Module	SEPR
Year	2019/20
Assessment	2
Team	Salt N Sepr
Members	Alberto Lee, Archie Godfrey, Jacqueline Eilertsen, Jake Schoonbrood, Joshua Levett, Matteo Barberis
Deliverable	Architecture (Arch2)

Architecture Report

It is important that we map out the inheritance when using an object-oriented approach. Doing this wrong can result in unpredictable behaviour and additional complications. UML helps us counter this issue as it makes it easier to design a model where we can break a system down into components that communicate with each other.

UML (Unified Modelling Language) is a standardized modelling language that is often used to model software systems. We used UML to describe the architecture of our system. This is because it creates a visual representation that describes classes, attributes, methods and inheritance which is good for object orientated programming. The use of UML also streamlines the development process.

We used an online tool known as draw.io to develop our UML diagram. Although the tool is not tailored to UML specifically, it provides a seamless interface to aid in the development process. It’s also a free-to-use tool that allowed multiple users to edit it without the need to constantly swap files as it sync’d to our project files.

Changes in Architecture

Figure 2 is the UML Diagram that we first designed in Assessment 1 before the implementation phase. We have changed this design to a more object-oriented approach. We also have addressed additional requirements in Figure 1. We also noticed some of the methods we needed to create were already provided within the graphics library. An example of this is the Timer class in Figure 2, it was simply replaced by an attribute in Figure 1.

