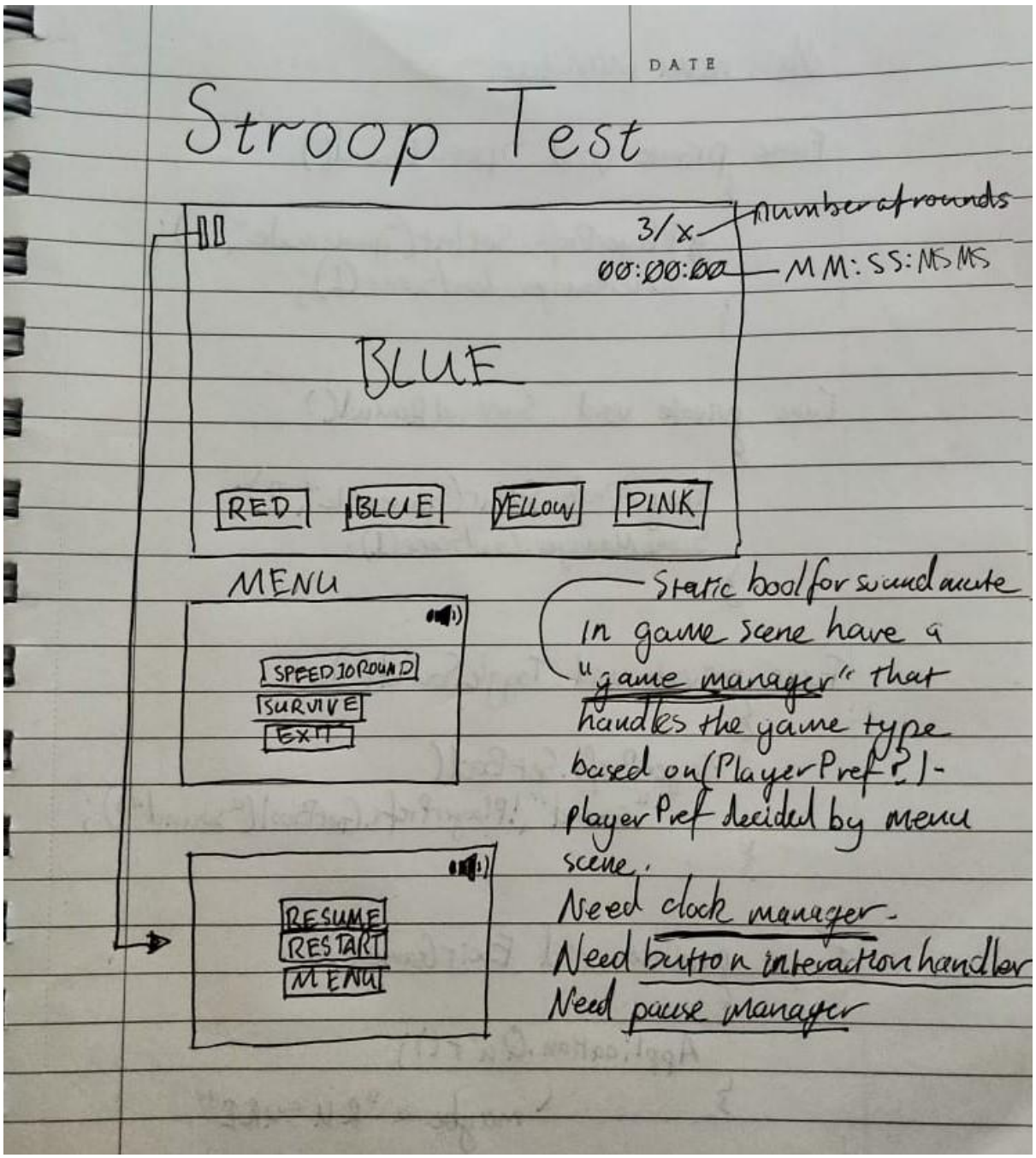
Mighty Kingdom

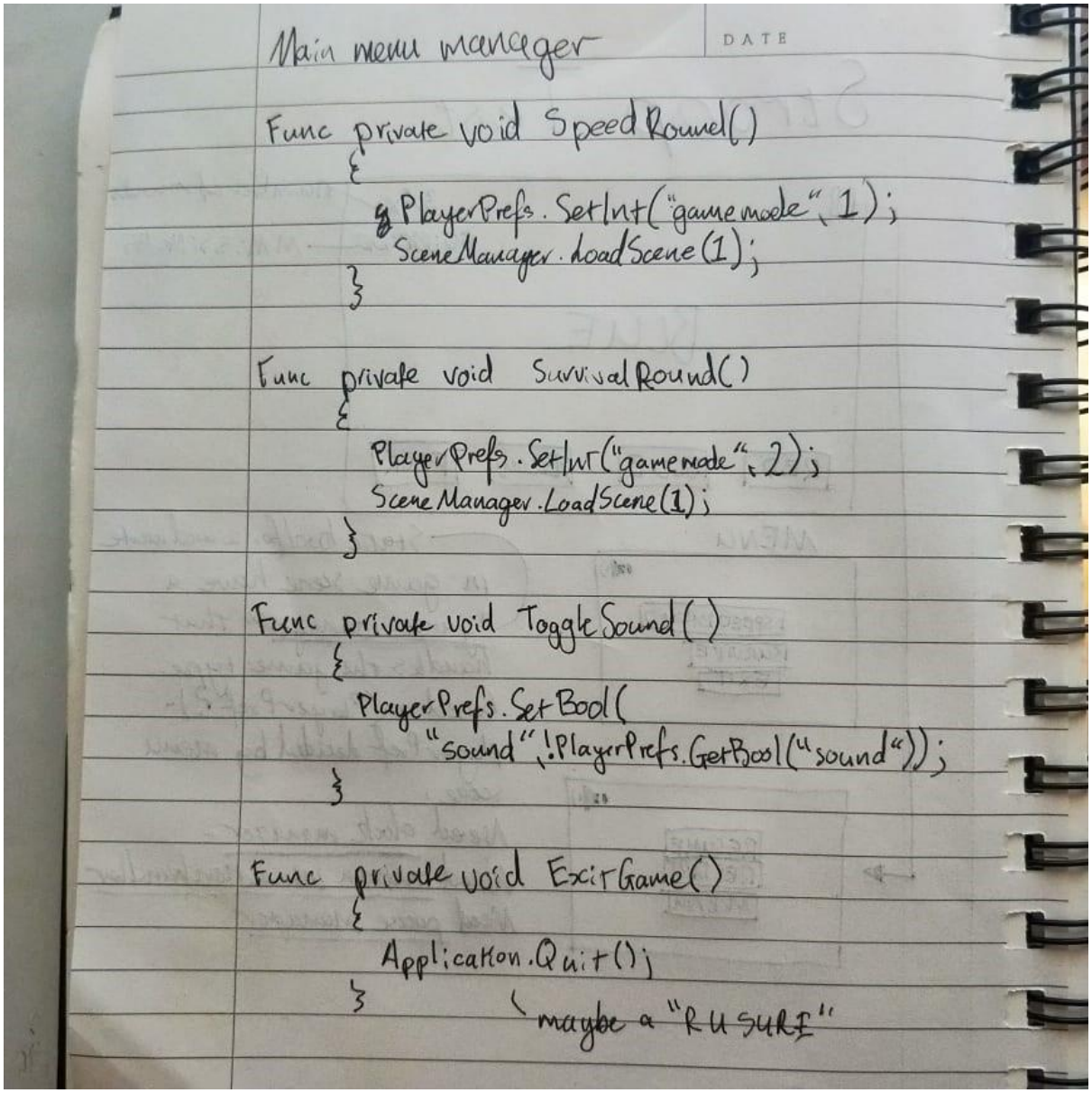
Graduate Programmer Test

Ben Ioannidis

Initially I did some basic UI plan drawings, and laid out the scripts I needed for managing gameplay, as well as handling program flow (menus).



Initially I had planned to use the singleton pattern, and have my Game Manager script be a persistent object across scenes and use player prefs to set and check the game mode (as an int), which can be seen in both the above and below images. I decided against it as I have implemented this in my 3 previous projects, so decided to try a different design pattern, using a static class (the GameModeData class).



I created a checklist on my whiteboard outlining the features to be implemented and the scripts required for the implementation.

I started by working on my GameManager script, where I first set up a "colour" enum for ease of colour selection via the editor.

Along with this, I created a hashset with the editor selected colours to ensure distinct colours in the list (remove duplicates), then copied the colours back to the colour list.

I setup the static GameModeData class to hold required information on the current gamemode for the GameManager to be able to reference, parameters such as the modeName, and the score limit for the mode are accessible.

I also created the basis of the functions required for button presses, and handling the comparison, and outcome of the user answer.

I started setting up the basic UI I was going to use, and hooked it up to my GameManager script, then ran through some quick testing to ensure everything was working the way I had intended.

I found and fixed an issue caused by me copy-pasting code, where the button event functions were all checking whether colour.Red was correct.

I decided I could easily use a small amount of UI elements, just changing the text based on the current gamemode in the GameModeData class. This led to me making the addition of a description for the gamemodes, that will be presented to the player before starting their selected game.

Following on from this I tested the speedround in it's entirety with the addition of the stopwatch I implemented, then started working on the menu scene and the survival mode, which was a little trickier.
I created essentially the opposite of the stopwatch, the timer, which would count down instead of up, and had the time being checked in Update where the game would be ended once it reached 0.

I setup the basic script to handle the menu scene button press events.

Following this, I added in the audio clips, and created a script for muting/unmuting which meant also adding a new bool to the GameModeData script, this way it could be set and checked both in the menu, as well as the pause menu in the game scene too.

I fixed a little bug with the score at the start of survival rounds reading: "0/0" and saved the project for submission.