Continuous Integration

Assessment 2

Group 6:

Igor Smollinski <is942@york.ac.uk>
Pranshu Dhungana <pd861@york.ac.uk>
Zack Tyler-Kyle <ztk500@york.ac.uk>
Phoebe Russell <pbr508@york.ac.uk>
Sanjna Srinivasan <ss3264@york.ac.uk>
Sam Savery <sgs527@york.ac.uk>
Ewan Hutcheson <@eh1776york.ac.uk>

Summary (3.4.4.5.a)

Our Continuous Integration implementation consists of two nearly identical workflows, one triggered on a git push event in a GitHub Pull Request, and the other triggered on a git push event to the main/master branch of the GitHub Organisation's repository. Both of them run the same testing suite using a strategy matrix; the three strategies used are as follows:

- ubuntu-latest
- macos-latest
- windows-latest

Each strategy consists of a clean compilation of the main game package, followed by a clean run of the testing package. The strategies are named after the OS image used by the GitHub workflow runner, with '-latest' denoting that the image used is the latest image for that operating system in use by GitHub.

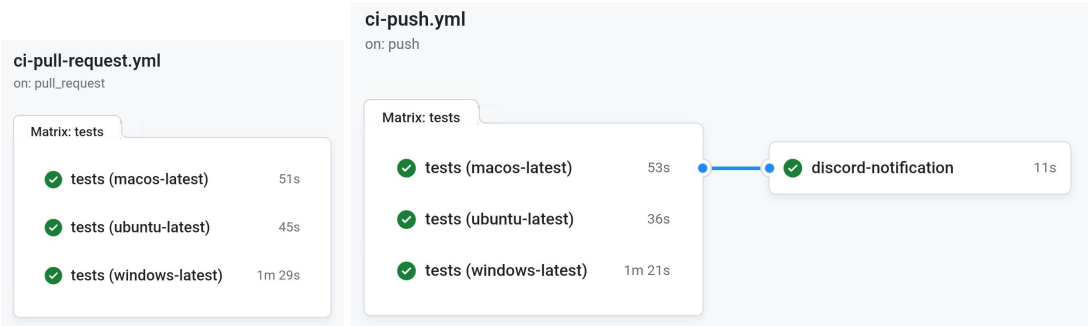
Despite Java programs being compiled to bytecode to be executed later by a JVM, meaning the behaviour of the program should be equivalent across all supported operating systems, we have decided to run the CI suite across all of them to identify OS-specific issues (e.g. hardcoding a filepath which cannot be understood by one of the OSes).

We have set up a branch protection rule for the main/master branch of the GitHub repository which requires checks to pass on a Pull Request targeting it, where the workflow triggered on pushed to PRs is automatically treated as part of the check suite - this ensures that our approach to CI is actually continuous.

The only other difference between the two workflows is that the workflow triggered by a push event has an additional step triggered only if the rest of the workflow succeeds: a humorous message is sent to a channel in our group's Discord guild (colloquially known as a 'server').

Aside from the workflows, our repository is set up to send webhooks to a separate channel on the aforementioned Discord guild, allowing our team members to respond faster to new events, such as git commit pushes, the opening of Pull Requests, the merging of new commits into the main/master branch, et cetera.

Pictured below are screenshots of recent examples of successful runs of each workflow.



Breakdown (3.4.4.5.b)

```
name: Java Push CI (Gradle, Temurin, JDK 11)

on:
  push:
    branches: master

jobs:
  tests:
    strategy:
      matrix:
                os: [ ubuntu-latest, windows-latest, macos-latest ]
    runs-on: ${{ matrix.os }}
    steps:
      - uses: actions/checkout@v3
      - name: Set up JDK 11
        uses: actions/setup-java@v3
        with:
          java-version: '11'
          distribution: 'temurin'
          architecture: 'x64'
      - name: Validate Gradle wrapper
                                    uses: gradle/wrapper-validation-action@e6e38bacfdf1a337459f332974bb2327a31aaf4b
      - name: Build with Gradle
                                    uses: gradle/gradle-build-action@67421db6bd0bf253fb4bd25b31ebb98943c375e1
        with:
          arguments: build
      - name: Run unit tests
           run: ./gradlew :tests:test --tests "com.team3gdx.tests.*"
  discord-notification:
    runs-on: ubuntu-latest
    needs: [tests]
    steps:
      - name: This, I like!
        env:
                  DISCORD_WEBHOOK: ${{ secrets.DISCORD_WEBHOOK }}
                                    uses: "Ilshidur/action-discord@0.3.2"
        with:
             args: 'Jackpot! All unit tests across all platforms are passing - this, I like!'
```

Above: Fig. 1

This workflow is triggered when a push event happens on the master branch, either from a direct push or from merging a PR.

The strategy matrix is defined to run on the latest images for Ubuntu, Windows, and MacOS.

The first step is to checkout the code to the runner locally.

The second step is to set up the local Java system. Here, we use the same version, distribution, and architecture as during development.

The third step is to validate that the Gradle Wrapper is set up correctly.

The fourth step is to run a clean build to detect compilation errors.

The fifth step is to run the test suite.

The sixth step, which does not exist on the PR-triggered workflow, is to send a humorous but appropriate message to the group's Discord guild.