Continuous Integration Report

ENG1 Team 9

Jacob Dicken
Bertie Cartwright
William Croft
James Dovener
Henry Chan

Continuous Integration Methods & Approaches

When developing our game, we placed an emphasis on frequent integration, ensuring that when one or more team members were contributing on a given day, they would push their changes (and pull any other members' changes) at least once per day. This was followed in the majority of cases throughout the development process.

Early in development we set up support in our build system for generating automated test coverage and code style reports. Before pushing, any contributor is encouraged to generate these reports and make an informed decision on whether to add new tests or refactor their code prior to review. When performing code reviews, this process was also followed, with another member generating the necessary reports in addition to manually checking the code. Ensuring our code followed the same conventions on, for example, indentation and method naming made development significantly easier by reducing the mental labour involved in understanding another person's code.

On each push to GitHub, an automated workflow detailed in the infrastructure section was run - this provides a copy of the executable JAR generated, a code coverage report and a code style report, which are uploaded to the job results page and can be found on our GitHub. The results of this workflow generate a notification in a channel on our team discord via a webhook, providing a notification to other team members when new changes are pushed and allowing reviewers to see whether tests are still passing. This helped ensure that integrations happened frequently enough by reminding other team members to pull new changes.

We attempted to follow the principle of sticking to the main branch where possible, and where a feature took multiple days to implement we would merge the latest changes from main, ensuring no branch falls far behind as this would be difficult to integrate later.

We followed the principle of integrating small changes frequently as this lends itself well to an Agile development process and minimises issues with integrating entire systems at once. Implementing this system earlier in Assessment 1 would have made our development process run much more smoothly at several key points where integrating separate systems became a concern.

Continuous Integration Infrastructure

Early in development, we added support for JaCoCo[1] and Checkstyle[2] to our build system. These are tools that integrate with Maven to provide automated code coverage and style checking reports respectively.

We chose to use GitHub Actions to support our continuous integration processes. We implemented a workflow with three jobs - test, build and release. This workflow is triggered on each push to the GitHub repository.

Test

- Runs all automated tests with the maven test command
- Generate a code coverage report with JaCoCo
- Generate a checkstyle report

Build

- Runs only if the Test stage passes
- Build a copy of the current state of the project using the maven clean package command
- Upload a copy of the executable JAR generated

Release

- Runs only on commits tagged as a release, otherwise this job is skipped
- Generates a GitHub release and attaches the executable JAR generated in the build stage

The output from our GitHub actions workflow was also reported to a channel in our team discord through a webhook set up at the start of development. This provided a summary of the tests that passed and failed, and indicated whether each job was successful.

[1] https://www.jacoco.org/jacoco/
[2] https://checkstyle.org/