Manual Testing

Manual test-cases were done for:

- Appearances of all visual elements (size, position, colour etc.) such as sprites for buttons, chefs, customers, items and tiles
- Effects of powerups (does it have the right effect, does it do too much / too little etc.)
- Customer behaviour (movement speed, which orders appear)
- Button clicking (does it click when the cursor is on it, does it click when the cursor is off it, what happens when it is clicked etc.)
- Controls (does the key do what was expected when (not) pressed)

Our **manual testing** was done using, where possible, a small simulated world, which we called “miniTestWorld”. This essentially consisted of a blank world in which we rendered visual functions of the game, such as drawing the chefs, making sure that the directional drawing worked, making sure that the food sprites drew correctly, and making sure that the chef models reacted appropriately to picking up items. This was all interactable by the tester, making sure they could actually check and isolate different specific functions, and see if there were any errors. We were not sure if such tests would be necessary, as we could theoretically assert that as long as the working variables behind the sprite drawing functions were as expected, then everything must be working, but we considered that this would be a better option to make sure that we could more reliably say that the code worked as expected and intended.

We also had to do some testing simply in the game, both due to time constraints and to us it seemed like the most efficient way of conducting the tests. These included the powerup function tests, making sure that they did as expected and intended, and aided in development of them as a whole. This possibly wasn’t the most effective or rigorous way of going about it, but due to time constraints we did what we could to have what evidence we could get.

Generally we feel like our tests cover the majority of testable functions and eventualities possible in our game, and this has helped us move forward with the project, by giving us understanding of, and the ability to improve the given code. We think that we’ve struck a good balance between prioritising automatic/ unit tests and using manual tests where necessary, such that our tests are fairly complete and correct. We tried to keep the scope of all of the individual tests to a minimum where we could, as this allowed us to identify errors and problems more easily, which was especially useful when working with the station tests and working out how they interacted with other entities in the world.