Team iPatch

Assessment 3: Implementation report

Christian Pardillo Laursen
Filip Makosza
Joseph Leigh
Mingxuan Weng
Oliver Relph

On requirements:

We have made some changes to the previous team's requirements[1] in order to better fit the requirements specified by the client and improve the final product. The changed requirements are:

- Questing system has been removed
    - It was deemed unnecessary, and work done towards the questing system could be used in areas more relevant to the requirements specified by the client.
- Items have been removed
    - Their function has been replaced by the departments.
- Departments now function as shops
    - The scenario specified that there should be a way to spend the plunder, and since the previous team built an infrastructure around departments but left them unused we decided to convert them to shops.
- The minigame is not strictly a "gambling game"
    - The previous team did not implement a minigame of their own, so when designing we decided to have a reflex-based minigame to reward players who perform well

And we also see the architecture of their project,the following classes were added by either extending or fulfiling their previous architecture.[2]

New classes:

MinigameScreen

- Handles minigame functionality (input, rendering, UI, layout)

ShopUI

- Handles shop functionality (input, UI, layout)

MinigameBuilding

- Stub class for Tiled map editor

Changes:

Pausing

- Temporarily stops execution of the game
- Allows the player to pause the game at will
- Pauses the game when the player enters shops or minigames

Related requirements: **2.13, 2.14**

There is no requirement which specifies that pausing has to be implemented, but it was necessary to add a way to pause the game to stop the player from being attacked by enemies while on the shop or minigame screens, requirements **2.13** and **2.14** respectively. There was no way to pause the game in the Assessment 2 software.
GameScreen was extended to keep track whether the game was paused for any reason, as well as to pause the game when entering a menu which took player control away from the ship. The **HUD** class was extended to display a message informing the player when the game is paused, and various entity classes (**Entity**, **LivingEntity**, **NPCBoat**, **Player**) were extended to prevent them from moving, firing, or reacting to input while the game is paused.

Final boss requires capturing all other colleges to access

- Requires the player to capture multiple other colleges before fighting the boss of the final college (Derwent)

Related requirements: **2.12**

Requirement **2.12** specifies that “each gameplay should have an objective [which] should not be immediately achievable”. This was not in place at the Assessment 2 software.
The **GameScreen** class was extended to check whether the player had captured the other colleges on the game map, not allowing them to progress to the Derwent fight if they had missed any colleges. The **HUD** class was extended to display a message to the player if they attempt to capture Derwent early, informing them that they have to fight the bosses of the other colleges first.
In addition, the pause menu shows which colleges have been captured already by colouring them green if captured and red otherwise, easing the job of the player in attempting to find out which college it needs to challenge to be able to complete the game.

Our extension for the project keeps the open world map layout, but prevents the player from accessing certain areas unless they have enough progress. Bosses become more difficult in more distant colleges, ensuring the player is challenged as they progress. Requirement **2.12** has been met.

Minigame

- Adds a new minigame which is triggered similarly to a shop when the player is in close proximity to a Minigame Building.
- Allows the player to pay some gold to attempt a game of reflexes. The player gets substantially more than they paid in (generally five/ten times as much) if they win, losing only the gold they paid in if they lose.

- Four difficulties, with the easy difficulty serving as a tutorial to understand how to play the minigame. Higher difficulties cost more gold to play, but they give the player more gold for winning.
- Suspends the main game to prevent the player from being destroyed when they are playing the minigame and restores the game state to prevent loss of progress once they are done.

Related requirements: **2.14**

The Assessment 2 software did not implement the minigame and the architecture documents did not mention the minigame in any capacity, but it was part of the requirements document as described in requirement **2.14**. Because there were no guidelines encouraging a specific way to implement the minigame, we decided on a way to implement and integrate it into the software ourselves.
Minigame functionality is implemented in the **MinigameScreen** class, and it is integrated into the game by extending the **BuildingManager** class with code which instantiates the minigame. We added the **MinigameBuilding** stub class to allow us to place minigame locations on the world map using the Tiled map editor, similarly to how colleges were implemented in the Assessment 2 software. The **PirateGame** class had to be extended to allow the switchScreen() method to work for instances of the minigame, and the **ScreenType** enum was changed to include **MINIGAME** as a valid constant.
Our extension to the project implements a minigame which is playable, can be used to gamble for gold, and it completely mechanically distinct from the main game. As such, requirement **2.14** can be considered met.

Added additional colleges and departments
- 2 additional colleges
- 1 additional department

Related requirements: **2.5**

The Assessment 2 software implemented several colleges and departments, only three colleges and two departments, instead of the five and three respectively specified by requirement **2.5**. We extended the **departments** and **colleges** JSON files to define additional buildings, and modified the game map to add them to the world. Requirement **2.5** can be considered met.

Changed the role of the departments
- Departments now function as shops
- Each department provides unique upgrades

Related requirements: **2.13**

As per requirement 2.13, there should be a way for the player to spend gold and get upgrades or repair their ship. This was implemented by having 3 departments, each of which provide either healing, speed upgrades or projectile upgrades.

When departments are approached, a notice saying *Press E to enter department* appears, implemented in the **HUD**. The player will then enter a department screen where the gameplay is paused and they are presented with different upgrade buttons which they can press to obtain the upgrade if they have enough gold, as well as a gold counter and the current parameters of their boat. This is all implemented within the new class **ShopUI**.

Refactoring old code

- Bugfixes
- Improvements to code quality

Related requirements: **2.11**
There is no specific requirement for this, but for the sake of maintainability and ease of reading, as well as adding small extra features to allow us to implement the requirements for assessment 3, some of the previous code has been altered.

This includes enabling the level up rewards, overloading the **Projectile** constructor to accept damage as a parameter (required for difficulty increases and damage upgrades), refactoring the **NPCBuilder** to not produce a new instance of itself every time it's generate method was called, and other small changes aimed at improving overall code quality.

In addition, the reward issued when defeating an enemy has been made to scale with the difficulty of said enemy. This matches the specification supplied by the client.

Not fully met requirements

There are a few requirements specified by the previous team that our code did not meet, but none of them were required for assessment 3. These include questing and items, which we decided not to implement, as specified by requirements **2.16** and **2.10**, as well as the requirement for a tutorial, **2.17**, which has been replaced by a manual.

Despite this, all the requirements specified by the client for the finalised product have been met, with the possible exception of bad weather which was not strictly required.

Reference

[1]https://team-ipatch.github.io/assessment3/Req3.pdf
[2]https://sepr4.github.io/web/submission/assessment2/Arch2.pdf