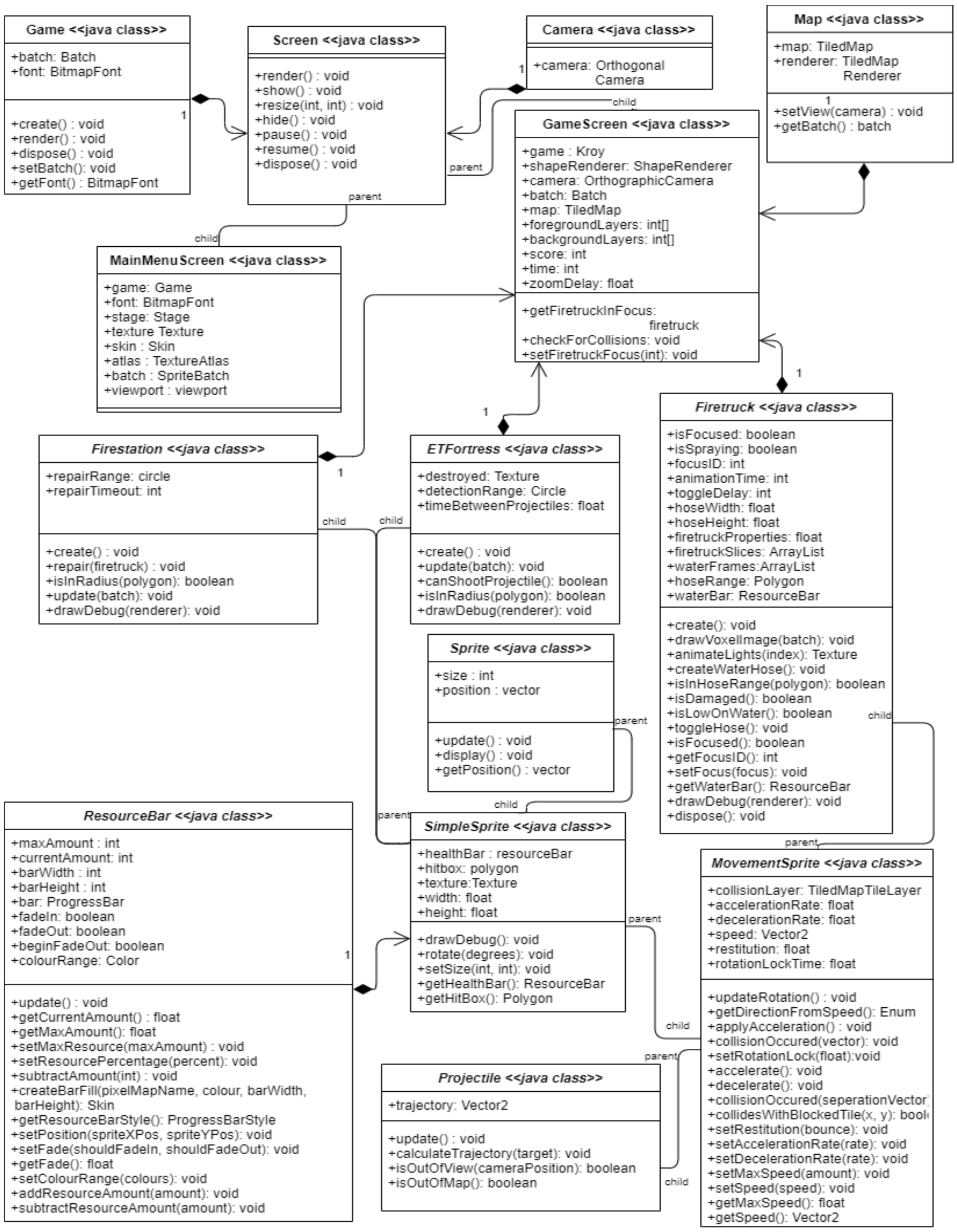

Figure 1

Game

The game class is where the main game operations will be executed.

Screen

This is a libGDX built in class which allows the game to be rendered, paused if necessary, resumed and other functionalities from which the classes MainMenuScreen and

GameScreen inherit from. This class is not part of the first architecture report we “didn’t want to give specifics on the number of screens in the final game”

Camera

The Camera class will hold operations handling the view of the game. It will handle resizing of the view as well as the translation of the view across the game world.

MainMenuScreen

The MainMenuScreen class contains the main menu for the game, allowing a user to decide what they want to do. This meets the requirement “MENU” established in Req1.

GameScreen

The GameScreen is a class which inherits functionality from the Screen class and extends it. The rendering of the game and the collision detection for fire trucks, buildings and projectiles is handled here as well as calculating the damage they do. GameScreen also handles inputs for switching between firetrucks and then toggling the water hose on the selected truck.

The GameScreen class meets the requirements “VARIED_TRUCKS”, “WIN_GAME” and “CREATE_MAP” from Req1.

Map

The Map class will handle all operations involving the creation and rendering of the map. The map will take in a map created by Tiled, specified in our Methods and Planning document. This also meets the requirement “CREATE_MAP” from Req1

ETFortress

The “AlienBase” described in the previous architecture report has now become an ETFortress. The ETFortress will be able to shoot projectiles at the player and the spawnVehicle function will now be handled by the GameScreen. This meets the requirement “DESTROY_ENTITIES” from Req1.

Firetruck

The Firetruck class represents a fire truck, it can move and collide with other objects within the game. The class controls the creation, animation, and handles user input for the direction of the fire truck. Differently from the previous architecture, the “Vehicle” class has become a more generalised “MovementSprite” class, from which the Firetrack inherits to allow movement. This meets the requirements “CONTROL_TRUCK”, “CONTROL_SPRAY” and “DESTROY_ENTITIES” from Req1.

ResourceBar

In the previous architecture report, fire stations and fortresses would only have an integer variable to hold the health value. We decided to improve readability by providing a bar to represent the integer value. ResourceBar is generalised s that classes can use the ResourceBar to display health bars for vehicles, buildings and water bars for fire trucks.

This also meets the requirement “DESTROY_ENTITIES” as the resource bar can be used as the main health bar that determines whether an enemy is destroyed or not.

SimpleSprite

Instead of having “collidables” components for each object in the game, as designed in the previous architecture, the “collidables” are now hitboxes within the SimpleSprite class. This is the main sprite; all other sprites inherit this class. SimpleSprite manages the creation and modification of a sprite; health bars, hit boxes and sizes of sprites.

MovementSprite

The MovementSprite class implements functionality to allow the sprite to move. This is separated from the SimpleSprite in order to keep static sprites simple. Therefore, it is the parent class of any moving object in the game. In the previous architecture report, the Vehicle class would have handled the movement of the fire tracks and alien vehicles but now they have a common parent.

Sprite

The Sprite class is part of the graphics library module that we are using, libgdx. The class "holds the geometry, color, and texture information for drawing 2D sprites using Batch". [1]

Firestation

The Firestation Class contains a radius in which the firetruck can be repaired and refilled. This meets the requirement RETURN_HOME as when a fire truck is in the range of the station it is repaired, and its cannon refilled.

