Module	SEPR
Year	2019/20
Assessment	3 - Implementation Report
Team	Mozzarella Bytes
Members	Kathryn Dale, Emilien Bevierre, Ravinder Dosanjh, Elizabeth Hodges, Daniel Benison, Callum Marsden

Implementation report

Updated requirements

ID	Description	Change	Justification
VARIED_ TRUCKS_FUNC	Trucks should have a unique spray distance, damage tolerance, recovery speed, attack points, acceleration and water cannon size	Edited	Edited to cover all specifications mentioned in the product brief
VARIED_ FORTRESS_FUNC	Fortresses should have a unique attack range, attack points and health points	Added	Was not in requirements, but is in the product brief
MINI_GAME	There should be a minigame that is different in style but aligned to the theme of the main game	Added	Was not in the reuqirements, is in the product brief
MINI_GAME_ ACCESS	The minigame should be accessed from within the game	Added	Conversation with client, before it was accessed from menu
LEADER_BOARD	Removed as this was a may requirement and could not be met in the given time	Removed	This was confirmed with the client
MENU	There should be a menu where the user can view controls, play or quit the game	Edited	Removed accessing the leaderboard and minigame

Brief overview of additional features:

- *Two more firetrucks:* now the player can play as 4 trucks (VARIED_TRUCKS). To give the trucks a unique spray distance, damage tolerance, recovery speed, acceleration, attack points and water cannon (VARIED_TRUCK_FUNC) a *TruckType* enum was created in *Constants* (line 61) with each truck being given a *TruckType.*
- *Three more fortresses*: there are now 6 fortresses (RESILIENCE). To give the fortresses a unique range and amount health attack points a *FortressType* enum was created in *Constants* (line 42) with each *Fortress* being given a *FortressType*.
  Enumeration allow fortresses and firetrucks to have unique values that can be easily changed. This makes it easier to balance the game and add new types of firetrucks and fortresses.(See balancing report https://sepr-docs.firebaseapp.com/implementation)
- *10 patrols* (CREATE_ENTITIES): the patrols follow a random path around the map and can shoot and be shot at (DESTROY_ENTITIES) by firetrucks. See pg 2
- *4 Carparks:* highlighted in the game by green rectangles they are safe havens where the player can swap their active truck, use their score to purchase up to 3 new trucks and keep the trucks which they are not currently controlling safe from patrols. For implementation see pg. 3.
- *The fire station can be destroyed:* After three minutes (FIXED_TIME) the firestation becomes vulnerable to attack and can then be shot at by patrols. Once the firestation's health is at 0 it is destoyed and trucks can no longer repair and refill (FIXED_TIME). For implementation see pg 4.
- *Minigame (MINIGAME):* Accessed by driving over a minigame sprite; score from the minigame can be used in *CarParkScreen* to purchase new trucks (BUY_ITEM). For implementation see pg. 3.
- *Screens:* Added a pause, how to play, gameover, minigame, carpark and story screen. See pg. 4.
- *Tutorial:* before the game starts to explain the controls, the player can exit it at anytime.See pg.4
- *Arrow:* Points to the nearest fortress, is shown when the player presses space. See pg. 4

Changes we made to the existing software:

Collision detection: Originally if a truck's vertex collided with a wall the truck could get stuck in the games collision layer. As the collision detection was done with a rectangle the same size as the truck the truck could not be rotated out of the collision layer because if the truck tried to rotate a collision would occur between the edge of the truck and the collision layer. Thus we used a triangle (*movementHitBox*

in *SimpleSprite* see line 59) instead of a rectangle to detect collisions so that the edge of the truck would not collide with the collision layer when the truck is rotated.

Toggling between trucks: Initially the player could toggle between trucks by pressing tab but could only control and follow one truck at a time. We also felt this system removed the benefit of having more then one truck as the player likely to play as the same truck for the whole game. This system would also mean the player would be unaware if patrols were attacking an out of view truck. Thus we created carparks (see pg. 3) where the player can change their trucks and keep them safe from patrols.

