**Module** SEPR

**Year** 2019/20

**Assessmen t**

**Team** Salt N Sepr

**Members** Alberto Lee, Archie Godfrey, Jacqueline Eilertsen, Jake Schoonbrood, Joshua Levett, Matteo Barberis

**Deliverable** Requirements (Req1_update)

# Introduction

## Single Statement of Need

**The client, from the University of York, intends to demonstrate through the use of developed game the scope of programming and design skills of the Computer Science department to prospective students and their parents ("users") on departmental and university Open Days, Post-Offer Visit Days and similar such events. The game, named *Kroy*, should be a virtual representation of the city of York, and the objective should be for the users to defend against an alien invasion of the city by spraying water on the aliens and their fortresses, set up at major landmarks around York. The game should be easy for the users to understand and play, and completable in short space of time.**

## Collecting requirements

**From the broad overview of the game, given by the client,** we came together as a group to discuss different aspects of the games and the requirements for each class and general game assets. We broke the requirements into two large sections, visuals and functionality.

In the functionality section, we broke down each unit in the game. For each unit we looked at what it should do and how it should interact with other units. We researched the IEEE requirements standard [1] in order to present our findings effectively, illustrated below in each of the tables. **The IEEE requirements standard has been used to design and populate the tables of requirements, with the necessary information given in the different tables and use cases.** We then sent this information to the client so that we could schedule a meeting and clarify that we had a good understanding of the requirements of the game and discuss anything we had missed.

After meeting with the client, we learnt that there would be little to no sound on the Open Days the game would be played on. This means that the visuals used should be a large focus as they will be one of the main features that differentiate our game from other student's games. As well as this, prospective students will have a limited amount of time to play the game therefore it should be designed to finish in a short amount of time, 5-10 minutes.

The main focus of the software should be to run on a computer, however considerations should be made so that it can be easily ported to mobile devices. The main difference between mobile and PC is the way the user will interact with the system, using a keyboard rather than a touchscreen. Therefore, the controls cannot be overly complex on PC in order to simplify the transition. Another reason for simple controls relates to the target audience. Not all students, and their parents, will be familiar with PC gaming. Therefore, the controls must be easy and quick to learn.

**Furthermore, several design decisions were included in the early process for an easier transition within functions needed. These can be found under functional requirements.**

## User Requirements [2]

**ID Description Priority**

e water the fire truck sprays SHALL

CONTROL_TRUC K

ection the fire truck travels in SHALL

CONTROL_SPRA Y

RETURN_HOME Return a firetruck to the fire station to repair and refill it SHALL

VARIED_TRUCKS Play as 4 different fire trucks SHALL

GAIN_INCOME The user should earn money/points from destroying

aliens and/or their fortresses

SHALL

CREATE_MAP The user should be able to explore a

controlling the firetruck

SHALL

WIN_GAME Once the user has destroyed all 6 different fortresses

they win the game **(by DESTROY_ENTITIES)**

SHALL

CREATE_ENTITIE S

er alien patrols throughout

SHALL

SHALL to destroy alien patrols and
em with water

SHALL
SHALL

DESTROY_ENTITI
ES

NO_VIOLENCE There will be no violence to appeal to target audience SHALL

the game

OPEN_SHOP The user should be able to find out how much different

SHALL

fire trucks cost at any point in the game

MAY

LEADERBOARD See a local leaderboard to compare sco

BUY_ITEM The user should be able to buy different fire trucks

other players

from a shop

MAY

MAY

## Functional Requirements [2]

MENU There should be a menu so that the user can access

the leaderboard, the minigame, edit settings or play

**ID Description User Requirements**

AY_FUNC

cannon will be

CONTROL_TRU
CK_FUNC

**uses the controls,** the fire
n the appropriate direction

CONTROL_SPRAY
CONTROL_SPRAY

CONTROL_TRUCK
CONTROL_TRUCK

CONTROL_SPR

When RETURN_HOME
the firetruck returns to the firestation
RETURN_HOME _FUNC
it will repair and refill over a defined amount of time
FIXED_TIME After a fixed amount of time the user cannot
repair their fire truck at the fire station **(fixed time is 3 min)**
RETURN_HOME
NO_HEAL Aliens should not heal after taking damage DESTROY_ENTITIES
FORTRESS_HE AL
DESTROY_ENTITIES
DESTROY_ENTI TIES_FUNC

Alien fortresses should heal over a duration after taking damage. The more damage the longer it takes to heal
DESTROY_ENTITIES
VARIED_TRUCK S_FUNC
If the alien patrol or fortress runs out of health, it should be removed from view
VARIED_TRUCKS
CREATE_MAP_ FUNC
Able to swap between trucks **using a single action. Different trucks have different spray distances, damage tolerance, recovery speed and acceleration.**
CREATE_MAP
OPEN_SHOP_F UNC
A section of the map should be displayed to the user so that they can navigate it
OPEN_SHOP
BUY_ITEM_FUN C
The user should be able to press a button and it then opens a shop GUI that allows the user to upgrade or buy new fire trucks.
BUY_ITEM
**GAIN_INCOME_ FUNC**
When the user tries to buy an item, it should compare the value of the item to the balance and if the user has enough, add the item to his inventory for use.
**GAIN_INCOME**
**WIN_GAME_FU NC**
**When patrol or fortress is hit the user will collect money/points that will benefit them.**
**WIN_GAME**
**CREATE_ENTITI ES_FUNC**
**Destroy the fortresses and the user will win the game. the user name announced as winner and score displayed.**
**Should be able to spawn patrols**
**CREATE_ENTITIES randomly when the user has pressed play and increase with difficulty or time. entities should always act the same way every game.**
**MENU_FUNC For future assessment. MENU**
**LEADERBOARD Should display top 5 players. LEADERBOARD**
**_FUNC**

