Implementation Report

SEPRet Studios

Anthony Nel, Ben Silverman, Jake Billington, Leah Moss and Shouyi Yuan

Implementation of architecture and requirements

In this assessment, we build upon the previous team's (Mozzarella Bytes) concrete architecture from assessment 2 and implemented almost all of the requirements in order to create the full game.

While developing, we made sure we built upon the concrete architecture from the previous team. This meant that we coded with consistency and used what the previous team had done structurally to add the new features, which includes creating classes and adding assets to the correct folders and used clear, understandable names.

We did this by first assessing what we needed/wanted to change about the program of the project we picked. Next we assessed the project requirements to find what needed to be added to the program. We divided these into tasks so that each team member could work on a task(s) and so that we could complete the requirements for assessment 3 efficiently. You can see below what features we changed and what new features we added.

Significant new features

We implemented a range of new features to further satisfy the product brief and clients requirements. For some features we adapted and replicated pre-existing code, and for others we needed to add entirely new code to achieve new functionality. One requirement was to implement 2 further fire trucks to bring the total to 4. Given that the game included pre-existing fire trucks, to implement more we only needed to increase the array of instances of the pre-existing FireTruck class. Furthermore the previous program also included unique types for each fire truck, such that they are constructed with different maximum volumes of water, movement speed and range. The two new fire-trucks were given the names "Speed Truck" and "Ocean Truck", and the types were using the again pre-existing FireTruckType enumerated data type which contains all changeable data. The "Tank Truck" has 150 water reserve, 0.4 attack points, a speed of 1, range of 7, and a maximum health points of 250. The "Attack Truck" has properties of a max water reserve of 50, speed of 2, range of 6, attack points of 0.6 and maximum health points of 50.

In addition to the new fire trucks, we also added 3 more fortresses. For these fortresses new assets were made in the same style as the existing fortresses using the same tileset however the new fortresses represent other locations in York, namely the train station, the shambles and the York Minster. These fortresses also have varying fortress types implemented in the same way as the fire truck types. However given the space that the new number of fortresses take up we felt it was prevalent to expand the map from the original map provided to us. We again used the same tileset already used for the map while expanding to maintain continuity within the game. Using the old map and adding fortresses would mean that the range of the fortresses may cross over resulting in very difficult gameplay which directly conflicts with the requirement of simple intuitive gameplay put forth in the product brief.

Completely new implementation includes the introduction of ET patrols which consist of new ET sprites designed to the same box size as the fire truck sprites with similar themes as the surrounding map. The patrols travel on a set of hardcoded paths which can be added by

setting a list of corner tiles and the program creates a full path from this set. In the same way as the fire trucks, as the ETs turn they are redrawn with a new sprite relative to the way they are facing. All 4 direction sprites are saved within the asset folder of the program alongside the fire engine sprites. The patrols also attack using the same attack methods of the fortresses however doing less damage than the fortresses. The fire engines are also able to attack the ET patrols as they do with the fortresses. The collision handling of fire trucks hitting fire trucks is also used for ET patrols hitting other patrols or fire trucks. There are faces attached to patrols that show the user how the aliens are reacting to the state of the game. The faces begin as happy, while the timer is at its maximum, after the user attacks any fortress and the timer reduces by a quarter the faces change from happy to confused, then when the time reduces by the same interval (to reach half of the starting time) to become annoyance. After the time reduces to under its final quarter the faces change to anger. The faces stay as angry until after the mothership destroys the fire station, after this the faces become very happy. This emotional scale helps to show the user the passing of time in clear stages and the more worrying state to the user of the fire station being destroyed.

Fortresses also increase in power as time goes on, they progress in levels, and all fortresses will increase in level after a certain time has passed where their attack points increase and their health points increase to make destroying them more difficult as time passes.

The minigame has also been implemented in the game which is triggered on an entirely new game screen on the event that 3 fortresses have been destroyed by the player. The mini game, being on a new screen pauses the state of the main game when it is triggered and allows the user to return to the main game after the minigame has been completed. The minigame has a separate package within the program containing modifications on the classes “FireTruck”, “ET”, “Entity” and the new class “Droplet”. These classes are all brought together and shown upon the “MinigameScreen” class. The minigame functions by moving a fire truck left and right on the bottom of the screen and firing a droplet upwards at oncoming ETs as they move down towards the bottom of the screen.

