SEPR 2019/20 Assessment 4

Team CheatCodez

Jonathan Grout, George Lesbirel, Cheuk Wang Wu, Lauren Quarshie, Muaz Atif, Lillian Coultas

Implementation and report

New Feature implementation

Requirement	**Changes to GUI**	**Changes to Code**
Powerups	For the pickups, we decided to make the trucks drive over them to pick them up as opposed to a score based unlock or something similar. This means we have added pickups spawning on the roads. To show that a pickup is active we have added a column of 5 boxes in the left hand side of the screen. When a pickup is active, it will fill one of the boxes.	Each type of powerup is stored in an enum which is passed into a class which controls interactions with the powerup. The powerups applied to a firetruck stack ontop of eachother. We save these in an Array that can be iterated through.
Difficulties	Our implementation of the difficulties allows the user to select a difficulty before starting the game. From the main menu they are able to choose: easy, normal or hard as their difficulty.	Variables that are tweaked by difficulty are multiplied by a difficult constant. This constant varies depending on the difficulty selected at the main menu. For example, selecting Easy will half several enemy stats (* 0.5)). Each difficulty is stored in the constants file.
Game Saves	For the game saves we added a new screen that had the ability to interact with the game saves. Each save is shown as a button on the screen with an exit button. The screen takes an “action” parameter which is either save or load which determines the action taken when a save is clicked. The pause menu has “save game” and “load game” buttons added to it.	In order to make the game saves, we use LibGdxs inbuilt Json library. Each class we want to save implements “Json.Serializable”. This allows us to make a Map of the values from each object we want to save. We only use the writing method as we could not get the loading function to work properly. Each object we want to save is then put into an array. We can then read this back by loading the array and processing each json object inside. Each object is put into a respective temporary array and then passed into a method which loads that type of object. The current game has the ability to save 5 different

		games, this was done to reduce clutter in the file system and we did not see a need to save more.

Extra Features

Requirement	**Changes to GUI**	**Changes to Code**
Rewards System	The reward system has added a new label to the game screen displaying the current task and the time left to complete	There is an array containing the current achievements in the game screen and a new Achievements class. This class is responsible for everything to do with the achievements. The class is easily expandable as adding different types of achievement can be done by adding a new number to the checking method and a new method for checking. New Achievements are created every 30 seconds if one is not already active

Achievement format:

- Destroy X Fortresses/Patrols in Y seconds, where X is the number of objects to destroy and Y is the amount of time the user has.
- To expand, you just need an achievement with a numerical aim and a time limit. This could be “Drive around the map in 20 seconds” and then monitor the users position. In the code you would add a new type (integer in the constructor) and a new method check{NewAchievement}(). Then invoke this method in the checking method.