Implementation

Team 6
Sanjel Sadikaj
Adam Hagan
Tudor Ciobanu
Na Tang
Armintas Zadeika
Liu Zhang

Implementation:

In the code we wrapped all new code in the following comments:

```
// Added code start
// Added code end
```

And wrapped all modified code in the comments:

```
// Modified code start
// Modified code end
```

Some types of changes were not marked with comments - the addition/removal of whitespace to conform to style, and the addition of new imports. Most of these changes were performed automatically by our IDE and so could not be wrapped. All other changes to the code are marked with wrapper comments.

The additional requirements were met without significantly changing the architecture.

- UR_POWER-UPS was implemented mostly through the creation of a new PowerUps class, with some additions to the collision logic in the Map class.
- UR_CHANGE_DIFFICULTY was implemented mostly through additions to the Map class, as well as the creation of ChooseDifficultyUI - a simple UI class.
- UR_SAVE_GAME was implemented using libgdx's preferences system (more details in the New Features section), and is mostly contained in the Game class.

New Features:

- ChooseDifficultyUI was added - this is a fairly simple UI class that displays the currently selected difficulty and allows the user to change it by clicking buttons. Clicking "play" moves the user to the boat selection menu.
- PowerUp was added. Similar to Obstacle, the behaviours each powerup's effect are defined in the collision handling code in Map.
    - While it is similar to Obstacle, we decided the classes didn't have enough common code to justify making an abstract class for non-boat river objects.
- InfoUI was added - this is an extremely simple UI class that displays some instructions, and has a back button to return to the main menu.
- OptionsUI was added - this is another fairly simple UI class which lets the player change their music volume and control scheme (from WASD to arrow keys).
- We created a pause system in the Game class, which allows the player to pause the game at any time to save.
    - This isn't directly linked to a requirement, but we felt it was important that the player could save in a paused environment.
- We implemented a saving and loading system - most of the code for this is in the Game class, though some of it (like constructors for loaded objects) is in other classes in the core package. Once a player pauses the game and clicks the save button, the game automatically gets the required data and adds it to a file.
    - We implemented save and load using libgdx's preferences system. Preferences provides a persistent data storage location that works on any platform or OS, with the actual storage depending on the platform the code is run on (in the case of Windows, a file is created in "User/.prefs")

- Enough information to recreate all the objects in the game world (maps, lanes, the player, AIs, obstacles, powerups, plus some meta information such as the current leg number) is stored in preferences when the game is saved.
- For our purposes, preferences is functionally a map of String -> (Float || Int || String). Keys are given a unique name that identifies the object and its location (for example, the 18th obstacle on the first map in the first lane would be given the name `Map0Lane0Obstacle18`), followed by the property being saved (a full key might be called `Map0Lane3Obstacle7PosX`).
- When loading, the game rebuilds the game world by creating objects using the information in preferences.

Change Report

- Game changes:
  - Added two new buttons (instructions and continue).
    - Fulfill requirements: UR_DISPLAY_INSTRUCTIONS 1.2.3, UR_CHANGE_SETTINGS 1.2.4
  - Added a pausing system - Game tracks if the game is paused, and a branch in the main render loop will either run the game code or the code for the pause state.
  - Responsibility for handling of the box2d physics world was moved into Map - more details on this change in the Map section.
  - Time reductions are reset on a new leg.
    - Fulfill requirements: FR_POWER_UP_ITEMS 2.0.12, UR_POWER-UPS 1.0.3
  - Previously all three maps were created when a game was started - maps are now created as needed when the leg starts.
    - This was done in order to reduce the loading time when starting a game - NFR_FAST_TRANSITION requires all transitions to take less than 0.5 seconds, and the original transition was over 5 seconds.
    - While this wasn't enough to meet the requirement (discussed below), the change was kept as it was partially successful.
  - Input was extended to allow arrow key controls (if selected in options).
    - Fulfill requirement: UR_CHANGE_SETTINGS 1.2.4
  - The bulk of the saving system was added to this class - more details on this system can be found in the new features section above.
    - Fulfill requirements: UR_SAVE_GAME 1.1.2, FR_SAVE_&_LOAD 2.0.11
- Map Changes:
  - All code relating to the handling of the box2d physics world was moved into Map from Game.
    - The world is now stored as a property of Map instead of Game, and is created in Map's constructor. The functions for setting it up have also been moved to Map as a result.
      - This was done because maps have a 1:1 relationship with worlds - while it was previously possible to run Map's methods with inconsistent worlds, this resulted in buggy behaviour and

crashes. Map was given responsibility for its world in order to remove this inconsistency.