The timer was also implemented, as stated in the product brief there must be 15 minutes of gameplay, starting from the moment that the user first attacks any fortress. After 15 minutes passes the ETs destroy the fire station. The clock is displayed at the top of the game screen, and when the clock reaches 0:00 a new alien sprite will spawn at the top of the screen and will attack the fire station with large blue blasts. The sprite will slowly move towards the fire station and eventually destroy it and any fire trucks within the station. The player can still continue and the game will only end after all the fire trucks have been destroyed or all fortresses have been destroyed.

Significant changes to previous software

Some changes that we made to the previous software that do not include new features and additions are: changed and shortened the splash screen at the start of the game, extended the map, changed the initial specifications for fire trucks and fortresses and moved the code for ET attacks to a separate class.

The first thing the user will see when they run the game is the initial splash screen which contains the logo of the team(s) that have developed the game. We had to change this in two ways. The first was a simple addition of our own logo before the logo of Mozzarella Bytes since we took over to complete the development. The second change we made was to make the time that these screens are displayed shorter. We did this in order to make sure the game started more quickly and did not leave the user waiting for too long. This therefore satisfies the requirement UR_ENJOYABILITY which states: *"The game must be enjoyable to play by prospective students and their guardians."*

Another change we had to make was to extend the map. This was a quick decision we made before implementing any additional features for assessment 3 since we knew we would have to add another three ET fortresses. We did this by using the software Tiled to extend from a 40x24 map to a 50x30 map and then we filled in the extra tiles using the same tileset to make sure the theme of the extended parts was the same as the original. This extended map allowed us to add more fortresses without having to worry about the range of fortresses overlapping and creating a game which was too difficult or confusing for the user. Therefore, this change again satisfies the requirement UR_ENJOYABILITY.

We also changed the parameters specified in the instantiation of the fire truck and the fortress objects. We did this for two reasons; we thought the game was too difficult and would therefore not satisfy the UR_ENJOYABILITY requirement and the game would've been made even harder by the requirement FR_FORTRESS_IMPROVE which states: *"ET fortresses improve over time"* and would again prevent the user from enjoying the game. To do this we reduced the health and attack power of the fortresses but also increased the attack power of the fire trucks.

A change we made which slightly alters our architecture was moving the code for fortress attacking to a new class called EnemyAttackHandler. The requirement UR_PATROLS which states: *"There should be at least 2 ET patrols that the user aims to avoid*" tell us that there will be ET patrols which attack fire engines. Since ET fortresses and patrols both attack fire engines we created this class which is much more efficient than duplicating the code in both the fortress and patrol classes.

Features not fully implemented

We have implemented almost all of the requirements for the full game, with all of the requirements with the shall (the highest) priority being completely implemented. Some requirements we did not meet were two of the patrol requirements; FR_PATROL_INCREASE which states: *"The number of patrols should increase throughout the game"* and FR_PATROL_SIGHT which states *"Patrols should chase fire trucks that are within their range of sight."* Having the number of patrols increase over time would have created a lot of added complexity to the code and took away from the time we had to implement the features with the highest priority which this one did not have (it had the should priority). The patrol sight again would have added complexity but it also would have made the game quite difficult and could have prevented us from meeting the UR_ENJOYABILITY requirement. Since this requirement was only a may priority (the lowest) we decided to not implement it at this stage.

Another requirement which we did not fully implement was FR_ANIMATION which states: *“The fortresses and fire trucks should change appearance as they are destroyed.”* The team which we took over also did not fully implement this feature but provided health bars and sound feedback to fire trucks which aids the appearance of the fire trucks being damaged but does not actually fully implement this requirement. We made no improvement to this since again we put our emphasis on implementing the highest priority features and saw that the priority was should and therefore not essential to the game.

Some supplementary features had been discussed to help in the gameplay of the program that had not been fully implemented. For example implementing dialogue may assist in the transitioning from the main game to the minigame, with some context to the user as to how the minigame relates to the context of the Kroy story. The dialogue would take the form of an image on the bottom margin of the screen containing a bordered text box with a sprite above it signifying who was speaking. The two characters proposed for this dialogue were a fire marshal and an ET overlord. The dialogue would allow the user to understand the importance of the effect of the minigame, i.e.: the ETs were reinforcing the fortress from an orbiting mothership and must be stopped. This also assists with intuitiveness of the gameplay. This dialogue may also be able to inform the user of the reasons behind an ET’s weakness to water and appraise the player of how ET fortresses may improve over time.

Another not yet implemented feature was the addition of some form of power-up or reward to occur in the event that the user successfully completes the minigame, such as increasing the abilities of the trucks, adding more time to the clocks, net damage to all the fortresses or so on. This reward for the minigame seemed spurious and the team felt that more time was needed to be assigned to extensive testing and fine-tuning than the inclusion of spurious additions that will no closer achieve any of the requirements.