Controls: Originally the controls for the game were in the update method in *Firetruck* which was called each frame. This worked when the player could only interact with the firetruck, however we also wanted the player to be able to exit the tutorial and pause the game. As these will not happen very often it is inefficient to poll for whether these inputs have been pressed each frame. Thus to be able to generate an interrupt if the key triggering these functionalities is pressed we added *GameInputHandler*. Due to the games architecture *Firetruck* extends *MovementSprite* which updates the truck's location each frame, because of this *MovementSprite* needs to poll for the trucks movement every frame. As *GameInputHandler* generates interrupts and *MovementSprite* needs information each frame the controls for the firetruck are still in update.

Camera: Initially the camera zoomed out automatically when the firetruck was moving and zoomed in when the firetruck was stationary. To allow the player to control the zoom to their preference we overrode the scrollMethod in GameInputHandler. To prevent the player from being able to see too much of the map when they zoomed out we added a vignetteSepiaShader (see GameScreen line 323) which applies a vignette shader to restrict the distance the player can see.

Changes we added to the software:

Addition of patrols: To implement the path finding for the patrols numerous LibGDX AI libraries and interfaces were used see diagram 2. See https://sepr-docs.firebaseapp.com/implementation for a full diagram.

Figure 1

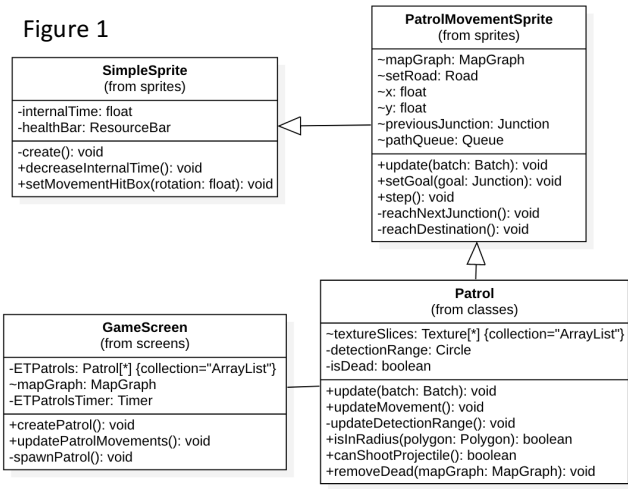
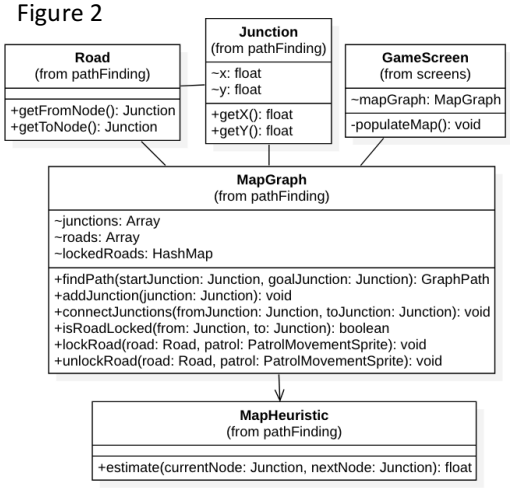


Figure 2



While there are less than 8 active patrols (stored in the arrayList *ETPatrols* see fig.1) a new patrol is spawned every 8 seconds. This is due to *ETPatrolsTimer*, a timer that calls *createPatrol* that calls *spawnPatrols* every 8 seconds which spawns a patrol if there are less than 8 patrols in *ETPatrols*.

Patrol inherits from PatrolMovementSprite (PMS) (see fig.2) which controls the movement of the Patrol. PMS makes use of mapGraph (an instance of MapGraph instatiated in GameScreen) to determine where the PMS should move to and the route it should take to get there. mapGraph contains the junctions of all the points where a patrol can make a decision as to which junction to go to next.

A patrol's spawn and destination junctions are randomly chosen from mapGraph. Using mapHeuristic, findPath in MapGraph uses a LibGDX A* path finding algorithm to generates a list of junctions the PMS has to take to reach it's destination; these are added to its pathQueue. The location of the PMS is updated each frame by the step function which calls NextJunction if the PMS has reached a junction. If

this junction is the patrols final destination or the road that the patrol wants to travel on next has a patrol on it, finalDestination is called giving the patrol a new route to a random junction; if the road is free the PMS starts to step down it towards its next junction.

