CONTINUOUS INTEGRATION

Cohort 1 group 11

Freddie Aberdeen
Mutaz Ghandour
James Given
Jasper Owen
Jungwan Park
Joe Reece
Ivan Shestakov

CI methods and approaches

CI pipelines

We have a single Java CI with Gradle pipeline in our project which automatically builds the source code and runs the tests we have written on the source code for 3 different operating systems: Ubuntu, Windows and macOS. We did this because we decided that having one pipeline would be easier to manage than having multiple pipelines, and it would save time that we would otherwise waste creating multiple pipelines when one could have the same effect.

Triggering the pipeline

The pipeline was made to trigger whenever someone pushed to main so that if any code that did not work was pushed to main, we would be quickly notified and could fix it quickly to ensure the stability of the main branch

Also, the pipeline triggers whenever someone submits a pull request to main because this allows us to ensure that any code that is going to be merged with main would be proven to work by passing our tests. This means that we can be confident that main will work after we merge our pull requests. This also means that any issues in our code can be identified and fixed before we merge the code into main, keeping it reliable.

Pipeline inputs

Our pipeline takes the following inputs:

For pushes to main:
- The name of the branch the push is being made to (hardcoded to main)
- A Version tag if one was included in the push
- The source code in main at the time of the push, including the pushed source code

For pull requests to main:
- The branch the pull request is being made to (hardcoded to main)
- The source code in the branch the pull request is being made from

For either scenario:
- The matrix of Operating Systems to run the commands on
- The commands run by the pipeline
- The permissions to read from or write to the source code
- For releases, the name and location of any artifacts produced in the build step

Pipeline outputs

For each operating system in our matrix, the pipeline outputs the following artifacts:
- The results of our tests (whether each test passed or failed)
- The jacoco report for our tests
- A JAR file that runs on that Operating System

This means that we can be certain that our source code works on all 3 operating systems, or if not, identify any issues that may appear on one operating system, but not another.

The pipeline also outputs checkstyle reports for as much of our code as possible, to allow us to make sure our code follows the Google Style Java guide [1] as much as possible.

CI implementation

Our Continuous Integration uses GitHub actions implemented using java CI with gradle in a gradle.yml file. The gradle.yml file has 2 main tasks: Build and Release.

Build task

The build task contains a matrix which contains the identifiers for each of the operating systems that it will run the steps on: Ubuntu, Windows and macOS.

It then uses a matrix strategy to run the steps on each of these Operating Systems to ensure that the source code works consistently on all of the Operating Systems, ensuring it works on any desktop. For each of these Operating Systems the YAML file:

- Builds the source code
- Runs all tests provided
- Generates the jacoco report
- Uploads the following artifacts:
    - The results of the tests
    - The jacoco report
    - The JAR file

The YAML file also generates one set of checkstyle reports for our source code, test code and lwjgl3 code. This is because the source code is the same on each Operating System, and it would be inefficient to check the same code matches the Google Java Style Guide [1] 3 times

Release task

The release task only runs if a version tag was pushed to the main branch and if the build task was successfully completed. It does this to ensure that only versions of the game that work can be released.

The release step downloads the JAR files for all 3 operating systems in the build matrix and puts the JAR files in the release page of the repository.

Sources

[1] Google Java Style guide:
https://google.github.io/styleguide/javaguide.html