Requirements

SEPR Team: Berbils

Module	SEPR
Year	2019/20
Assessment	1
Team	Berbils
Members	Dylan Henley-Marshall Edward Demkowicz-Duffy Emily Wisher Samkeliso Kimbinyi Joshua Waha Peter Robinson
Deliverable	Req1

Introduction

Requirement generation can be broken down into 5 steps. Initially, the stakeholder comes forth with a product. This product can then be analysed allowing for questions to be created that are used to elicit stakeholder needs. From the stakeholder needs, product requirements can be generated. These requirements can then be validated by the stakeholder to be accurate in encompassing the full product. If these requirements do not meet the stakeholder needs, they must be developed until they meet the criteria set out by the stakeholder.

To acquire stakeholder needs, interview questions must be developed that will capture all the intentions of the stakeholder for the product. The questions for this product were split into four main sections being Architecture, Visuals, Mechanics and Minigames (questions and answers are on the website). The answers to these questions showed the needs of the stakeholder. These needs can then be prioritized and then the requirements can be generated.

When generating requirements, the constraints on the system must be identified first in order to establish the feasibility of all the requirements generated. The constraints on the product are created through stakeholder needs. Some of these needs will explain the operational scenario and environment of the system, which then can be transformed into constraints.

The Use Cases allow us to easily create requirements based on the intended application of the system. It works by listing all the actors involved and then a list of all the steps they'd need to take to complete a task. It includes context on the event, what happens on success and all side steps that can be taken. This allows us to intuitively see what requirements are necessary to complete each step and goals of the Use Case. All other requirements were elicited through stakeholder needs.

Following the IEEE documentation (IEEE 29148 (on website)), the requirements must conform to the following rules:

- Requirements should be necessary
- Requirements should be appropriate
- Requirements should be unambiguous
- Requirements should be feasible
- Requirements should be verifiable
- Requirements should describe a singular characteristic
- Requirements must be an accurate representation of the entity need from the stakeholder
- Requirements should be consistent with each other in terminology
- No external information should be needed to understand the requirement

All changes to requirements throughout the life-cycle of the project must be confirmed by group member vote/stakeholder interaction In order to prevent the requirements distorting to too large a degree (i.e. “requirements creep”). If requirements change too much, the quality of the product will decrease.

Statement of Requirements

User Requirements

ID	**Description**	**Priority**
UR_Smooth	Should run smoothly	Should
UR_Consequence	Users actions should have visible consequences	Shall
UR_Accessible	The Game should be accessible to many users	May
UR_Player	User should be able to use the player entity easily	Shall
UR_Status	Users should be able to see their progress	May
UR_Pause	The Game should not force users to play	Should
UR_Time	The Game should be suitably timed	Shall
UR_Rules	The Game shouldn’t allow the rules to be broken	Shall
UR_Progress	As the user plays the game should progress	Shall

Functional Requirements

ID	**Description**	**User Requirements**
FR_Menu	“Customer pressing escape” shall lead to “game pause/menu screen” during gameplay	UR_Pause
FR_Move	“Customer pressing arrow keys or ‘WASD’ keys” shall lead to “Fire engine moving” during gameplay	UR_Player
FR_Score	The System shall always list the user current score during gameplay	UR_Status
FR_Pause	The System shall always allow for the game to be paused	UR_Pause
FR_ Difficulty	The System shall always make the game will get more difficult in a linear fashion, more alien patrols and harder fortresses	UR_Progress
FR_Quit	The System shall always allow the user to quit the game at any point	UR_Pause
FR_ Restricted	The System shall never allow the user to move onto restricted tiles during gameplay	UR_Rules

FR_Win	The System shall allow the user to reach the winning state	UR_Progress
FR_Lose	The System shall allow the user to reach a losing state	UR_Progress
FR_ Minigame	Triggered when the fire engine is in sight of alien patrol for a certain amount of time	UR_Progress
FR_Attack	When the attack key is pressed the fire engine will start the attack sequence	UR_Progress
FR_Destroy	The fire station is destroyed 8mins after the first fortress is destroyed, taking away the ability to repair fire engines	UR_Progress
FR_Change	When entering the fire station the user can change and repair/refill fire engines	UR_Player

Non-Functional Requirements

ID	**Description**	**User Requirements**	**Fit Criteria**
NFR_Length	The Game length should be suitable for Open day	UR_Time	The Game lasts 10-15 minutes
NFR_FPS	The Game's framerate should run smoothly and not be tied to game mechanics	UR_Smooth	Runs consistently above 30 FPS
NFR_Input	The Game should respond quickly to inputs	UR_Smooth	Input lag should be no more than 100 ms
NFR_Uptime	The Game must be operational for the whole open-day	UR_Time	The Game should not crash, uptime 99%
NFR_Loading	Time waiting during gameplay should be low	UR_Time	Start time and loading screens should only last a maximum of 5 seconds
NFR_Debug	Include a debug console	UR_Rules	Log of system operations should be kept and accessible
NFR_Demo	The Game should include demo	UR_Player	Games should play itself without input from user in demo mode
NFR_Tutorial	The Game should teach users how to play	UR_Accessible	Short screen explaining controls

NFR_Colour_ Blindness	Accessible for people with colour-blindness	UR_Accessible	Has colour-blind option which changes the colours to colour-blind friendly colours
NFR_Text	Text should be readable	UR_Accessible	All text should be English and over font size 12
NFR_Screen	Should fit on most standard laptop screens	UR_Accessible	Fits on screen sizes from 13-28 inches
NFR_ Operating	Should run on any system regardless of operating software	UR_Accessible	The Game can be run on Windows, Mac and Linux
NFR_Controls	Controls should be intuitive	UR_Player	Follow conventional controls (WASD)
NFR_ Objectives	Objectives should be intuitive	UR_Player	Follow conventions (alien base colour red)
NFR_ Highscore	Highscores should be documented and shown	UR_Status	Game end score is recorded then shown
NFR_Memory	The Game should be small enough for small harddrive spaces	UR_Accessible	The Game should be no bigger than 1GB

Use Cases

Name: Play Game
Context: Play the game at a computer science open day, on laptop
Primary Actor: Prospective student
Precondition: Game has loaded and is running demo screen
Trigger: User presses play which starts the game
Main Success Scenario
1. User takes control of Fire Engine and traverses map of York
2. User attack Alien bases
3. User destroys Alien base, returns to step 2
4. After all bases are destroyed User wins
5. Enter name for recording score

Secondary Scenarios
1.1. User enters sight of Alien Patrol and minigame begins
1.2. User completes minigame and loses health proportional to how they did
2.1. User enters Fire Station
2.2 This allows user to repair/refill Fire Engine and change the type of Fire Engine
2.3 If the fire station is destroyed it can’t be used to repair

Success Postcondition: Top high score are shown for a suitable amount of time and then game returns to demo screen. User moves away