A hashmap, *lockedRoads* in *MapGraph* (see fig.2), is used to keep track of which roads have which patrol on them. This was added to prevent patrols overlapping. If the road a patrol wants to travel down is not locked road then the patrol can add the road that it is travelling on (its *setRoad*) to the hashmap. Once a patrol has stopped travelling on a road (it has reached a junction) it unlocks its *setRoad* road from the hashmap by calling *unlockRoad* so another patrol can travel on it.

As *PMS* inherits from *SimpleSprite* (see fig.2) it has a health bar, the value of which decreases when a fire truck attacks it. Once it is at 0 the patrol is removed from ETPatrols in *GameScreen* hence disappearing from view (AUDIENCE_ACCESSIBILITY). To shoot a projectile at a truck the truck has to be within the patrols detectionRange (checked in isinRadius) and the patrol cannot have fired a projectile in the past second (checked in canShootProjectile), this check stops the patrol for constantly firing.

Carpark: As patrols can access the entire map (CREATE_ENTITIES) and there has to be at least four trucks (VARIED_TRUCKS) patrols could attack trucks that the player is not controlling. Hence carparks were added so the player could change the truck they are controlling and keep trucks they are not currently controlling safe from patrols. If the truck's location matches a CarParkEntrance (see constants line 157) the truck enters a carpark and *CarParkScreen* is created. If a truck is parked in the firestation carpark and the firestation is not destroyed then the truck will repair and refill (RETURN_HOME).

Shop: shop functionality was added into *CarParkScreen* so that the player could see which trucks they could buy (OPEN_SHOP) alongside the trucks they already had. As the player buys and changes trucks in the same place the player can easily change to a truck that they had just bought. To buy a truck the player clicks on an unbought truck and, if the player's score is greater than or equal to the price of the truck (BUY_ITEM_FUNC), they purchase the truck and the truck's price is minused from their score.

Mini game (see fig.3):

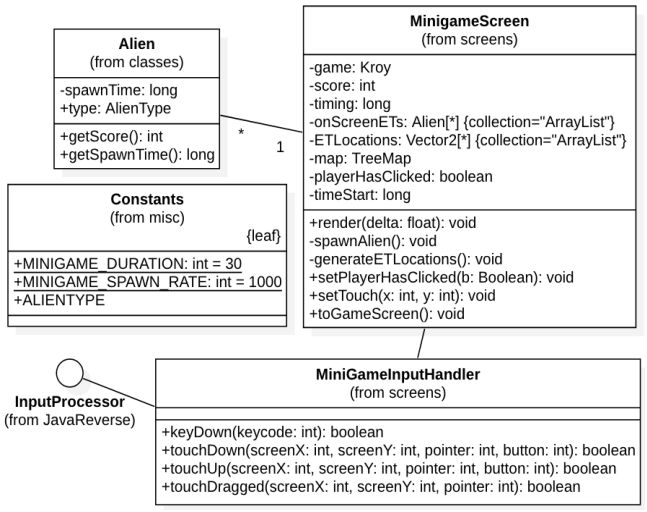


Figure 3

MiniGameScreen is triggered from *GameScreen* when the firetruck collides with a *MiniGameSprite*. In the minigame multiple Aliens spawn at random locations on the screen which the player has to click on to score points before they disappear. The player has 30 seconds (defined in MINI_GAME_TIME in constants) to score points which can then be spent in the shop.

Each Alien has an AlienType (an enum in constants see fig.3) containing how long they stay on the screen, their chance of coming up on the screen and how many points the player gets from clicking on that alien. Storing these values in an enum makes it easy to add more types of aliens and balance the game.

spawnAlien, which is called every second (defined in MINI_GAME_SPAWNTIME in constants) spawns an Alien of a random type at a random location. The Alien's *AlienType* is determined from *map* which maps the chance of each *AlienType* appearning to that *AlienType*. A value from 0.0 to 1.0 is then randomly generated and rounded up to the nearest entry in *map*, the *AlienType* associated with that nearest entry is the *AlienType* of the *Alien* that is spawned.

