Implementation Report

Introduction

For our progression to Assessment 3, we were required to complete the evolution of another team's project, whilst fulfilling the constraints which had been laid out by said team. In the process we were to take on board their architecture, requirements and code and continue with their vision in mind. As such, we went on to develop a sailing mode with keyboard input (**C3,F6**), a new minigame section (**F4**), and a background music system (**F17**)[1], to mention but a few additions. We were able to fulfill all requirements inherited for this assessment.

In regards to architecture, we decided to implement multiple extra elements to allow us to improve and extend the work of the previous team to fulfill the new objective. To do so, we first reviewed the previous team's Architecture Report and discussed how it had been designed to fulfill its purpose to the greatest of its ability. This meant that we were then able to understand which sections should be improved to allow for the new goals, as well as where any new elements would connect best in the current architecture design. The list of elements designed to expand and work in cohesion with the previous architecture is as follows:

BaseScreen, BaseActor, PhysicsActor: These classes were added as we decided to implement the template method design pattern in our game. They enabled the process of generating new Actors and Screens to become much easier as a lot of codes were reused, which also made it effortless keeping track of existing methods and attributes as well as adding new ones.

SailingScreen, SailingShip: These classes were implemented to fulfill requirements C3 and F6 of developing a sailing mode with keyboard input, the SailingShip class only deals with the ship in the SailingScreen unlike the Ship class which deals with the ship in CombatScreen.

MinigameScreen: This class was implemented to fulfill requirement F4 of including a minigame separate from the main game.

DepartmentScreen: When we took over the game, the DepartmentScreen (among others) would not properly scale on high-resolution monitors or to different window sizes. Instead of properly scaling, the game would simply stretch, causing the buttons to not work as intended. To fix this, we implemented all the GUI elements as actors instead of hard drawing them to the screen like the other team, as actors can be added to a stage with a FitViewPort that can be scaled. We also fixed a glitch wherein the game crashed if the player attempted to sell their weapon when the shop was already full. The game now checks for this and prevents the sale if the shop is full.

CombatScreen: This class was changed in a similar way to the DepartmentScreen as it had similar issues with regards to scaling. In addition to this, we fixed a few underlying issues such as a glitch with the fire button which essentially allowed you to skip turns caused by the button not properly deselecting a weapon after it was fired.

Additions and Changes to the Software

This section outlines the most significant changes that we have implemented to the project since acquiring it from the previous team. Each entry will include a brief summary of the change made, followed by an explanation of why we found it necessary to include in the improved program and how we did so. Note that references to the Requirements document [1] are used in multiple explanations to assist in justification of ana addition.

Change Made:	Converting from Batch Rendering to Stage and Actor Rendering
Explanation:	Decided to use Scene2D's library, rather than the previously adapted Batch Rendering method for ease of implementation of other features. While it does alter the look of the user interface, it meant that when we could work with differing resolutions without affecting positions of objects in the game. Further, we found it preferable when deciding to implement a pause function in the game, due to the ease of using multiple stages with a single InputProcessor.

Change Made:	Drastically changed how the render loop of all their screens worked
Explanation:	The inherited game had significant problems with memory and CPU usage due to the use of local variables inside of the render loop. Due to the speed that the render loop is repeatedly called, the memory of the machine running the game would quickly fill up as the java garbage collector would not be called often enough. Therefore we changed the screens to instantiate at the start and only change these references if needed, rather than creating a new variable each frame.

Change Made:	Implemented Music System
Explanation:	Added a music/sound system as per requirement **F17**, which allowed us to easily add new audio, whether it be background music or sound affects. To do this we used the LibGDX Audio module which allows for creation of Sound and Music objects, allowing us to add the sound effects and music very easily.

Change Made:	Added an all-new sailing mode
Explanation	Implemented a sailing mode to be initiated at the game start, along with a full map consisting of all the required colleges and departments. We then integrated this with the existing systems, such as the combat and department screens to form a cohesive game with the ability to access all screens from one-another.

Change Made:	Changed logging to use LibGDX's logging system.
Explanation:	Changed all the logging and debug messages to use the LibGDX app.log() system instead of System.out.println(). This allows us to change logging state to show everything, just info and errors, just errors or nothing simply by changing one line of code.

Change Made:	Implemented a minigame section
Explanation	Added a third game section, consisting of a Rock-Paper-Scissors style minigame that the player can bet their in-game money on, fulfilling requirement **F4**. This new section is contained within the MinigameScreen class and alters the player's gold using the pre-existing functions available in GameManager.

Change Made:	Removed previously drawn sprites
Explanation:	As the previous group had only implemented 2 departments and 3 colleges they had only generated assets for those. However, seeing as we have now implemented more colleges and departments we lack the equivalent assets needed and have no way to create more. Therefore, we instead decided to go for a stripped down UI which no longer uses/requires these assets, which seemed to work better for the team as well as provide a cleaner UI style for the user.

References

[1] SEPR ‘Requirements Updates’ Pi-rates. Available: http://www-users.york.ac.uk/~jf1279/docs.html [Accessed 17th Feb. 2019]