REQUIREMENTS

NP Studios – Team 8

Team Members

Lucy Ivatt, Jordan Spooner, Alasdair Pilmore-Bedford, Matthew Gilmore,
Bruno Davies, Cassandra Lillystone

Introduction

Kroy is a single-player 2D game where the user must control fire-engines and defeat the aliens which have invaded the city of York. The game will be structured using levels which increase in difficulty as the user progresses. Each level will have a time limit in which the player must have destroyed the alien compound otherwise the aliens will destroy the users main base in retaliation.

Single Statement of User Need

The game must be based on the city of York to help engage our target audience of prospective students attending University of York open days. As well as this, the game must be fun, engaging and have a strong replay value due to its additional purpose of being sellable to customers.

Stakeholders

Our main client and direct, primary stakeholder for this project is Professor Dimitris Kolovos, a lecturer at the University of York. He will be our primary point of contact regarding what the outcome of the projects needs to be. The focus of this stakeholder is that we produce a marketable game at the end of the project. The University of York Communications Office is also one of our stakeholders. Their aim for this project is that the final product must help them advertise the university during open days and UCAS days, and therefore the game must have obvious ties to the city itself.

Requirement Gathering

To gather the most accurate requirements for this project, we will be in contact with the client, Dimitris, consistently throughout the full project life cycle. This contact will occur primarily through both email and in-person interviews.

To begin the requirements gathering process, the client brief was analysed and both important and vague areas were highlighted. This ensured we included critical requirements as well as making sure we knew what areas we needed to discuss both internally and with the client. We then created an overall plan for the game while keeping our target audience in mind and organised a meeting [Figure 1: Initial Meeting Email] to pitch this idea to the client. During this meeting we discussed our overall ideas [Figure 2: Initial Client Interview] and ensured there were no additional requests or objections before moving forward. Examples of additional requirements and clarification we received include that the total game time must last for approximately 10-15 minutes, the overall game and mini-game direction agreed upon as long as it's not repetitive and the use of the game at open days is the projects primary focus with selling being the secondary.

During the production of the formal requirements, we made sure to use memorable, unique and relevant ID tags [1] as this would be beneficial when referring to them elsewhere in documentation and during both team and client discussion. For the writing of the requirements, we decided to use EARS [2] format. This is a proven syntax to create unambiguous and concise but detailed requirements while allowing us to have a consistent format throughout the team.

These requirements were sorted into the three tables: User Requirements, Functional Requirements and Non-Functional Requirements. We believed this was a clear and efficient way to display them. We used a slightly modified version of the IEEE 'organised by feature' SRS format [3]. We made sure that each user requirement has a functional or non-functional requirement linked with it, so that every requirement can be traced back to fulfilling a user's need and therefore has a direct purpose within the game. Finally, each requirement has a priority associated with it using the MoSCoW classification technique. We decided on this as it is a quick and easy to understand format which can deal easily with additional requirements [4] and therefore perfect for the agile process we will be using.

User Requirements

M – Must Have S – Should have C – Could have W – Won't have

ID	Description	Notes
UR_ start_screen	User shall select four options from the welcome screen (Start, Options, Credits, Quit) (M).	
UR_save_ load_quit	An option should be available to 'save and quit' the game upon entering the fire station. Any saved games should be able to be resumed by a user. (M)	Close the game application to quit without saving.
UR_select_ level	User shall open the map in the fire station and choose a level to play. (M)	Alternative: Complete games sequentially, restart when last level complete.
UR_pause	User shall pause the game and bring up a menu with options (Resume, Restart, Save, Quit) (S)	
UR_ minigame	User shall refill water by completing the increasingly challenging and engaging minigame. (M)	The mini game can't be too difficult as this would take away from the main game.
UR_instruct_ engines	User shall select and instruct their fire engines - attack the enemy and move around. (M)	
UR_seeHUD	User shall see a HUD showing health, water and mini map (C)	Alternative: in game visual
UR_victory_ screen	User shall be notified when they complete a level. (M)	Alternative: automatically taken back to the fire station.
UR_refill_ warning	User shall be notified when they are close to needing to refill or repair. (S)	Notification will not hinder the user's ability to see enemies
UR_strategy	User shall strategise how to manage and deploy their fire engines. (M)	
UR_attack_ warning	User shall be notified when the fire station is about to be destroyed. (M)	
UR_interest	User shall be drawn in by the game, and not be bored. (M)	
UR_fresh_ health	User shall be able to start each level with full health (C)	Alternative: they don't and their health level continues on after each level.
UR_ease	User shall be able to understand the game - be able to finish without being confused by mechanics. (M)	

