Module	SEPR
Year	2019/20
Assessment	3
Team	FarmJabStudio
Members	Jay Brooks, Amy McArragher, Rian McQuillan-Howard, Benjamin Kelly, Ahmed Tariq, Marcel Miro Puges, Faaiz Shanawas
Deliverable	Implementation Report

How each class meets requirements and architecture

Class	Brief Justification and Related Requirements
Entity	Each of the entities inherit characteristics from the entity class. Entity(int, Texture) method initialises an entity with the correct texture. takeDamage(float) decreases health by the damage output of the projectile the entity is hit with. directionTo(Entity), distanceTo(Entity), directionTo(float, float), distanceTo(float, float) returns distance and direction from another entity or an x/y coordinate on the map to detect collisions and check if another entity is in range. The properties health and maxHealth, determine the current health and maximum health that an entity can have. x and y float properties are used to track the current position of any entity on the map. The boolean property destroyed is used to determine whether or not an entity still exists on the map this is turned to true when the health is depleted. **(UR_HEALTH, FR_FORTSHOOT, UR_FORTSHOOT, FR_COLLISION)**
Moveable(Interface)	Each entity also inherits from the Moveable interface so iit's movement on the map can be controlled. turnLeft(), turnRight(), goForward(), goBackward() methods determine the four directions of movement for each entity. Entities trigger these methods either by user input, randomly or as a result of interactions with other entities. **(UR_DRIVE, FR_CONTROLS)**
Attack (Interface)	Each entity also inherits the Attack interface which contains the single method attack() which triggers an entity to shoot.**(FR_SHOOT, UR_SHOOT, FR_FORTSHOOT, UR_FORTSHOOT)**
Firetruck	water and maxWater fields are used to store the amount of water in the firetruck. Acceleration, deceleration, direction and velocity, are used to determine the properties of the movement of the firetruck. The drops list stores each water drop or bullet in a list. range and flow rate floats determine the speed and distance the fire truck can shoot. piConstant float is used to calculate the angle of trajectory of the water drops. takeWater(int) and refill() are used to refill the fire truck with water. turnLeft(), turnRight(), goForward(), goBackward() define the 4 directions of movement of the fire truck triggered using the WASD keys. The fire truck also has the int property water and the float properties velocity and direction, which enable each firetruck to have different capacity and speed e.tc **(FR_SPEED, UR_DRIVE, FR_CONTROLS, FR_REFILL, UR_REFILL)**
Projectile	direction and velocity determine the movement of projectiles. timePassed and disposable variables are used to remove the projectile from the screen. lifeTime, startTime floats are used to determine when the projectile was created and when it should be removed if it does not collide with another entity. turnLeft(), turnRight(), goForward(), goBackward() are used to define the movement of each projectile. **(UR_SHOOT, UR_FORTSHOOT, FR_SHOOT, FR_FORTSHOOT)**
Fortress	This class contains the list of goo (bullet) objects in the list goos and number of fortresses. Boolean value able_to_attack acts as a switch enabling attacks on the firetruck when in range and disabling when not. piConstant float used in calculating the projectile trajectories. startX,startY, damage floats control starting position and damage output of each new fortress. maxHealth int, barHealth texture tracks current fortress health and displays it to user in the form of a health bar. createGUI(), addGoo(Entity, Float), draw(Batch) are used to create and initialise an instance of the fortress. attackSpiral(), attack(), attack(Entity, int), attack(float) used to define the different types of attacks the fortresses can do. getStartX() and getStratY() return the x and y coordinates of each fortress to be used to identify a successful attack from a firetruck. directionTo(Entity) determines the direction of attack for each fortress. **(UR_FORTSHOOT, FR_SHOOT, FR_FORTRESSHEALTH, UR_INFO)**
ETPatrol	The ETPatrol class is an extension of the Fortress class with movement capabilities added in. x and y floats determine randomly chosen position patrol moves to, changed upon reaching the coordinates or having moved in the same direction for 5 seconds calculated using the movetimer float. If ETPatrol gets in range of Firetruck it stops moving. movex, movey, direction, velocity determine the distance,