## Use Cases [2]

**Scenario ID**
Lose game Repair and refill
fire truck
**Primary Actor**
Destroy fortress Purchase item
from shop
Player of the game
**Pre- condition**

Player of the
Player of the
Player of the game
game
game
Player has water
Player is in the
Player has no
Player has in their tank and
shop and has
remaining fire
moved their fire fortress has
navigated to the
trucks after a fire
truck to the fire health
item they want to
truck is
station buy
destroyed
**Trigger** Player sprays
Player’s fire water at alien
truck is on top of swarm
the fire station
**Main Success Scenario**
Player clicks the
Player’s fire buy button
truck is destroyed
1) Player’s truck’s health is increased over time
**Secondary Scenarios**
1) Player sprays
1) Player clicks
1) Player takes at fortress
the buy button
damage from an 2) Fortress takes
2) They have
alien damage
enough money
2) Player’s 3) Fortress’
for the item
health reaches 0 health reaches 0
1) Player moves away from fire station so repairing stops
**Success Post- condition**
1) The Player

1) The Player
1) Player stops stops spraying
does not have
taking damage before the
enough money.
before their fortress’ health
Purchase is
health reaches reaches 0. The
cancelled and
0. Game fortress then
the Player is told
continues begins to heal
why 2) The Player runs out of water before the fortress’ health reaches 0. The fortress then begins to heal
Player’s fire truck’s health reaches its full value
**Requireme nts**
The fortress
Player receives
End game disappears from
the item
screen is shown the scene
to player
**DESTROY_ENT**
**BUY_ITEM WIN_GAME RETURN_HOM ITIES**
**E_FUNC**
**DESTROY_ENT (user/funct**
**BUT_ITEM_FUN ional)**
**ITIES_FUNC**
**C**

## Non-Functional Requirements [2]

**ID Description User**
**Requirements**
**Fit criteria**
TIME_ACCES SIBILITY
FIXED_TIME The game is
**completable** within 5 minutes due to limited time on open days
GAME_DOCU MENTATION
After a fixed amount of time the user will no longer be able to repair fire trucks, therefore the game will always end.
It should be easy to
WIN_GAME **With a single game,** understand that the game is
**the user should** won by destroying all 6 alien

**understand the** fortresses. This can be done
**objective of the** by a small tutorial
**game. (No advanced setting)**
RESILIENCE The game should only be won when all 6 alien fortresses are destroyed
WIN_GAME If the game is won
when exactly 6 fortresses are gone
AUDIENCE_A CCESSIBILITY
The game should be appropriate for prospective students and their parents
GAME_ACCES SIBILITY
Instead of showing violence,
NO_VIOLENC the enemies will just
E disappear in order to satisfy the target audience.
The system must have a
MENU The game must have a menu so that the user can
minigame and access the main game and
therefore the user must the minigame.
be able to access it. **This will be accessible from the main menu.**
OPERABILITY The game should be playable
on a PC but considerations should be made for mobile versions in the future.
Users will play the game on a PC on open day
SECURITY The game should not ask for
any sensitive information when displaying scores on the leaderboard. Instead, a nickname should be used
CONTROL_TR UCK
LEADERBOAR
The leader board will D
be displayed to lots of people and sensitive information should not be shared. **Username and score will be displayed**

## Updates and changes

- Updated the *Single Statement of Need (SSON)* to clarify the relationship between the client, the end-users/audience, and an overview of the requirements of the game, in response to feedback given on the *SSON* in Assessment 1. - Revised the initial discussion on requirements, such as removing irrelevant information about asset discovery, in an effort to focus more on the game requirements rather than meeting these requirements. This is also a revision based on the feedback given in Assessment 1. - Clarified how the IEEE standard has been used to inspire the design and information within the requirements tables, in response to the feedback given in Assessment 1. - Removed design decisions incorrectly listed as requirements in requirement descriptions. - Added additional functional requirements where some user requirements had no associated functional requirements. This is important for the implementation of user requirements. - Refined existing functional requirements where relevant examples were not given. - Provided a reference to user and functional requirements for use cases, allowing for more transparent traceability. - Clarified further the non-functional requirement fit criteria to enable them to be more binary (Pass/Fail) when assessed.

## References

[1] "29148-2011 - ISO/IEC/IEEE International Standard - Systems and software engineering -- Life cycle processes --Requirements engineering - IEEE Standard", Ieeexplore.ieee.org, 2011. [Online]. Available: https://ieeexplore.ieee.org/document/6146379. [Accessed: 01- Nov- 2019].

[2] "Lecture 2: Requirements Engineering", *York VLE*, 2019. [Online]. Available: https://vle.york.ac.uk/bbcswebdav/pid-3188304-dt-content-rid-8697295_2/courses/Y2019-006404/Requirements%281%29.pdf. [Accessed: 01- Nov- 2019].