3.a https://github.com/rgbalex/How-Hard-Can-It-Be

Meeting user requirements.

User requirement	Functional requirement(s)	Achieved by:
UR_GAME_INIT	FR_GAME_RESET	For assessment 2, a reset() method has been added to the GameManager, which resets itself, game entities and other managers to their initial state.
UR_SHIP_CONTROL	FR_SHIP_KB_INPUT	The PlayerController component class checks for keyboard inputs each frame and adjusts the velocity of the player ship's RigidBody component. PlayerController also checks for space button presses to shoot. The GameScreen class handles aiming with the mouse pointer and prevents accidental shots when interacting with other UI elements.
UR_COMPETING_COLLEGES	FR_COLLEGE_ENTITY_TRACKING	There are 5 factions spawned in the game, with different colours and flags. Each entity has a Pirate component, which tracks health, allows combat and tracks entity destruction.
UR_LEARNING_CURVE		In the GameScreen class, a tutorial outlining the game's controls is permanently rendered on screen. Other UI elements are labelled clearly and displayed prominently.
UR_GAME_DURATION		Non-defender buildings belonging to colleges (those that do not shoot back) are destroyed with one shot, allowing colleges to be captured relatively quickly, letting the user progress.
UR_FRIENDLY_SHIP_ENCOUNTER	FR_FRIENDLY_AI FR_FRIENDLY_INTERACT	The game is able to distinguish ally and enemy ships via their factionid and the isAlly method in the NPCShip class. Allied ships follow the player and shoot at enemies that engage with them, helping the player to progress in the game. The AINavigation component allows allied ships to steer towards the player.
UR_HOSTILE_SHIP_ENCOUNTER	FR_HOSTILE_AI	For the Pirate component class, a method to track targets was implemented. When the player comes within attack range of an enemy NPC, they get added to the NPC's list of targets. An NPC will then pursue its target, and, if within range, will fire in

		repeated intervals, which can be changed to accommodate different difficulty levels.
UR_FIRE_WEAPONS	FR_PLAYER_FIRE FR_PLAYER_AMMO	For the player class, shooting is done by scanning for either space bar presses or mouse clicks, which then fires an instance of a CannonBall class in the direction of either the mouse click or directly ahead of the ship. The ammo capacity is stored in the Pirate component of the player, which, for assessment 2, allows to dynamically change the maximum ammo capacity via upgrades from the shop.
UR_BULLET_DODGE	FR_BULLET_TRAVEL	The speed CannonBall instances travel with allows them to overtake the ship even when it is moving, but not move too fast to not be able to track and dodge them. CannonBalls have their own sprite, which is rendered on screen for the duration of their flight.
UR_HOSTILE_BUILDING_COMBAT		Enemy colleges have one defender building each, added as part of assessment 2. These are special instances of the Building class that can target and shoot at the player. To defeat a college, the player must destroy the defender first.
UR_HOSTILE_COLLEGE_CAPTURE		Enemy colleges and buildings have a Pirate component, which tracks when a building is destroyed for the Building class and when all college buildings are destroyed for a college class. Once that happens, the college is marked as captured.
UR_EARN_MONEY	FR_MONEY_TRACKING FR_MONEY_UPDATE	The player's Pirate component has a plunder field, which keeps track of the player's money. Upon killing enemy ships, collecting powerups or completing quests, the player receives an amount of plunder that can be changed for various difficulty levels.
UR_EARN_POINTS	FR_POINTS_TRACKING FR_POINTS_UPDATE	The player's points are stored in the GameScreen class and are updated with time. More points are received upon completing quests, earning plunder and navigating through bad weather.
UR_QUEST_PROGRESS	FR_QUEST_TRACKING FR_QUEST_RANDOMISE	The QuestManager class checks whether the current quest is completed

	FR_QUEST_OBJECTIVE	and communicates it to the GameManager. The sequence of quests is randomly created at the start of the game via the QuestManager's CreateRandomQuests method. The last quest in the sequence is always a boss fight, involving killing a seamonster (instance of DuckMonster), added for assesment 2.
UR_GAME_WIN	FR_BOSS_UNLOCK_TRACKING FR_BOSS_SPAWN FR_GAME_WIN	When the player has completed all quests, an instance of DuckMonster is placed on the map by the GameManager. Once the monster is defeated, the player is shown a screen displaying their statistics over the playthrough.
UR_GAME_LOSE	FR_PLAYER_DEFEAT FR_SCENARIO_FAIL	If the player's pirate component reaches zero health, the game finishes and the player is taken to a screen showing their achievements. This is checked in the GameScreen class on frame updates.
UR_SHIP_COMBAT		NPCShips have a Pirate component, responsible for firing cannonballs and tracking health, allowing for ship combat.
UR_OBSTACLE_ENCOUNTER		The user cannot go over certain cells of the map, like land, beach or grass, preventing them from accessing certain areas.
UR_WEATHER_ENCOUNTER		In set intervals, the GameScreen class darkens the playing field and renders a rain animation, showing that a bad weather period began. During this period, the player earns more points and a velocity is applied to their ship, representing harsh winds.
UR_SPEND_MONEY	FR_PLUNDER_SPEND	As part of assessment 2, a shop screen has been added to the game. It allows the player to spend some plunder to permanently increase ammo, speed, damage and health. To enable this, special fields and setters were added to the Pirate class to track maximum possible health.
UR_SAVE_STATE		A SaveManager class has been introduced along with a load method in the GameManager to restore the game to saved state.
UR_DIFFICULTY	FR_DIFFICULTY	The menu screen on startup allows the

		user to choose between 2 difficulty levels, which adjust the damage and health of NPC's as well as decrease the damage of the player in the "hard" mode.
UR_POWERUPS_QUANTITY	FR_POWERUPS	5 distinct types of powerups are available to be collected, leading to different effects to the player's ship.
UR_POWERUPS	FR_POWERUPS	Each powerup activates a timer within the PlayerController class that ensures the effect stops within a set amount of time. A set of boolean fields within the Player class determine which powerup is in effect and act upon the player.

Significant changes / new features.

- Text class has been deleted as it was not being used
- Added offset and size fields to the Renderable class to enable healthbars
- Shop and pause menus added to the GameScreen, with a boolean value that pauses rendering and entity updates while these screens are open.
- Pirate component's fields specifying maximum health, ammo and damage are now able to be modified during game execution to allow for permanent upgrades.
- A RandomWaterCell method has been added to the GameManager, which randomly picks a water cell from the map graph, enabling powerups to reliably respawn off land.
- For entities participating in combat, an algorithm was implemented to render a healthbar, consisting of a red and green element. These elements were dynamically resized every frame to correctly represent the current health of the entity.
- Powerup, SaveManager, DuckMonster and KillDuckQuest have been added to fulfill assessment 2 goals as outlined in the requirements document.

Missing functionality.

- Usage of obstacles as part of the game is quite limited
- Loading the game lacks robustness - game sometimes stutters upon repeated reloads.