Module	Engineering1 (Eng1) - COM00019
Assessment Title	Assessment 2, Cohort 2
Team	Quakthulu (Team 16)
Members	Aaron Price, Alec Coates, Charlie Curedale-Rayner, Eleanor Griffin-Smith
Deliverable	Implementation

Implementation - Part B

Requirements and Architecture Implementation:

UR_MAINTAIN_PREVIOUS:
FR_MAINTAIN_PREVIOUS - we have not drastically changed the previous teams code in any major way. All of their code still runs in the way the previous team intended.

UR_CHOOSE_DIFFICUTY:
FR_DIFFICULTY_SCREEN - when first starting the game the user is given a choice of Easy, Medium or Hard. This decision affects the number of obstacles that are generated in each lane and subsequent leg of the game.

FR_DIFFICULTY - the selected difficulty from the difficulty screen is 1 for easy, 2 for medium and 3 for hard. This integer value is then multiplied by the leg difficulty and this becomes the number of objects that are created.

UR_SAVE:
FR_SAVE_SCREEN - during a race a user can press the escape key to bring up the pause menu, and select the save/load option. This in turn brings up a saving and loading screen where the user may save up to 3 times. They also have the option to delete saves.

FR_SAVE - our implementation uses a JSON file to save our game data. If the option to save the current state is selected from the save screen then a SaveLoadGame object is generated. This object, using the current state of the game, goes through all the necessary data structures and either calles their saveState method, which converts the object to a json friendly format, or directly saves the data to the json file. Depending on the save slot chosen in the menu a file with an appropriate name is generated and filled with the collected data.

UR_LOAD:
FR_LOAD_SCREEN - **SEE FR_SAVE_SCREEN**

FR_LOAD - upon the selection of a load button a previously described save file is loaded up and iterated through setting each piece of data to a new object of the same class as it was saved from.

Significant and new features:

Sprite Descriptor classes - The libGDX library has a json class that we utilized to save out game data. The only problem is that the json class is only able to save primitive data types, thus we needed a way to represent a class of complex variables as a class with only primitive variables. We researched the idea of a sprite descriptor class where each class we wished to save has a subclass of only primitive types that could be converted to the json format by the libGDX class and saved with ease.

Significant changes made to the previous software:

We have made no significant changes to the previous teams code. All of the code we were given still functions. The only changes we made were adding our own code and thus nothing has changed.

Not (fully) implemented:

Power Ups - Due to our lack of teammates the requirements of power ups has been dropped from our requirements and implementation.