When the player clicks on the screen *keyDown* in *MiniGameInputHandler* is called which sets *hasPlayerClicked* to true and calls *setTouch* with the coordinates where the player clicked. These values are used in *render* to see if the player has clicked on an alien. An InputHandler was added to solve the

issue where if the player clicked on an alien and lifted up and another alien spawned there it would register that the player had clicked on that alien. This was solved by overriding *touchUp* in *MiniGameInputHandler* to set *hasPlayerClicked* to false when the player stops touching the screen.

New screens: In addition to *MiniGameScreen* and *CarParkScreen* mentioned above *PauseScreen*, *StoryScreen*, *HowToPlayScreens* and *GameOverScreen* were added. *StoryScreen* is created after the player presses 'play' in the main menu and gives the player some background to the game. *PauseScreen* can be accessed within the game or minigame by pressing Escape, this is implemented in the keyDown functions in their respective InputHandlers. *PauseScreen* is passed *GameScreen* or *MiniGameScreen* so the player can resume the game in the same state that they paused it in. The how to play screen can be accessed only via the pause screen. *GameOverScreen* which displays whether the player has won or lost (WIN_GAME_FUNC) is created after the game has ended. The interactions between screens can be seen at https://sepr-docs.firebaseapp.com/implementation

Tutorial: As the player may not read the user manual or view the how to play screen a tutorial was added at the start of the game (GAME_DOCUMENTATION). When the player is in the tutorial *isTutorial* (see line 294 in *GameScreen*) is set to true, in this state popup tips are added to a queue that are displayed every ten seconds at the bottom of the screen and the player can drive around the map without being shot at. If enter is pressed the GameInputHandler sets *isTutorial* to false and the player starts the game.

Firestation can be destroyed: firestationtimer (see *GameScreen* line 268) is used to count down 3 minutes (FIXED_TIME). When the timer reaches 0 the Boolean flag *isVulnerable* is set to true which means that patrols can attack the station. As *Firestation* extends *SimpleSprite* it has a healthbar which decreases when it is attacked; when the firestation's health reaches 0 its texture changes and trucks can no longer repair and refill there (TIME_ACCESSIBILITY, FIXED_TIME).

Arrow: When gathering feedback for our user manual a player said it was hard to find fortresses, thus we added an arrow to point in the direction of the nearest fortress. The arrow isVisible can be seen when the player presses the space button (see GameInputHandler line 42). *setNearestfortress* (see line 299 in *GameScreen*) is called each frame and returns the nearest fortress to the truck. The position of this fortress is passed to *aimAtTarget* in *Arrow* (see line 35) which sets the direction the arrow should point in.

Features not fully implemented

- Entities should always act the same way (CREATE_ENTITIES_FUNC)
  We interperated this to mean that every game there will be the same number of patrols that all move constantly and have the same health and attack points, thus this is what we implemented. However, to increase enjoyability their paths are random hence they do not "act the same way."
- The user should be able to press a button to open a shop (OPEN_SHOP_FUNC)
  Allowing the player to access the shop at any point in the game prompted implementation questions such as: where would the bought truck be spawned? Could the active truck be attacked by patrols truck while the shop was open? Hence to maintain internal consistency and to make the controls more intuitive for the player the shop can be opened by driving into a carpark.
- Destroyed patrols or fortresses should be removed from view (DESTROY_ENTITIES_FUNC)
  The previous group did not meet this requirement and instead added a *removeSprite* (see…..) function in *SimpleSprite* that changes the texture of destroyed entity. As fortress inherits from simpleSprite we used this method so it looks like the player is flooding the fortress which is more enjoyable then the fortress disappearing. Destroyed patrols are removed from view.
- After 3 minutes the user cannot repair their truck at the fire station (FIXED_TIME)

As the map is large 3 minutes was not enough time for the player to attack a fortress and repair their trucks before the station is destroyed so instead after 3 minutes the station becomes vulnerable to attack by patrols; once the patrols have destroyed the station trucks cannot repair and refill there.