Module	Engineering1 (Eng1) - COM00019
Assessment Title	Assessment 2, Cohort 2
Team	Quakthulu (Team 16)
Members	Aaron Price, Alec Coates, Charlie Curedale-Rayner, Eleanor Griffin-Smith
Deliverable	Continuous Integration Report

NB: Due to our small team this was dropped from the assessment

Continuous integration methods

Frequent integration: The main feature of continuous integration is frequent integration. This is the process of merging code often so there will be more common code between the developers. This helps reduce the risk of conflicting code when merging versions of code.

As sections of new code are developed, they will be committed to the code base at logical intervals. E.g. halfway or when a section of a requirement is completed. Other developers can then use the updated code base for further development so it will be easier to integrate new changes. The reason we decided to commit at logical intervals instead of multiple times a day is because we are a small team with fewer requirements than others therefore less changes are happening simultaneously.

Version control system: A VCS is used to keep track of code changes. This allows developers to be able to access the newest versions of the code. We chose Git as our VCS because it is the most widely used and as such has a lot of documentation and is very reliable. We chose GitHub to host our Git repository because it is the service that all of our team members are most familiar with and already had accounts set up with for easy access.

All artefacts for the game will be stored on the GitHub repo so that continuous integration builds always run off the newest code. This helps our development team to keep up to speed with each other even though we are working separately.

Automation: Turning raw code into a distributable game can be complicated so it is better to have this done automatically with a build tool.

For our project we have decided to use gradle because we have used it before, and it works with Java. Building our project automatically saves time and allows for quick personal testing when working on code.

Testing: We use several methods of testing to catch and diagnose bugs and errors throughout the program.

When committing a new version to the code base, tests are run automatically to ensure integration is successful. Also, when the final game is produced it is tested with a mix of manual and automatic testing.

Integration machine: All previous integration methods should be performed by an integration machine. This is also responsible for notifying the team on the success of builds.

Continuous integration infrastructure

GitHub:

- GitHub has been used as our integration machine
- It allows team members to work on their own separate branches from the main branch and then merge them back.
- It also keeps a track of push and pull requests to branches and flag potential merging errors.
- The feature Network graph can also help team members visualise the different versions of the code merging. See figure 1

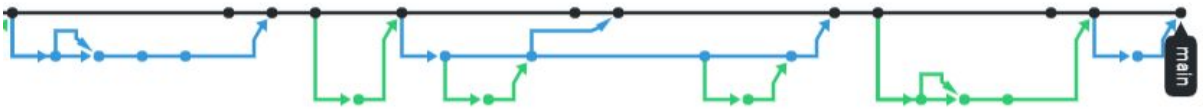

Figure 1 – A section cut from Network graph from our GitHub repository

- GitHub also has badges which we use to track the status of builds and the code coverage. This is displayed at the top of our repository. See figure 2.

Figure 2 – Badges displayed on our GitHub repository

We use Github Actions to generate our CI reports because it integrates well with our VCS, also being Github. The first CI workflow checks that the current implementation builds. It first runs the unit test because they are lightweight and cover a wide portion of the code. It then runs the integration tests because they take much longer and only test the file saving. It finally then tries a full build of the program and passes if all steps have passed - if any steps fail, it won't bother executing the rest to save on compute time.

The next CI workflow checks the code for flaws and vulnerabilities against Github's list of vulnerabilities.

The third workflow (showing the third and fourth badges) checks the test coverage. The first badge shows branch coverage of all the classes that have >0 tests executing on them and the second badge shows instruction coverage of all the classes that have >0 tests executing on them.

The fifth badge shows the number of open pull requests, making it easy to see if there are any changes waiting to be merged.

The sixth badge shows whether the Github website is up. This helps indicate whether there are any problems with changes to the site.