        - The logic for handling and stepping the world has also been moved to Map - this includes the physics simulation and special collision logic (such as obstacle damage and powerup effects).
            - This was partially done as a logical consequence of moving the storage of the world to Map, and also done in order to facilitate testing of that code, as the Game class can not be instantiated in the testing environment (see testing report).
                - https://mrpoketes.github.io/mrpoketes.github.io-ENG1-Team6-Website2/docs-new/test2.html
    - The logic for updating penalties has been moved to Map, to enable it to be tested.
    - Map now takes a GameDifficulty in its constructor, which is used to determine how many obstacles to spawn.
    - A new function has been added that initialises the lanes using save data (as opposed to generating new ones).
- Lane Changes:
    - The original constructor has been modified to create power ups in addition to obstacles,and also takes a number of obstacles and a number of powerups to generate.
        - Fulfill requirements: FR_POWER_UP_ITEMS 2.0.12, UR_POWER-UPS 1.0.3
    - A new constructor has been added to create a lane from lists of already existing power ups and obstacles - this is used when creating a lane from save data.
        - Fulfill requirements: FR_POWER_UP_ITEMS 2.0.12, UR_POWER-UPS 1.0.3
    - When generating a new lane from obstacles, obstacles and power ups are no longer generated for the full length of the map - instead they're placed up to (and slightly ahead of) the finish line.
        - This ensures that lanes are fair, as previously a lane could have more or fewer obstacles depending on how many were created above the finish line.
- Obstacle Changes:
    - A handful of fields have been added to store construction information, which allow it to be saved later.
- Boat Changes:
    - The boat's type (which determines its base stats and sprite) is now stored in a field to allow it to be saved.
    - A new constructor has been added to create a boat with a current speed and stamina value - this is used for creating boats from saves.
- AI Changes:
    - Corrected a bug where AI boats were being initialized with the wrong values.
- MenuUI Changes:
    - Buttons leading to the new instructions and options menus were added.
        - Fulfill requirements: UR_DISPLAY_INSTRUCTIONS 1.2.3, UR_CHANGE_SETTINGS 1.2.4

    - The play button now leads to the difficulty select UI instead of the boat selection UI.
        - Fulfill requirement: UR_CHANGE_DIFFICULTY1.3.2, FR_CHANGE_DIFFICULT 2.0.13
    - A button to continue the saved game was added. This button does not appear if there is no save data.
        - Fulfill requirement: FR_SAVE_&_LOAD 2.0.11
    - When there is save data, the exit button is moved to make room for the continue button (which occupies the space that the exit button used to be in).
        - Fulfill requirement: FR_SAVE_&_LOAD 2.0.11
- ResultsUI Changes:
    - The times and penalties for each boat are now rounded to the nearest integer, instead of being given to 8 decimal places.
    - Each boat's time reduction (from powerups) is shown next to their penalties, and is subtracted from their time.
    - The results menu now tells you to click to continue to the next leg, or wait for the other boats to finish if the current leg is still ongoing.
- No significant changes were made to the Player, UI, ChoosingUI, GameOverUI, GamePlayUI, or ScrollingUI classes.

Unimplemented Requirements

NFR_FAST_TRANSITION - the transition time when entering the racing state is ~1.5 seconds, which isn't under 0.5 seconds.
We were able to reduce the transition time by about 70% by staggering the creation of the three maps, but to reduce it further we would need to decrease the time taken to load a map. This is not possible due to limitations of the library used to create physics bodies for the objects on the river.