Module	SEPR
Year	2019/20
Assessment	2
Team	Dalai Java
Members	Jack Kershaw, Max Lloyd, James Hau, Yuqing Gong, William Marr, Peter Clark.
Deliverable	Requirements

Requirements

Single Statement of Need: The system is a game to be shown at open days for prospective students that involves the player defeating an alien invasion by taking control of the York Fire station.

The project is a game called KROY, designed for the purpose of being put on display for prospective students and parents at the University Open Day and Post Offer days. It is intended to show off the work achieved by students of the University in order to entice prospective students to attend the University of York. For this reason the product has a variety of requirements to maximise the appeal of the department. These requirements include everything from the required language that we are to be using along to certain visual elements of the game in order to make the game attractive to both prospective students and their parents alike.

The game is set in a future version of York in the year 2042 where Aliens from the planet of Kroy have invaded earth and set up a number of fortresses around York at the Famous attractions such as Clifford's Tower and the Minster. The player will be the leader of the resistance using the abandoned fire station as the resistance base. The aliens have 1 weakness, water. Fire engines can destroy the fortresses and kill the aliens, but they are patrolling the streets looking for the fire station to wipe out the last of the resistance.

This project's requirements were formed from a detailed product brief as well as discussions with our customer to allow any confusion to be quickly resolved in order to ensure that the final product is as close to the customers expectations as possible. These requirements were then discussed thoroughly allowing us to think up ways in which we can approach the task in an effective and organised fashion. We presented our requirements by adapting the concepts shown in the IEEE Requirements Specification [1] We formatted our tables for user requirements and functional requirements as per the description in section 4.1.1 in the IEEE Requirements Specification. Furthermore, we used the format for functional requirements precisely described in section 4.1.3 to allow us to include detailed descriptions and unique identifiers for each of the functional requirements. This allows them to be clearly organised based on why these are our requirements and allow us to reference them later on. This table based method also allows them to be laid out in a manner that is easy to follow and find exactly what we are looking for when it comes to beginning development further on in the process.

The most important part of the project is ensuring that all stakeholders are happy with the final result. These stakeholders are the customer, University of York Communications Office and the end user themselves. Each of these has a different interest in the project and what the requirements of the game are for them. Throughout the process the needs of each of these stakeholders have been closely considered when eliciting each of our requirements. By considering the needs of the stakeholders early on, it allows us as a team to know exactly what we need to prioritise in the development of the game and what aspects are not needed or at least of a lesser importance.

We have split our requirements into three categories: User Requirements, Functional Requirements and Non-Functional Requirements. User Requirements are those the user will find important such as UI elements; Functional Requirements are those that form part of the core element of the game with Non Functional Requirements being those that are not entirely necessary but will improve the system for the end user. Then each requirement was given a priority based on the brief and the interviews with the customer. These priorities were based on how important it is to each stakeholder for that given feature to be implemented during the development of the game. This will allow us to better delegate our time in development allowing a more focused approach which will allow a more efficient and elegant solution.

User Requirements

ID	Description	Priority
UR_UX	The system shall offer a pleasant user experience. The game should be enjoyable by SEPR cohort and the customers.	Shall
UR_DEVICE	The system should be able to run on both PC and mobile to allow a larger number of people to play the game	Should
UR_DEMO_MODE	While no one is playing the game, it should automatically change to a demo mode where the game is run by itself using video or AI.	Should
UR_RESUME	The system should allow the user to pause and resume the game.	Should
UR_FORTRESSES	There should be a number of unique fortresses in the game based on real locations in York.	Shall
UR_FIRE_ENGINES	The game should have a number of unique Fire Engines on the map, which users must use to destroy the enemy	Shall
UR_SINGLE_PLAYER	The game shall be played by a single player.	Shall
UR_ET_PATROLS	There should be a number of ET patrols on the map which can attack both the Fire Engines and Fire Stations	Shall
UR_GAME_OVER	The game should end in a clear and logical way	Shall
UR_COMPARE_SCORE	The user should be able to compare their scores with others.	Should
UR_MINI_GAME	There should be an embedded mini-game, completely different in style from the main game, but aligned to the theme of the main game.	Shall
UR_WRITTEN_IN_JAVA	Use Java as programming language.	Shall

Functional Requirements

ID	Description	User Reqs
FR_CHANGE_GAME_MODES	Allow the user to change between game modes with limited steps.	UR_DEMO_MODE
FR_RESUME	There should be a resume button on the game screen that allows users to pause and resume their game.	UR_RESUME

