UNIVERSITY OF YORK

DEPARTMENT OF COMPUTER SCIENCE

Continuous Integration

Group 15 - HesHus

Adam Robinson

Tsveta Ivanova

April Jack

Oliver Goodwin

Luke Jackson

Luca Hammond

Chris Rolfe

<u>**Continuous Integration Report**</u>

For our continuous integration, we are aiming to integrate work at least once a day, which allows us to stay up-to-date on each other's work, helping to avoid errors. We can also use the day-to-day approach to break up the workload to fit this schedule and make the overall implementation process more manageable, an issue we faced during the first assessment.

To implement this, we will be using Github Actions. This is the most useful method for us as we are already using GitHub pull requests to share our code, and Actions can be seamlessly integrated alongside this so there is no major disruption to how we are already working. Furthermore, it is suitable for the approach outlined above as it allows us to automate the building, testing and deployment of our code. It also provides useful features not outlined, such as a server to run code on, meaning that there is no problem with using different machines and OSs, as all code must be built to work on a singular server.

We made sure that any time the main line did not run as expected, for instance the actions workflow not building, it would be fixed immediately. This was to make sure that we always had a stable version of the game available. For issues such as code being broken by a new merge onto the main, the project tests itself using our Junit tests automatically on every commit. If the build is broken, everyone that is a contributor to the project will be notified, allowing someone that may understand the reason behind the fault to fix the project.

To assign individuals to certain tasks, we used Github issues. This allowed us to keep track of what everyone has completed, as well as distribute work equally to create a steady workflow.

Continuous Integration Infrastructure

The foundation of our Continuous Integration relies upon the version control system, Git, in order to streamline the implementation process.
The main method of implementing CI will be through Github Actions. Whenever a pull request or a push is made onto the main branch, the actions workflow checks whether the program can be run on the three major operating systems (Ubuntu, Windows and Mac). This is done using the corresponding Github runner. Alongside this, the workflow checks the tests directory of the project and produces a Jacoco report on the tests, so whenever a new commit is to be made, the author knows whether it has passed all of the tests or whether they have broken another part of the code. If the commit is to the main branch, a jar file will be created for the newest version.

The general structure of the workflow is as follows:

- Actions workflow receives a push or pull request.
- Actions checkout grabs the code from the directory.
- It then sets up the correct versions of JDK and gradle.
- The Github runners check the compatibility of the OS and runs the tests.
- The workflow then returns a Jacoco report depending on the results of the tests.
- A .jar is made of the game and uploaded to the Github's release page