	direction and speed at which the ETPatrol travels. chooseTarget() used to determine the random location in which the ETPatrol will travel to. **(UR_SHOOT)**
Kroy	This class is used to control the sprites in the game, using the methods create() and render() to initialise a sprite and dispose() to remove a sprite from the game appropriately
MainGame	This class is used to create and define the main game window. It inherits the Kroy game class, and all entities are passed into it, all the entities are stored in the array entities. currentTruck and camTruck are used to follow the active fire truck as the user plays. pmap generated the Pixmap and the map texture is applied onto it. **(UR_AESTHETIC, FR_GUI, NFR_GUI)**
MainMenu	This class is used to define the menu screen. It inherits the kroy game class and menuImage allows the appropriate texture to be assigned to it. render(float) and resize(int, int) initialises the menu screen and allows it to be resized in width and height. show(), hide(), pause() and resume() are used to allow the window to be minimised and inactive. dispose() allows the menu screen to be deleted and when the game is run. **(FR_GUI, NFR_GUI, UR_AESTHETIC, UR_INFO)**
MainWin/Main Lose	This class is used to render and control the win/loss screen after the game is won or lost. It inherits the kroy game class and endImage allows the appropriate texture to be assigned to it. The show(), pause(), resume(), hide() methods allow the window to be minimised and inactive. render(float), resize(int, int) creates the window and allows the width and height to be adjusted by the user. Finally the dispose() method deletes the game window when closed.**(FR_WIN, FR_LOSE, FR_GUI, NFR_GUI, UR_END)**
MiniGame	This class is used to define the minigame which needs to be completed to refill the fire truck at the station. It creates a sprite of 3 aliens (alien1, alien2, alien3) and a truck. The pipes array contains a list of all the pipe sprites and the correctipes array is an ordered list of the correct combination of pipes to complete the minigame. The selector sprite is used to move between available pipes and the current pipe is stored as an int in currentPipe. The create() method initialises the minigame.render(float), show(), drawPipes(batch), initialises the minigame and displays each of the pipe sprites. handleInput() receives input from the user and moves the selector accordingly. checkforWin() and getSolution(int) checks the current game status with the winning combination in correctpipes. resize(int, int), pause(), resume(), hide() allows the window to be resized in width and height and minimised and made inactive. **(UR_INFO, UR_REFILL, FR_REGEN, UR_HEALTH, FR_REFILL)**
FireTruckMenu	This class is used to represent the information of all the firetrucks to the user on the screen. The table elements backgroundTable, iconTable, barTableTruck1, barTableTruck2, barTableTruck3, barTableTruck4 are used to render the table outline and truck icons and appropriate information for each fire truck in a table format. The progressBars array stores all the progress bar objects for each truck providing a graphical representation of health and water levels. progressBarStyleHP() and progressBarStyleWater() methods are used to create the progress bar indicators. create() and addProgressBar(ProgressBar, float), renders the table and adds each progress bar to it as it is called. **(UR_INFO, FR_GUI, UR_AESTHETIC, NFR_USABILITY, NFR_GUI)**

Significant Changes to Software Architecture Report

Change	**Report and Justification**
Fire Truck	Fire truck controls changed from the arrow keys to the WASD keys. Contradicts requirement