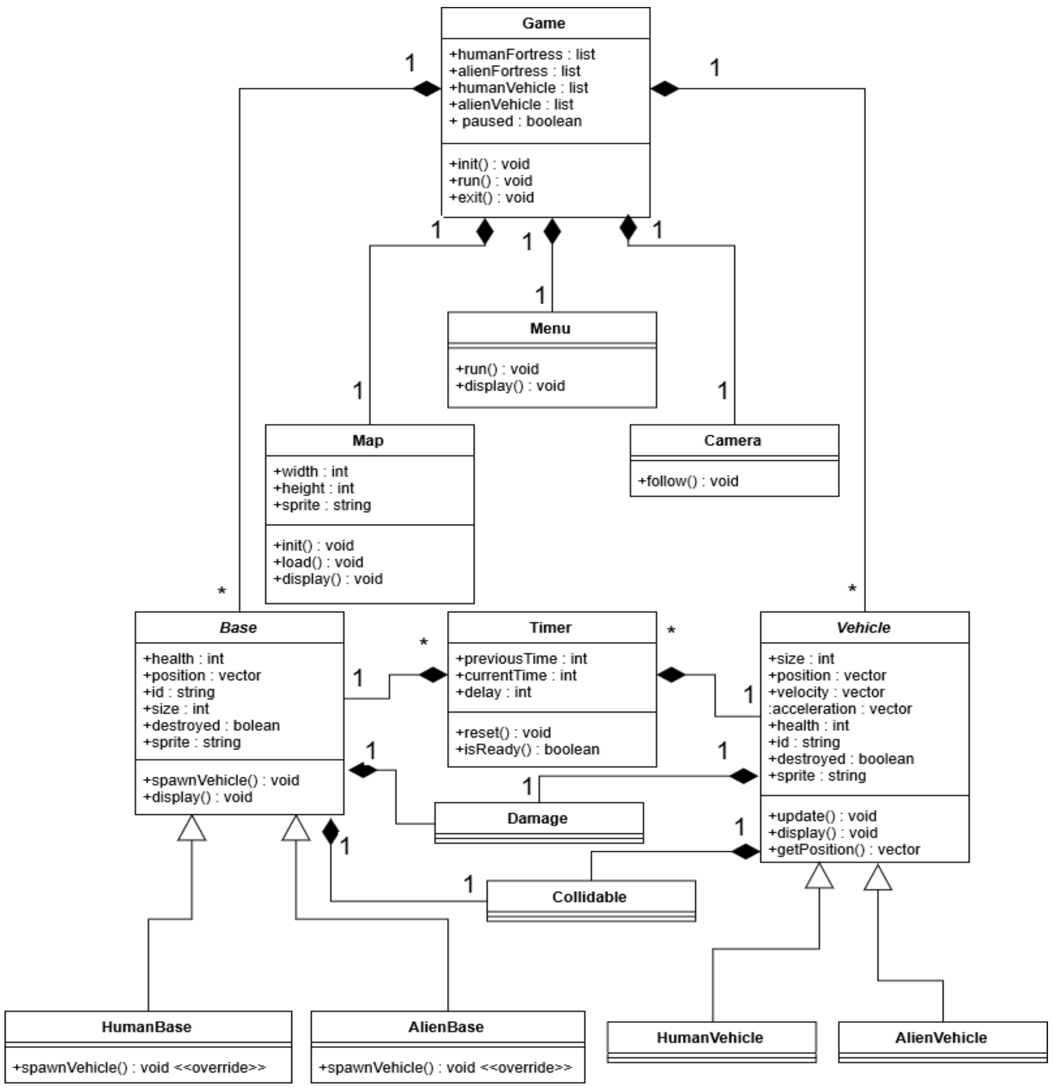

Figure 2

References

[1] "Sprite (libgdx API)", *Libgdx.badlogicgames.com*. [Online]. Available: https://libgdx.badlogicgames.com/ci/nightlies/docs/api/com/badlogic/gdx/graphics/g2d/Sprite.html. [Accessed: 02- Dec- 2019].

