easily link each user requirement to one or more system requirements in a clear manner. Each requirement is given an appropriate identifier and a suitable priority level associated with it.

User Requirements

Requirement ID	Description	Risks and Assumptions (if relevant)	Priority Level
UR_WIN	The game must be won when the user defeats all KROY bases.	KROY bases are too hard to be destroyed, impossible to win.	Shall
UR_LOSS	The game is lost when all fire engines are destroyed.		Shall
UR_PLAYABLE	The game should be playable and enjoyable by the SEPR cohort.	May cause the game to be too generic by trying to please everyone.	Shall
UR_REFILL_REPAIR	The user must return the fire engines to the station to refill and repair.		Shall
UR_FIRE_ENGINES	All fire engines should be unique.	Too many different types could lead to confusion	Shall
UR_ET_FORTRESS	All ET fortresses should be unique.	Too many different types could lead to confusion	Shall
UR_PATROLS	The aliens should have patrols.		Shall
UR_ENGINE_NUMBER	There should be at least four Fire engines.	Players may find it difficult to get along with a large number of different tools	Should
UR_FORTRESS_NUMBER	There should be six different ET fortresses.		Should
UR_MINIGAME_TRIGGER	A minigame should trigger at appropriate times during the game.	The minigame may not trigger at the appropriate times, causing players to be uncomfortable.	Shall
UR_MINIGAME	The minigame should be different from the main game but contextual.	The minigame may be too similar to the main game or not contextual.	Shall
UR_MOBILE	The game should be able to be ported across to mobile platforms.	The game may not be able to be ported or difficult to port to mobile due to lack of cross-platform features	Should
UR_MAP	There should be a map for the game.		Should
UR_TIME	The game should not last too long.	The game may drag on and be boring to play.	Should
UR_DIFFICULTY	The game should get more difficult as it progresses.	The game could get too difficult causing the player to quit.	Shall
UR_CONVENTIONS	The game should follow standard conventions.		May

System Requirements - Functional

Requirement ID	Description	Risks and Assumptions	Design Requirement
FR_FINISH	The system must recognise when all the fortresses or all fire engines have been destroyed.	Assumes there will always be a fixed number of fortresses.	UR_WIN, UR_LOSS
FR_UNIQUE_ENGINES	Each fire engine must have a unique set of stats.	Random generation of stats may result in some fire engines having the same stats.	UR_FIRE_ENGINES
FR_UNIQUE_FORTRESS	Each ET fortress must have a unique set of stats.	Random generation of stats may result in some fortresses having the same stats.	UR_ET_FORTRESS
FR_PATROL_STATS	All patrols must have stats but don't have to be unique.	The patrols may not have suitable stats.	UR_PATROLS
FR_STATS_IMPROVED	Some or all ET fortress stats must improve over time.	The stats may improve too much, making the game too difficult.	UR_FORTRESS _IMPROVE
FR_ACCESS_STATS	User must be able to access all fire engines' stats.	Need to be able to show the stats without hiding too much of the information.	UR_FIRE_ENGINES
FR_STATION_DESTROYED	Fire station should be destroyed ¾ of the way through the game.	This assumes there is a fixed length to the game.	UR_TIME
FR_MENU	The game should launch into a menu.		UR_CONVENTIONS
FR_FULLSCREEN	The game should launch in fullscreen.	This assumes the hardware used has the capacity to play the game in	UR_CONVENTIONS,

		full screen.	UR_MAP_SIZE
FR_TURN_BASED	The game should follow turn-based conventions for gameplay.		UR_CONVENTIONS
FR_PATROL_MOVEMENT	All patrols should move around the map.		UR_PATROLS
FR_MAP_SIZE	The size of the map should not hinder the gameplay of the user.	Different hardware may be able to handle different size map generation	UR_MAP
FR_ALL_MOVEMENT	No fire engines or patrols should be able to move out of reach of the player.	Assumes the player can interact with all objects that are currently on the screen. The risk is that the player may not realise they cannot interact with multiple fire engines at the same time	UR_MAP, UR_FIRE_ENGINES, UR_PATROLS
FR_INTUATIVE_CONTROLS	The controls decided on should follow the normal gaming conventions.	The risk here is that we follow conventions blindly without considering what is best for our game	UR_CONVENTIONS, UR_PLAYABLE
FR_DESTROY_FORTRESS	A fortress must be destroyed when it runs out of HP.		UR_WIN, UR_ET_FORTRESS
FR_INFORM_USER	A user must be notified if a change occurs eg. a fortress is destroyed.	too many changes occur at the same time could overwhelm or block the screen.	UR_CONVENTIONS, UR_PLAYABLE

