CI Summary

Our team's CI infrastructure methods involved frequent commits, almost daily merges and automated testing to ensure the smooth development of our product.
Few commits were made per day on average based on the scope of work done within the commit. Merges were made almost daily to ensure that testing could happen before code was committed to the main production environment. These merges were viewed and accepted by another developer to facilitate collaboration and ensure code quality. A build file was used to automatically build and test the code to ensure that it is all still running as expected.

CI Infrastructure Report

The product's CI infrastructure uses JUnit tests to facilitate the automated testing of players, assets and LibGDX in the context of the product. It is implemented using a YAML file stored in the Github repository and GitHub Actions. It is relatively lightweight because of the scope of our project.

The infrastructure is comprised of GitHub for version control, GitHub Actions as a CI platform and JUnit as a testing framework. GitHub is beneficial because it is cloud-based and allows commits to be tracked and reviewed and for developers to return to previous versions if need be. GitHub Actions pairs well with GitHub and allows for the CI server to remain within the existing tools used for the project. JUnit allows for specific components of the product to be tested.

CI Process:

1. Developers make a pull request once changes are made
2. Developers initiate a push request to the shared repository on GitHub
3. Changes are reviewed and accepted by another developer
4. GitHub Actions trigger CI flow
    a. Install dependencies
5. Tests are executed
6. Merge request made onto the main development branch

The aim is that this process happens daily or otherwise as frequently as changes are made which ensures that code is integrated into the project with minimal errors.

Some benefits of the CI process outlined are that daily commits ensure testing is done daily and errors and bugs are identified and fixed earlier. The process is also not overly complicated given the limited scope of our project. It is also all cloud-based which means that the project is easy to maintain from anywhere. FInally, because testing is so focused, it allows for more extensive development as only specific components are tested at a time.

References:

[1] GitHub, "Automating builds and tests with GitHub Actions (Four steps)," Feb 02, 2022. [Online].Available:https://github.blog/2022-02-02-build-ci-cd-pipeline-github-actions-four-steps/

[2] Octopus Deploy, "Difference Between CI and CD," [Online]. Available:https://octopus.com/blog/difference-between-ci-and-cd

[3] Shiksha Online, "CI/CD Pipelines with a real-life example: Implementation and best practices," 2024. [Online]. Available: https://www.shiksha.com/it-software/software-development/articles/continuous-integration-ci-blogId-148355