Continuous Integration Report
Eng1 Group 29

Adam Hewlett
Ani Thomas
Dan Kirkpatrick
Dominik Hagowski
Matthew Crompton
Niko Chen

Summary of our Continuous Integration (CI) Methods and Approaches

CI1 Maintain a Single Source Repository
This was one of the approaches our team used. This is appropriate for the project as we want to ensure that each team member knows where our work is located. By sticking to the main branch we can ensure that each team member will be working on a recent branch.

CI2 Each Commit Should Build the Mainline on an Integration Machine
With this approach our team will be able to check that the implementation works, this is done through a CI server located on GitHub Actions. This will ensure that our project doesn't just work on one system. We ran into this problem in Assessment 1 where the game wasn't scaling correctly, this is a prime example of why we need this approach.

CI3 Fix Broken Builds Immediately
This should be done in our project, so that everyone who is working on the mainline has working code to work on top. The mainline is our team's shared reference so without fixing it could cause problems for other team members.

CI4 Keep the Build Fast
By using this method we can ensure that our team receives rapid feedback (a key feature of CI). For example we could set up parallelisation to run builds of different distributions simultaneously. With rapid feedback we can quickly get back to working on the development, thus, saving development time.

CI5 Commits to the Mainline Daily
This is an appropriate approach we can use as it will be easier to integrate into the mainline, and will allow other team members to stay up to date with an individual's work. Although we may not be working on our project daily we can definitely commit each day that our project is worked on.

CI6 Make the Build Self-Testing
The program may run and build successfully, but it needs to work as intended. With automated testing setup (which runs as part of the build process) we can catch unexpected behaviours before buggy builds are released.

CI7 Automate the Build
Builds involve a lot of different steps and thus may require long commands, to compile, run tests, package code, etc. With a single command that can do everything it will both save time during the build process and reduce confusion. When our code changes we can automatically use the same command in a CI tool.

Methods and Approaches we Considered

1. Making it Easy to Get the Latest Executable
   We decided to pass on this approach because we may not have many releases, and as our project is small there probably won't be many people wanting to access a new release during the main development period.
2. Testing in a Clone of the Production Environment
   Whilst we aren't creating the game for mobile environments, we are making it for various desktop operating systems. It may take a long time to run the tests on all the OS's and the development period is quite short. To ensure that all of the requirements are met, we are going to hold back on this method until we know that we have time to complete it.

Actual Continuous Integration Infrastructure Used
CI1 - The aim of this goal was to maintain a single source repository. This was achieved through the use of a GitHub repository. We can ensure that all our work is synchronised, and ensure that no one is working on an outdated version.
CI2 - The aim of this goal was to make sure each commit is building on the mainline, on an integration machine. As stated in the goal we would be doing this through GitHub Actions, we successfully created the CI server, but ran into many issues such as the jar not running but it was still being generated. We then would download the jar artefact and run it on various OS's and devices to ensure consistency.
CI3 - The aim of this goal was to fix broken builds immediately. We were relatively successful in this, whenever the build failed within GitHub Actions we would look into why it failed, fix this problem, and then commit again allowing it to run through the integration machine.
CI4 - The aim of this goal was to keep the build fast. We didn't run into any problems with this, our build isn't very complex, and our team is relatively small, this means that most commits were done relatively quickly. Our builds usually took around 1 minute, which we believed wasn't significant enough to begin setting up parallelisation.
CI5 - The aim of this was to commit to the mainline daily, I do believe that if we were working on the code daily this would have taken place. However, we weren't working on the code everyday, so we didn't need to commit to the mainline daily.
CI6 - The aim of this was to make the build generated self testing. We have not achieved this goal as we are yet to set up testing in GitHub Actions. However once this is added to GitHub Actions we will be able to see bugs before a build is actually released. We have however, created a check style sheet, this allows us to ensure that our code formatting is consistent throughout.
CI7 - The aim of this goal was to automate the build. We have been able to partially automate our build. On every commit we have a jar generated (that works), a check style sheet that highlights the errors in code formatting. The key thing we are missing here is automated testing.