UNIVERSITY OF YORK

DEPARTMENT OF COMPUTER SCIENCE

ENG1 Group Assessment 2
Team 1

Auber

CI2.pdf - Continuous Integration

Group Members:

Jonathan Davies

Jamie Hewison

Harry Smith

Zee Thompson

Mark Varnaliy

Continuous Integration Methods and Approaches:

We made sure to use CI when working on this project to ensure that everyone is always up to date with the current code base, helping to eliminate redundancy and promote collaboration. We also had a short deadline for this project, making the use of Continuous Integration vital to creating a cohesive product in a timely manner. To make sure that we were all working on the same code as much as possible each member pulled from the repository before working, and pushed when done. As most of the development was done asynchronously this made sure that each time a member was going to add to the repository, they had the most up-to-date version of the code. We used a mix of manual and automated testing for our game, with the automated testing being implemented using circleci and JUnit tests. This was incredibly useful in helping us to see when pushes to the repository had broken the code and needed to be rolled back/changed. The ease of automated testing of the game greatly assisted our agile workflow as we did not have to manually test the game after each iteration, only the new features.

Continuous Integration Infrastructure:

We chose to use Git as our repository, as it is easy to use and prevalent within industry. We all used IntelliJ as our development environment, which greatly simplified this as it has built in git integration. This allowed us to pull, commit and push all from inside our IDE. We then integrated our github accounts with circleci to provide the basis for our Continuous Integration infrastructure. We also considered using github actions to implement our CI workflow but decided circleci seemed a better fit as it focused specifically on helping bring CI into a project, whereas github actions is more general and helps implement many different workflows. Using circleci we were able to easily keep track of the workflow of the project, and our own individual as well as the collective pipeline's. We used some automated testing to check the features of the game that could be tested with JUnit were working, done by circleci for each new iteration. However for many features, such as UI, there is no way to automate testing so it was done manually. To keep track of manual testing requirements we used a shared google doc. We were able to automatically build the jar file with circleci on each iteration of the game, ensuring that we had a complete and deliverable product on each update to the code. This was done by re-writing the circleci config file to build an artifact using gradle.