MiKroysoft

Module	SEPR
Year	2019/20
Assessment	2
Team	MiKroysoft
Members	Daniel Crooks, James Rand, Irene Sarigu, Alfie Jennings, Charlotte Clark, Jasper Law
Deliverable	3 Implementation & Report
Website	https://mikroysoft.github.io/

To run the game, open the executable jar folder and run the jar file. The accompanying map information file must be located within the same folder as the jar.

(a)
Find documented code here: https://mikroysoft.github.io/assessment2-source-code.zip
And javadocs documentation here:
https://mikroysoft.github.io/assessment2-documentation.zip

(b)
In our implementation we have managed to successfully fulfill many of the requirements we set out to achieve. One requirement we found more challenging however was the requirement listed with the id NFR_CONTROLS. This requirement was to ensure that the game could be played on either a touch screen or keyboard to allow portability between a desktop computer and a phone device as shown in another requirement with the id UR_MOB. Our reasoning for not being able to do this is that we needed a back button to go from the game screen to menu, as per the requirement FR_ESC. Whenever we tried to click on the button the fire engine would move towards the button instead of triggering the button. It is theoretically possible to write some code in the InputController class to fix this issue but we discovered this issue late into the development process. With this constraint of time we decided to instead make it so you could exit the game screen and into menu by pressing the escape button on a keyboard. Unfortunately this does conflict with the requirement NFR_CONTROLS which states that we must only use either a touch screen or mouse.

The requirement FR_INCORRECT was not implemented correctly, this is because we were unable to identify a way to effectively save the game objects in the game screen. For this reason when the 'ESC' key is pressed the game goes to the menu and the game data is lost. This means that neither FR_WIN or FR_LOSE are met. And the requirement is therefore not met. To complete this requirement we must find a way to save the game data so that the game can be restored if the user wants to return to their game. As well as saving we must also write the implementation for restoring the game, instead of reverting to default values.

FR_TUTORIAL is also unimplemented This requirement consisted of providing a tutorial at the start of the game. This proved a difficult problem to solve in a manner that satisfiable as our original vision for this requirement was ambitious. We wanted the game to start off in the game screen environment where an interactive tutorial starts that the user could skip. This would not only have required more code to restrict the user so they could only do what the tutorial wants them to do but also more graphical work to provide more sprites to display the game functionality. This would obviously increase the workload quite significantly so we did not have enough time to fulfill this requirement. Furthermore, we were concerned about how this type of tutorial may impact other requirements such as UR_TIME as this more in depth explanation would be more time consuming for the user meaning they may not be able to play the game in 5 minutes or less. The solution for the tutorial we went with was an instruction document that can be displayed when clicking on the instruction button. We felt that although this was not the ideal solution; it was an effective one as it quickly explains how to play the game and the user can skip it. The only problem was that it is not at the start of the game and therefore does fulfill the FR_TUTORIAL requirement.