Continuous Integration Report

Babar Khan
George Rogers
Jacob Turner
James Crump
Chloe Remmer
Shijie Lin

1 Continuous Integration Methods - Part A

Continuous Integration refers to the practice of integrating tasks into the whole system immediately upon completion. After any integration, it is important to check if all of the tests for the system pass. This means that it is easier to spot where an error may be and what snippet of code may have caused the problem.

Since our team is using an Agile methodology (SCRUM), continuous integration is a key part of our sprints as we require frequent updates to the system. We have multiple people working on developing the code for the game, therefore it's important that whenever one of them has completed their task for the sprint that they then merge their snippet of code and test it. This means that if there are any errors after the merging of the code, that person knows why that is and can warn the other team-members about the error so they can wait to merge their code until after the error is fixed. If there was no delay in the merging of code after an error, there'd be more code for the Software Development team to look through to detect that error and fix it or they may then have multiple errors to contend with. Considering we have a tight deadline and short sprints, it's important to waste as little time as possible and our methodology would make the continuous integration process more efficient.

Above is an example of what outputs would be for the continuous integration tests. Any tests that have failed are marked with a red 'x' whilst tests that have failed are marked with a green '✔'. This makes it easy to see whether or not a test has failed.

2 Continuous Integration Infrastructure - Part B

Our group decided to use GitHub to develop the code and allow for multiple people to collaborate on it at once. A key benefit of using GitHub is that it allows for automated Continuous Integration through the use of GitHub Actions, which allows us to create custom continuous integration workflows directly in our repository. We opted to use Github Actions as it works on any operating system and with any programming language.

As we have been using Java as our primary programming language, we have chosen to work with the Gradle build system, compared to any alternatives such as Maven, as Gradle is known for being incredibly flexible. This is useful as we have created our own unit tests.

When we commit code to our repository, we can continuously build and test the code automatically. Our workflow is designed to build our code and run the tests whenever new code is pushed to our repository. As stated in the above section, it is important for us to consistently check every new bit of code added so that there only small amounts of code needs to be debugged if any errors occurred. When the tests are run, we can see the results of the test in the pull request and check if there are any errors. If it passes, then the changes made will be fully merged.