ENG1 - Assessment 2

CI Report

CIReport.pdf

Group 4

Mikaella Loppnow	ml2708
Tom Daly	td1026
Ethan Buss	eb2225
Dillon Pandya	dp1195
Ereife Odusi	ed781
Harriet Kirby	hk1114

Methods and Approach

In our project, continuous integration was used heavily in our agile development approach to our 2D game. This meant as we added new features to our game, our code integrations were continuously tested using completely autonomous testing techniques. This meant minimising the chance of introducing new errors. The reason we implemented the CI methods and approaches is because members of our team may work on the project on their own for a long period of time and when they eventually merge it back to the master branch using GitHub which is where we stored all of our files on a shared repository, it made it merging the changes difficult due to the potential of code duplication and other code being overwritten which could lead to many bugs if it was not dealt with properly. This meant putting the proper infrastructure in place to ensure this did not happen.

Our approaches consisted of the following:
Automated Testing: By automatically running unit tests on the new commits allowed for immediate feedback on the new code. This meant if there were any issues with the code they were immediately shown so that the team could revise the code that did not work and try it again.
Build Verification: By being able to push new builds automatically, it ensures that our game is always playable. By only pushing out versions that work and automatically increasing the build number. This prevented any human error in build creation as all components and dependencies of the application fit together correctly. This prevents integration issues and ensures playability of the game.
Code Quality checks: We implemented code quality checks in order to ensure the high standard of our code. We did this by implementing static code checks to check for any code quality issues.

These CI approaches are triggered on every push to the shared repository's main branch, as well as pull requests. This made sure any and all changes were verified before being integrated into the system and uphold the integrity of the game.

By implementing the following approaches listed it enabled the team to have rapid feedback on any changes made to the code base. This means any changes to the code that could perhaps have broken or created a bug in the system are brought up immediately. This is particularly useful when there are multiple people contributing towards the code. By automating these mundane and repetitive tasks, such as builds and tests it meant we could focus on actually adding to the game and more on development of the system rather than manual testing which would take up a lot of our time and make sure we kept on time for the duration of the project. As our code base also grew with the addition of new features like the scoreboard and streaks, our CI applications were also scaled up to include the complex tests required to check if these features worked. This meant that even though our system was constantly and rapidly evolving and growing our CI approaches were able to meet the needs of the project.

The CI methods we as a group implemented were a foundational part of our development cycle that improved the quality of our game and the reliability of it too and enabled us to make a fun and engaging 2D game for users to play.

CI Infrastructure

For our 2D game, we set up our continuous integration infrastructure through the use of GitHub Actions. By using this tool we were able to automate our methods and approaches we outlined making sure all the commits and merges were functional and buildable for the main branch.

We chose GitHub actions due to the fact that it is already integrated with our shared repository which contains all the code for our game. This meant we can automate our workflow directly where we did our work as a team. Our infrastructure we set up included multiple workflows and actions that got triggered by specific events on our GitHub repository like a push or pull request.

CI infrastructure we have implemented:

Build Workflow: This gets activated every push or pull request on the main branch. It compiles all the code on the repository and makes sure there are no compilation errors and if there are it immediately gets reported to the user.

Test Workflow: This runs alongside the Build Workflow on every push or pull request on the main request. It executes the pre-defined unit tests to validate the code changes using JUnit. It ensures that all the code passes the tests and makes sure the code base functions as it should.

Version Control Management: Successful builds that we deploy automatically get a build number that increments from the previous version complete with all the log files and executable code.

Notification Integration: We set up, through GitHub's interface, email and discord notifications about push or pull requests and the success or failure of the continuous integration process. This immediate feedback meant for quick iteration and a quick way to fix any issues that occurred improving the reliability of our code.

Security measures: The only people that can manage the CI workflows and code base was our group in order to maintain the code base and ensure nobody else can edit the code compromising it's integrity. By allowing only authorised users to access the repository it meant our CI infrastructure was reliable.

Our group's implementation of our CI infrastructure using GitHub Actions meant the development of our 2D game was efficiently run by using automated workflows for every development phase. This meant at every stage our game was in a consistent state to be deployed and worked on. It meant our game was of very high quality standards at all stages of the development process. It also removed risks and increased the reliability of the project. The way we laid out our CI workflow enables new developers to easily take over and scale it to encompass new features if needed.