Continuous Integration

Cohort 1 Team 9

Dominic Hall
Firas Marzouk
Ben Morrison
Harry Whittaker
Amelia Wigglesworth
Zehang Li

5. (a)

As a group we decided it would be useful to use a continuous integration method to ensure that implementation of code during the programming of the game was done in a smooth and safe manner. As there are multiple programmers adding to the code at once, it is not only easier for us to use a continuous integration method, but safer to ensure no code is overwritten or deleted by mistake that may have severe consequences with the execution of the code.

To summarise our working method, we have a central main code that is stored online using the version control software, Git; each of us can create a copy of it to work on locally so as to not directly edit the central code. Once we are happy with the modifications made, we attempt to merge it back with the main code which is done autonomously. If there are no conflicts it will merge quite simply, and update for everyone. However if there are conflicts, for example when a part of the code fails a unit test or is modified differently to the main code that breaks something, or such as two programmers modifying the same piece of code, then it gives an opportunity to rectify any problems caused and ensure the optimal code is written and merged within the main.

5. (b)

Our method of continuous integration consists of using GitHub to ‘branch off’ (make a local copy of) the main code, and for each of us to individually modify and implement changes to the code simultaneously without creating immediate and lasting conflicts. We chose to use GitKraken to better visualise the branches currently active as people worked on them, so we could see what point the code was at when the branch was created, and easily compare to see the changes made.

Using this method of continuous integration, we are able to keep a record of each significant modification made, and add an extra layer of security to the main code. Changes made are checked over by a different programmer before they are merged into the main code to ensure quality and effective programming is withheld, and that changes made are correct and run without errors. Any conflicts that are created between branches (e.g. if two people edit the same piece of code) when it comes to merging the code, are autonomously flagged and can be easily compared and rectified with few issues. This preserves the integrity of the main code.