3D Model Binary Vision System

Testing Policy

Flap_Jacks

Rani Arraf
Quinn du Piesanie
Jacobus Janse van Rensburg
Steven Visser
Marcus Werren

Contents

1 Testing tools

1.1 Travis CI

We are using Travis CI as our continuous integration testing tool. Travis was the option we chose because it is easy to use and has a fast system for setting up. As we are using GitHub, Travis CI integrates really well with GitHub and allows for an automated testing system to be called when a commit is pushed to any branch in our repository.

1.2 Mocha

We chose Mocha as our unit and integration testing framework. Mocha is a JavaScript testing framework for NodeJS programs, featuring browser support, asynchronous testing, test coverage reports, and use of any assertion library (from which we used Chai). As our system runs off of a Restful NodeJS API mocha provides all the needed features to test our system thoroughly.

Mocha provides all the necessary packages to check code coverage, as well as perform performance tests.

1.3 Bechtest

Integrated performance testing for Mocha based unit testing. No special tests are needed for this package, as it uses the unit and integration tests already created. This allows for fast and easy performance checks on the system.

1.4 Sonar Cloud

SonarCloud is an online service to catch Bugs and Security Vulnerabilities in your Pull Requests that can be used to check code quality. This service links to Travis CI and can be easily used with out repository.

2 Testing paradigms

2.1 Unit Testing

For the unit testing, we used the mocha framework. When a commit is sent to the Git repository, it triggers Travis CI, which runs the tests. The developer who committed to the repository can then see all the tests that were run in Travis. Coveralls are linked to the Git repository to check the status of the latest build as well as code coverage and other performance checks.

2.2 Integration Testing

For this, we used the same or similar frameworks that we used for unit testing.

2.3 Non-Functional Requirement testing

2.3.1 Maintainability

We are using Sonar Cloud to test our code quality. With better code quality means that we have a more maintainable system.

2.3.2 Usability

We have created a questionnaire that asks random users how usable our system is, with a rating score of easy, moderate and difficult on each different feature. With the results we have so far, we have updated the features that have a poor usability, however, the questionnaire is still going and we will keep updating features that are found to have a poor usability.

2.3.3 Reliability

We use a NPM package called Bechtest, that will tests the speed and performance of our system.

3 Git Information

The link to the tests in our repo:

`https://github.com/COS301-SE-2020/3D-model-Binary-Vision/tree/master/Application/test`