Appendix

Req1 (Requirements from Assessment 1):

User Requirements [2]

ID	Description	Priority
CONTROL_TRUCK	Control the direction the fire truck travels in	SHALL
CONTROL_SPRAY	Control the direction of the water the fire truck sprays	SHALL
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
WIN_GAME	Once the user has destroyed all 6 different fortresses they win the game	SHALL
CREATE_MAP	The user should be able to explore a map by controlling the firetruck	SHALL
CREATE_ENTITIES	The user should encounter alien patrols throughout the map	SHALL
DESTROY_ENTITIES	The user should be able to destroy alien patrols and fortresses by spraying them with water	SHALL
NO_VIOLENCE	There will be no violence to appeal to target audience	SHALL
OPEN_SHOP	The user should be able to find out how much different fire trucks cost at any point in the game	MAY
BUY_ITEM	The user should be able to buy different fire trucks from a shop	MAY
MENU	There should be a menu so that the user can access the leaderboard, the minigame, edit settings or play the game	SHALL
LEADERBOARD	See a local leaderboard to compare scores against other players	MAY

Functional Requirements [2]

ID	Description	User Requirements
CONTROL_TRUCK_FUNC	When the user presses the WASD keys the fire truck will move in the appropriate direction	CONTROL_TRUCK
CONTROL_SPRAY_FUNC	The direction of the water cannon will be controlled by the mouse	CONTROL_SPRAY
RETURN_HOME_FUNC	When the firetruck returns to the firestation it will repair and refill over a defined amount of time	RETURN_HOME
FIXED_TIME	After a fixed amount of time the user cannot repair their fire truck at the fire station	RETURN_HOME
NO_HEAL	Aliens should not heal after taking damage	DESTROY_ENTITIES
FORTRESS_HEAL	Alien fortresses should heal over a duration after taking damage	DESTROY_ENTITIES
DESTROY_ENTITIES_FUNC	If the alien patrol or fortress runs out of health it should be removed from view	DESTROY_ENTITIES
VARIED_TRUCKS_FUNC	Able to swap between trucks using a single button	VARIED_TRUCKS
CREATE_MAP_FUNC	A section of the map should be displayed to the user so that they can navigate it	CREATE_MAP
OPEN_SHOP_FUNC	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.	OPEN_SHOP
BUY_ITEM_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.	BUY_ITEM

Use Cases [2]

Scenario ID	Destroy fortress	Purchase item from shop	Lose game	Repair and refill fire truck
Primary Actor	Player of the game	Player of the game	Player of the game	Player of the game
Pre-condition	Player has water in their tank and fortress has health	Player is in the shop and has navigated to the	Player has no remaining fire trucks after a fire	Player has moved their fire truck to the fire station

		item they want to buy	truck is destroyed	
Trigger	Player sprays water at alien swarm	Player clicks the buy button	Player's fire truck is destroyed	Player's fire truck is on top of the fire station
Main Success Scenario	1) Player sprays at fortress 2) Fortress takes damage 3) Fortress' health reaches 0	1) Player clicks the buy button 2) They have enough money for the item	1) Player takes damage from an alien 2) Player's health reaches 0	1) Player's truck's health is increased over time
Secondary Scenarios	1) The Player stops spraying before the fortress' health reaches 0. The fortress then begins to heal 2) The Player runs out of water before the fortress' health reaches 0. The fortress then begins to heal	1) The Player does not have enough money. Purchase is cancelled and the Player is told why	1) Player stops taking damage before their health reaches 0. Game continues	1) Player moves away from fire station so repairing stops
Success Post-condition	The fortress disappears from the scene	Player receives the item	End game screen is shown to player	Player's fire truck's health reaches its full value

Non-Functional Requirements [2]

ID	**Description**	**User Requirements**	**Fit criteria**
TIME_ACCESSIBILITY	After a fixed amount of time the user will no longer be able to repair fire trucks, therefore the game will always end.	FIXED_TIME	The game should be playable within 5-10 minutes due to limited time on open days
GAME_DOCUMENTATION	It should be easy to understand that the game is won by destroying all 6 alien fortresses. This can be done by a small tutorial	WIN_GAME	The game should be easily understandable for the target audience within a single play through
RESILIENCE	The game should only be won	WIN_GAME	If the game is won

	when all 6 alien fortresses are destroyed		when exactly 6 fortresses are gone
AUDIENCE_ACCESSIBILITY	Instead of showing violence, the enemies will just disappear in order to satisfy the target audience.	NO_VIOLENCE	The game should be appropriate for prospective students and their parents
GAME_ACCESSIBILITY	The system must have a menu so that the user can access the main game and the minigame.	MENU	The game must have a minigame and therefore the user must be able to access it
OPERABILITY	The game should be playable on a PC but considerations should be made for mobile versions in the future.	CONTROL_TRUCK	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	LEADERBOARD	The leaderboard will be displayed to lots of people and sensitive information should not be shared.