controls	**FR_CONTROLS**, we decided to implement this change as some keyboards do not have arrow keys, which would make game unplayable as noted in the requirements risks. FireTruck speed increased to mitigate risk associated with **FR_SPEED**. FireTruck rotation speed changed to scale with the movement speed in order to make movement more seamless **UR_ENJOY.**
Fire Truck collision	Collision detection is now used to prevent the Fire Truck from driving “through” the ET fortresses. Change was made to adhere to requirements: **FR_COLLISION, UR_DRIVE**
Fire Truck range	The range of projectiles shot by the Fire Trucks has been reduced due to Fire Trucks outranging ET fortresses **UR_ENJOY.**
Entity location	Locations of entity (GetX and GetY) were being incorrectly reported due to locations not being taken from the centre of the object and constants were being added to locations to compensate for this. GetX and GetY were overwritten in this class to ensure correct locations are reported to reduce code complexity.
ET fortresses improve over time	Damage of ET fortress’s projectiles now increase the longer the game goes on. Change was made to adhere to requirements: **UR_DIFFICULTY, FR_FORTRESSHEALTH, NFR_TIME**
Projectile speed, range and damage changed	Projectiles from the ET patrols,ET Fortresses and FireTrucks damage increased by 0.02, damage and range of ET projectiles increased to improve the difficulty balance and pace of the game, **UR_ENJOY**
Additional FireTrucks added	Two more user controllable FireTrucks were added to the game in order to meet the Product Brief specifications for Assessment 3.
FireTruck sprites changed	FireTruck Sprites replaced with new ones that have their own colors in order to better meet requirements **UR_AESTHETIC**, **UR_ENJOY** and **NFR_USABILITY** as the FireTrucks are now easier to differentiate from each other
ET patrols were added	ET Patrols that wander the map randomly, shoot projectiles at the Fire Trucks and can be destroyed by the Fire Trucks added to the game in order to the meet the product brief specification and requirements **ET_PATROLS, UR_SHOOT**
Additional ET Forts added	Three more ET fortresses were added in order to meet the product brief specification for assessment 3
Minigame added	A minigame was added in order to meet the product brief specification and requirements **UR_MINIGAME,** it is won by making a connected path with pipes from a fire truck sprite to the ET patrol sprite, a basic tutorial was added to the minigame screen, thus meeting **UR_ENJOY, UR_AESTHETIC, UR_INFO, UR_TUTORIAL, FR_GUI**, upon completion of the minigame the FireTruck health and ammo is restored to maximum this contradicts **FR_REFILL** and **FR_REGEN** as ammo and health doesn’t restore automatically at the FireStation, we made this change as we felt it was a convenient way to fit the minigame seamlessly into the main game while maintaining **UR_HEALTH**
Fortress hitboxes	The hitbox size of the ET fortresses was increased in order to meet requirements **UR_SHOOT, NFR_GOALS, UR_ENJOY** as previously it was too difficult to hit the Forts
Disabled window resize	The game screen window is now non resizable, this is to prevent bugs and the user from making the game screen too small to be seen **UR_AESTHEHTIC**
Fort attacks	ET Fortresses attack patterns were changed so that all of them are unique in order to meet

changed	**FR_VARIATION** and **NFR_VARIATION**
Bridges width	Width of the bridges were increased in order to meet **UR_ENJOY** as they were difficult to navigate
Updated Splash screen	Splash screen information updated to reflect changes to controls in order to meet **UR_INFO, NFR_OPERABILITY** and **UR_TUTORIAL**.
Truck code refactored	Code implementing the speed of the truck in FireTruck class was refactored in order to meet **UR_UPDATES, NFR_MAINTAINABILITY**
ET Fort health bar	ET fortresses now have health bars above in order to meet **UR_INFO** and **NFR_USABILITY, UR_TUTORIAL**
Fire station Destruction Implemented	After 3 minutes have passed the Fire station turns grey indicating that it has been destroyed **UR_TUTORIAL**, minigame is no longer playable, meaning the Fire station can no longer repair or refill the fire trucks, thus meeting the product brief
Truck aiming changed	Fire trucks now shoot water in the direction of the mouse pointer, contradicts **FR_SHOOT** which states Fire Trucks must shoot in the direction of movement, change implemented because aiming and dodging ET projectiles simultaneously was difficult, which contradicted **UR_ENJOY**
Bug fixes	The Truck no longer bounces forwards when reversing, fixed by setting truck velocity to a maximum negative velocity instead of a maximum positive velocity when reversing at max speed. Truck no longer gets stuck when rotating against a wall, fixed by checking collision at the center of the truck instead of the corner. Trucks are no longer able to damage each other, upon death of a truck the player is automatically switched to the next alive truck. It is now impossible to switch to a dead truck.

Requirements deviated from or not yet fully implemented report

Yellow implies nearly fully implemented. **Orange** implies partially implemented.

Requirement	**Explanation**
UR_INFO	Splash screen doesn't indicate controls for the minigame, or that the fire station can repair the trucks.
UR_TUTORIAL, NFR OPERABILITY, NFR_USABILITY	There is no indication that the fire station has been destroyed other than it turning gray.
FR_REGEN, FR_REFILL	FireTrucks are instantly healed upon completion of the minigame (which takes place at the Fire station).
FR_CONTROLS	Fire Trucks move via WASD instead of arrow keys, deliberate deviation.
NFR_TIME	It is possible for the player to survive for longer than 10 minutes without losing, however winning the game is extremely difficult. During play testing, nobody took longer than 8 minutes to either win or lose when playing seriously.
Product Brief: Fire station destruction	Dies after 5 minutes only when a Fortress has been destroyed or after 7 minutes otherwise.