Continuous Integration

Cohort 2 Team 5

Harry Beaumont-Smith
Tom Nolan
Will Punt
Ruth Russell
Mimi Shorthouse
Lottie Silverton
Stanley Thompson

Our team adopted a continuous integration mindset, this in turn guaranteed the frequent integration of code changes into the main branch.

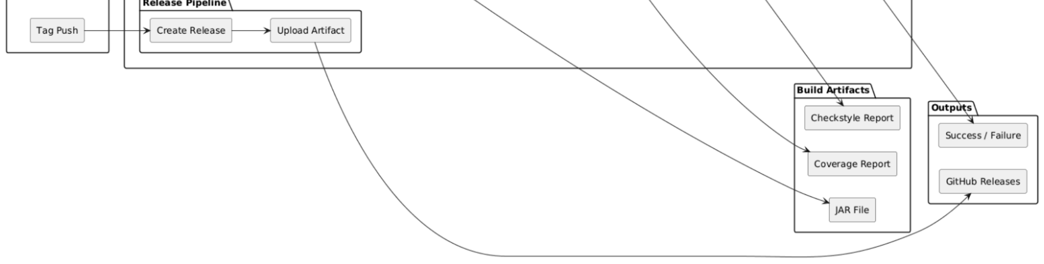
Continuous integration approach:

- Commits were pushed often to share progress as often as possible, as well as to reduce potential conflicts that could arise from integration. In addition, full systems were committed in place of partial integration, something that was done in order to prioritise testability over smaller incremental commits. Regular integration also in turn minimised the risk of later merge conflicts, enabling easier detection of bugs.
- Automated builds and self-testing were used to fortify our Continuous Integration. Each commit was set up to automatically trigger a build and to execute the created unit tests in order to ensure correct logic. Automated builds were treated with urgent care, in which errors were made a priority upon being outlined, preventing snowballing.
- Builds were executed in a clean, shared environment using Linux and Gradle to avoid “works on my machine” issues. This ensured reproducibility and consistent behaviour across all team members' systems’.
- Another crucial aspect of the approach was visibility and accountability. GitHub actions enabled a transparent view of the build status, alongside the commits pushed for each build. This in turn enabled the team to be able to identify failing commits, in turn improving collaboration and accountability.
- Visibility was something we brought into the project midway through the project and significantly improved integration awareness. It also in turn supported a more disciplined and collaborative approach to integration and was something we maintained for the remainder of the project.

Overall, this CI methodology was well suited to the nature of the project. It allowed the team to regularly check that the game was built and in addition ran correctly after the implementation of new features. While some practices were adapted to suit project constraints, such as less frequent but larger commits, the approach effectively addressed integration challenges and supported stable development, justifying its suitability for the project.

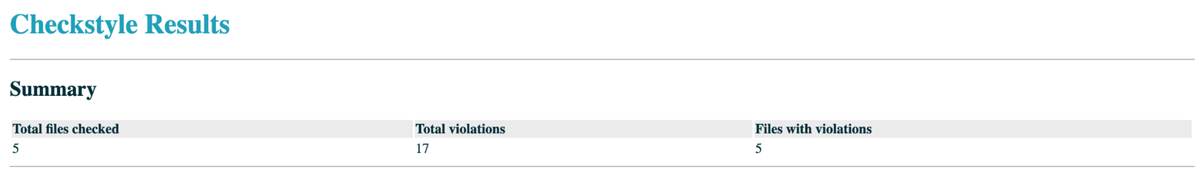
<u>Continuous integration Infrastructure</u>

The Continuous Integration infrastructure for this project was implemented using GitHub Actions integrated with the project’s GitHub repository. This resulted in seamless integration, enabling the CI work flow to be monitored alongside the source code.



- The workflow was configured using a YAML file and automatically triggered on pushes to the main branch, displaying the build progress and results on the GitHub actions.
- Gradle was used to automate the build processes, allowing the game to be built entirely from the command line without relying on an IDE. Moreover testing from the command line, ensured a consistent build both locally and within the pipeline.
- The CI workflow also produced build artifacts, accessible to the team, with tagged versions of the game being automatically uploaded to GitHub Releases. This ensured tested and stable versions of the project were easily identified.

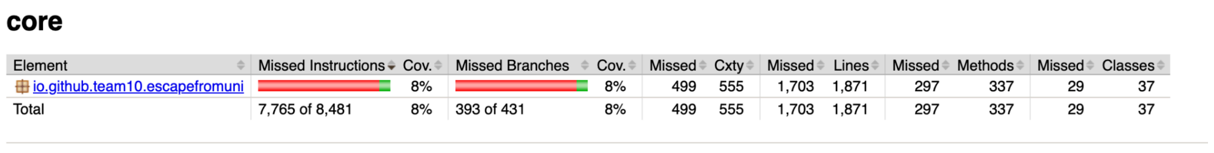
Automated tests were used to ensure new implementations did not break the existing functionality, validating core logic systems such as the achievement manager. JaCoCo was used to collect test coverage data, providing confidence that key areas of the code were exercised. In addition, CheckStyle was integrated both locally and in the CI workflow automatically enforcing coding standards on each commit. This was something that in turn improved the code's readability and maintainability. Below are the JaCoCo test report and CheckStyle for the most recent build.



Checkstyle Results

Summary

Total files checked	Total violations	Files with violations
5	17	5



core

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
io.github.team10.escapefromuni	🟥🟩	8%	🟥🟩	8%	499	555	1,703	1,871	297	337	29	37
Total	7,765 of 8,481	8%	393 of 431	8%	499	555	1,703	1,871	297	337	29	37

Other tools such as Jenkins and GitLab CI were considered upon deciding the tools to use. While Jenkins was a mature tool with extensive plugins, such as emails when things break and GitLab had integrated CI longer than Github, we did not choose these options. We instead decided to settle with GitHub Actions, as a result of its aforementioned seamless integration with Github repositories and ease of configuration. Overall, the work flow provided a stable, automated and most importantly maintainable process that supported the team throughout development.