FR_NUM_OF_ENGINES	There should be at least four Fire Engines.	UR_FIRE_ENGINES
FR_CONTROL_ENGINE	The user should be able to control the engines.	UR_FIRE_ENGINES
FR_ENGINE_SPEC_WATER	Each Fire Engine must have a unique spec in terms of the volume of water it can carry.	UR_FIRE_ENGINES
FR_ENGINE_SPEC_SPEED	Each Fire Engine must have a unique spec in terms of its speed.	UR_FIRE_ENGINES
FR_ENGINE_SPEC_RANGE	Each Fire Engine must have a unique spec in terms of the range.	UR_FIRE_ENGINES
FR_ENGINE_SPEC_DELIVERY_RATE	Each Fire Engine must have a unique spec in terms of the delivery rate of its water cannon.	UR_FIRE_ENGINES
FR_ENGINE_SPEC_DAMAGE	Each Fire Engine must have a unique spec in terms of the amount of damage it can take before it is completely destroyed.	UR_FIRE_ENGINES
FR_ENGINE_MAINTANANCE	Allow Fire Engines to be repaired and refilled at the Fire Station.	UR_FIRE_ENGINES
FR_CONTROL_ENGINE_MOVING	Fire Engines should move between the Fire Station and the ET fortresses.	UR_FIRE_ENGINES
FR_CONTROL_ENGINE_AVOID	Fire Engines avoid ET patrols on the way.	UR_FIRE_ENGINES
FR_ENGINE_DESTROYED	When a Fire Engine's health reaches 0, it is destroyed and can no longer be used	UR_FIRE_ENGINES
FR_ENGINE_IN_RANGE	Fire Engines must only be able to shoot when an ET is within its defined range.	UR_FIRE_ENGINES
FR_FORTRESS_SPEC_WATER	Each ET fortress must have a unique spec in terms of the volume of water it takes to flood.	UR_FORTRESSES
FR_FORTRESS_SPEC_RANGE	Each ET fortress must have a unique spec in terms of the range of its defensive weapons.	UR_FORTRESSES
FR_FORTRESS_SPEC_HEALTH	Each ET fortress must increase in health after a fixed amount of time in order to tip the balance towards the Aliens throughout game play	UR_FORTRESSES
FR_FORTRESS_SPEC_DAMAGE	Each ET fortress must have a unique spec in terms of the amount of damage these weapons can deal to Fire Engines over a period of time.	UR_FORTRESSES

FR_FORTRESS_DESTROYED	When an ET fortress reaches its water capacity, it is destroyed and is no longer able to use its defensive weapons	UR_FORTRESSES
FR_FORTRESS_IN_RANGE	Each ET fortress must only be able to shoot the Fire Engines when they are within its defined range.	UR_FORTRESSES
FR_ET_EVAPORATE	ETs evaporate when they come in contact with water.	UR_ET_PATROLS
FR_ET_SPEC_SPEED	Each ET must have a defined speed.	UR_ET_PATROLS
FR_NUM_OF_ET	There will be a maximum number of ETs allowed onto the map at any given time	UR_ET_PATROLS
FR_ET_RESPAWN_TIME	ETs should respawn a set amount of time after they have been destroyed.	UR_ET_PATROLS
FR_ET_ATTACK	ETs attack fire engines when the defensive weapons' canons are within shooting range.	UR_ET_PATROLS
FR_ET_FIND_OUT_ROUTE	After a fixed amount of time following the first attack to an ET fortress, ETs should be able to figure out where the Fire Engines are coming from.	UR_ET_PATROLS
FR_ET_DESTROY_STATION	Once ETs figure out where the Fire Engines are coming from, ETs will destroy the Fire Station. And from that point onwards, Fire Engines cannot be repaired or refilled.	UR_ET_PATROLS
FR_MINI_GAME_BEGIN	The mini-game will be played when an ET patrol fires at a Fire Engine	UR_MINI_GAME
FR_MINI_GAME_LOSE	If the mini-game is lost then the Fire Engine will suffer the damage given by the ET patrol	UR_MINI_GAME
FR_GAME_WIN	The game is won when all ET fortresses have been flooded.	UR_GAME_OVER
FR_GAME_LOSE	The game is lost when all Fire Engines have been destroyed.	UR_GAME_OVER
FR_NOTICE_GAME_OVER	The user should be able to know whether they have won or lost the game	UR_GAME_OVER

Non-functional Requirements

ID	Description	User Requirements	Fit Criteria
NFR_TIMING	Win or lose icon will appear in the game screen in a short period of time.	UR_UX	Display in < 2 seconds after winning or losing the game.
NFR_OPERABILITY	The game shall be playable by users that have not received game instructions.	UR_SINGLE_PLAYER	The users are able to play the game at their 1st time.
NFR_GAME_TIME	The game shall not last for a long time in order to ensure that enough people can play the game while still being a complete demonstration of implemented features.	UR_SINGLE_PLAYER	The game shall last between 3-10 minutes.
NFR_DOCUMENTATION	The code should be understandable so that when changing code bases, it is easy to know what is going on.	UR_WRITTEN_IN_JAVA	All code must be well documented and commented where appropriate.