Team 21

Zain Alshaikh

Corin Bertrand

Damien Heaton

Mandy Li

Brandon Oliver

Tom Tafijs

Part a:

For our continuous integration we used GitHub as our Version control system. This allowed us to use the GitHub Actions tab to view the results of our build and get the necessary feedback. We used this VCS as it allowed easy viewing of all the builds and details concerning them. In addition, it updates frequently as updates occur after every pull or merge request. This was important for our project as it ensured that every new item being pushed into the project is being tested. Furthermore, the group members are able to view what changes were made and whether or not they cause an error. In addition, this keeps them frequently updated with the most recent changes to the project.

We used the java CI with gradle as our build tool as it was easy to implement using since it was already provided by GitHub and is fast and reliable at building.

The continuous integration methods we used included unit testing. We did this by using the java CI with gradle and running automated tests after every commit or pull request. This allows us to make sure every new part of the code being implemented works as it should. The automatic testing also makes sure that we don't forget to test a certain feature and don't have to manually test every part. Then, a test report is then generated showing every successful build, every function it tests and the branch being tested.

All of this happened in two workflows, the java CI with gradle and the pages build deployment. Having only two workflows suited this project as the group didn't think adding any new ones was necessary. Furthermore, it allowed for quicker builds and faster testing.

Part b:

For the game development the infrastructure for the continuous integration is split into different branches that are updated every time a team member edits something in that branch. Once the build/test report is generated each one of the actions also has a build log which can be viewed in order to see the steps taken to build the action and the amount of time each step took.

The process of building can be shown in code written in Java gradle. This code is shown under the main tab of GitHub in the gradle.yml file (shown on the right hand side).

```yaml
name: Java CI with Gradle

on:
  push:
    branches: [ main ]
  pull_request:
    branches: [ main ]

jobs:
  build:

    runs-on: ubuntu-latest

    steps:
    - uses: actions/checkout@v2
    - name: Set up JDK 11
      uses: actions/setup-java@v2
      with:
        java-version: '11'
        distribution: 'temurin'
    - name: Build with Gradle
      uses: gradle/gradle-build-action@937999e9cc2425
      with:
        arguments: build
    - name: Archive production artifacts
      uses: actions/upload-artifact@v3
      with:
        name: dist
        path: desktop/build/libs/desktop-1.0.jar
    - name: Archive testing reports
      uses: actions/upload-artifact@v3
      with:
        name: testing-reports
        path: core/build/reports/
```

As for the website, GitHub pages were used to help build it. And every commit made to the GitHub pages branch changed the website as continuous integration ensured every commit to the website was added bug-free and updated the website once refreshed.