Functional Requirements

ID	Description and *User* Requirement it Links to
FR_display_timer	System shall display a timer - countdown until aliens destroy the fire station. Show when timer is over. ***UR_attack_warning***
FR_auto_save	Between levels user's progress is recorded and data shall be saved locally on system. ***UR_save_load_quit***
FR_pause_inlevel	System shall be paused during play. This stops all movement of patrols and the timer. Menu pops up with options. ***UR_pause***
FR_save_quit	System shall be able to save user progress when the quit button is pressed. ***UR_save_load_ quit***
FR_auto_repair	Between levels damaged fire engines health are restored back to full automatically. ***UR_fresh_health***
FR_unique_engines	Fire engines have unique spec - Volume of water, speed, range, delivery rate, max health. ***UR_strategy***
FR_unique_enemy	Enemies have unique spec - defensive weapons, weapon damage, volume of water needed to flood. ***UR_strategy***
FR_level_gimmicks	Levels shall be different to each other and intriguing. ***UR_interest***
FR_enemies_die	Enemies shall evaporate when they come in contact with water. ***UR_instruct_ engines***
FR_engine_destroyed	System shall notify the user when their fire engine is destroyed. ***UR_seeHUD***
FR_6_levels	The game shall include 6 levels of increasing difficulty. ***UR_select_level***
FR_end_game	User wins if they flood all enemy bases and complete the final level. User loses if all their fire engines are destroyed. ***UR_ease***
FR_new_level	The system shall take the user back to the fire station when they have completed a level. ***UR_victory_Screen***
FR_open_Minigame	When user reaches the refill tile the system shall start the minigame. **UR_ minigame**

Non-Functional Requirements

ID	Description	Fit Criteria
NFR_user_ instructions	Instructions for the game should be available to the user. ***UR_ease***	

NFR_readability	Users shall be able to read any text easily. ***UR_ease***	Text legible from 5 metres away.
NFR_game_ understanding	The user should understand the games focus and how to beat the game from the tutorial. ***UR_ease***	
NFR_menu_ accessibility	Users shall be able to navigate the menu without any prior experience. ***UR_ease, UR_startscreen***	
NFR_save	System shall clearly outline to the user all the possible methods of saving. ***UR_save_load_quit***	
NFR_artwork	Colour scheme and artwork should be fun and engaging. It shouldn't hinder the users understanding of the game. ***UR_interest***	
NFR_user_ interactions	User interactions with the game should be instant. No delay. ***UR_instruct_engines***	User actions displayed within <1 second.
NFR_error_ prone	System shall not be broken by cheats or glitches. ***UR_ease***	
NFR_main_ focus	Mini game shouldn't distract the user from the main objective of the game. ***UR_minigame***	
NFR_ingame_ warning	Warnings directed towards the user should be easy to understand and read. ***UR_attack_warning, UR_refill_ warning***	
NFR_buttons	All buttons should be labelled and have a known purpose to the user and be easily accessible. *UR_ease*	Labels should be legible from 5 metres away.
NFR_timer	User should be made aware when they have limited time left in the game. ***UR_attack_warning***	Accuracy of timer should be +/- 0.25 seconds
NFR_security	System should be able to store saved progress accurately. ***UR_save_load_quit***	
NFR_precision	Fire trucks positions after being moved should have a small error margin. ***UR_instruct_engines***	Margin of error less than 1%
NFR_operators	System shall be operable by users with any level of gaming experience. ***UR_ease***	
NFR_audit	System shall keep saved files. ***UR_save_load_quit***	Keep these files for a day at least.
NFR_resume_ time	System should load a saved game quickly. ***UR_save_load_quit***	Acceptable time of 5 seconds

References

[1] QRA Team, “21 Top Engineering Tips for Writing an Exceptionally Clear Requirements Document,” 31 October 2019. [Online]. Available: https://qracorp.com/write-clear-requirements-document/.

[2] A. Mavin, P. Wilkinson, A. Harwood and M. Novak, “Easy Approach to Requirements Syntax (EARS),” in *2009 17th IEEE International Requirements Engineering Conference*, Atlanta, GA, USA, 2009.

[3] IEEE, “IEEE Recommended Practice for Software Requirements Specifications,” 9 December 2009. [Online]. Available: http://www.cse.msu.edu/~cse870/IEEEXplore-SRS-template.pdf. [Accessed 2019 November 1].

[4] S. Hatton, “Choosing the Right Prioritisation Method,” in *19th Australian Conference on Software Engineering (aswec 2008)*, Perth, WA, Australia, 2008.

Appendix

Figure 1: Initial Meeting Email

Dear Professor Kolovos,

We are a current SEPR group, NP Studios, who have a practical this Friday (Cohort 1).

We would like to have a stakeholder meeting, and we were wondering if this is something we can do during the practical this Friday? Or, if it would be better to arrange for early next week?

As a group we have some ideas we would like to pitch to you and receive some feedback and input. This would help us to move forward to finalising our statement of requirements and progress to the next stage.

Thank you in advance for your correspondence.

Kind Regards,
Bruno Davies - NP Studios

Figure 2: Initial Client Interview

Main Points from Client During Initial Interview:

Main Pitch:

- Needs to be eye catching and keep the prospective student target audience in mind
- **Suggested Level Based System:** Was originally thinking open world and not level based, but is open to other ideas as long as it's not repetitive → Add an element of randomisation to each level → Different patrol pathways and number of patrols
- Primary focus would be open days and not selling *as much*.

Minigame Notes:

Client preferred the water pipes puzzle idea (which occurs when the users fire engine needs repairing) compared to the jigsaw and space invaders ideas.

Fire Engine Specification:

Confirmed we can use more than four fire trucks, they must be balanced against the number of fortresses.

Particular Locations:
The client was asked if they had any particular locations in mind but said that we could choose the locations.

Length of Game:
Between 10/15 minutes