Continuous Integration Report

Team CASEUS

(Team 7)
Eliot Sheehan
Matthew Turner
Daniel Atkinson
Hannah Pope
Mayan Lamont
Divyansh Pandey

a)

One of the main components of continuous integration which we adopted was the use of a single source repository which all members had access to. To make changes to the code, a member would fetch the latest version from the “main” branch in the repository and make adjustments to it locally. As the member implemented their change, they would regularly push it to its own branch on the shared repository so that the rest of the team could see how they were progressing. Once finished and any found bugs had been fixed, the member merges their branch into “main” (getting confirmed from the rest of the team if needed). These changes can be viewed by the rest of the team, with details such as time committed, and what code was changed. This method worked well for us as it meant we could maintain a stable branch that was playable while allowing multiple features to be worked on in parallel. Additionally, having the information of recent code changes helped keep each member up to date on the current version of the code.

We also made use of locally run tests to implement continuous testing. This aided team members to identify bugs in their code before merging it into the “main” branch, helping us keep “main” in a stable state.

The executable is located in the repository, and is easily accessible and downloadable if it is needed. This executable is also re-compiled after every commit manually, as we did not have an automated build process.

b)

To implement the single source repository, we created an organisation on the website on Github, which held our code, our tests and other relevant files. Each member of the team had full access to this repository. This meant we could see which users had modified specific code. We used the desktop application to retrieve and commit between Github and Intellij (our chosen java environment).

Our locally run tests were run straight in the Intellij environment. These ensured we could always test our code independently on our own workspaces, before pushing to the repository. The tests were held in a ‘test branch’, so that all team members had easy access to the most up to date test classes.

Along with our code, all relevant files were included in our github repository, including the game executable. After every commit we manually updated the executable (as we did not create a more optimal automated build process).

Occasionally, we used more than one branch in order to develop features that could have a significant impact on the current code. This was done to ensure frequent commits could still be carried out, without affecting members working on the main branch. These additional branches were then merged once development and testing had been completed. An example of this is when one member had a separate ‘boost’ branch, to avoid interfering with development on main whilst working on the boost feature. Once completed this branch was then merged back into the main branch. We also used the technique while working on the ‘saving’ feature.