System Requirements - Non-Functional

Requirement ID	Description	User Requirements	Fit Criteria	Risks and Assumptions
NFR_RESPONSIVE	The system should respond quickly to user input from the keyboard and mouse	UR_PLAYABLE	The response should happen <1 second after the input pressed.	Assuming that the player gives a valid input.
NFR_GAMELENGTH	Game should be completable in a reasonable timeframe	UR_TIME	The game should generally take <15 minutes.	Assumes the player is active for all those 15 minutes
NFR_IMPROVE_TIME	The fortresses should improve in a reasonable time frame	UR_TIME, UR_DIFFICULTY	Fortresses should improve every 3 minutes.	If the game goes on too long then it may become impossible to win.
NFR_MINIGAME_TIME	A minigame shouldn't take too long to complete.	UR_TIME	Minigames should be finishable in <2 minutes.	Assuming that the player understands how to play the minigame
NFR_INFORM_TIME	A user should be made aware when a change happens.	UR_CONVENTIONS, UR_PLAYABLE	A user should be informed of any changes in <2 seconds.	
NFR_PORTABLE	The game should be able to be ported to mobile in the future.	UR_MOBILE	The game should be able to be ported in <3 months.	May be difficult to fit all the elements of the game on such a small screen

Constraints

Requirement ID	Description
CON_APPROPRIATE	The game shouldn't have explicit themes and should instead be PG, and should be suitable for any user.
CON_LANGUAGE	The game must be programmed using the Java programming language.
CON_RUN	The game should run on any UoY computer science department computer running Windows or Linux.
CON_SOLD	The game must be able to be marketed and sold. This is achieved by avoiding the use of any copyrighted material or software which could cause licensing issues.
CON_ALIENS	The game must be alien-themed across the whole game including the minigame.
CON_SETTING	The game must be set in York.
CON_ACCESSIBLE	The game must have high contrast in colours to be accessible for all users.

Uses Cases For Functional Requirements

Use Case 1:

Name: Defeating all ET fortresses and beating the game causing the game to finish
Context: The user plays the game and manages to defeat the all 5 ET fortresses with fire engines still remaining.
Actors:
Primary Actor: User
Precondition: Player has destroyed all but one ET fortress.
Success Postcondition: Final fortress is destroyed and the user wins the game causing a winning event to appear.
Trigger: Primary Actor starts the use case by playing the game and the trigger for the end of the game is when the final fortress is destroyed by the user.
Main Success Scenario:

1) User defeats final fortress by getting its health to zero.
2) User is informed base has been destroyed.
3) User is informed that they have completed the game.

Secondary Scenario:
1.1) User's fire engines are destroyed by the ET fortress and the user loses.
Requirements:
Non Functional: NFR_INFORM_TIME, NFR_GAMELENGTH
Functional: FR_INFORM_USER, FR_FINISH
User: UR_WIN

Use Case 2:

Name: Fire engine approaches the fire station
Context: the user needs to refill their fire engine's water tank or repair their fire engine.
Actors:
Primary Actor: User
Precondition: The user has started the game
Success Postcondition: The fire engine's water tank and health are restored to full.
Trigger: The player approaches the fire station with a fire engine.
Main Success Scenario:

1) The user approaches the fire station with a fire engine.
2) The user chooses to refill and repair their engine.
3) The user is informed that their engine has been repaired and refilled.

Secondary Scenario:
2.1) The player's fire engine doesn't need to refill and repair and so can't be.
3.1) The fire station has been destroyed and so doesn't allow refills and repairs.
Requirements:
Non Functional: NFR_INFORM_TIME, NFR_RESPONSIVE
Functional: FR_ACCESS_STAT, FR_STATION_DESTROYED, FR_INFORM_USER
User: UR_REFILL_REPAIR

Loius E.
https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=502838&fbclid=IwAR0px3lryCljsB_eoM6jPqZCqHKd97RMuvKdXpfZVlB76z_Zlo8MpSs4Wgg&tag=1

https://pdfs.semanticscholar.org/062a/106a545dd2d9b1682c40183d7f2362d1dfd3.pdf