SE 3XA3: Test Report
GrateBox

Team 8, Grate
Kelvin Lin (linkk4)
Eric Chaput (chaputem)
Jin Liu (liu456)

December 7, 2016

Contents

List of Tables

List of Figures

This document will make frequent reference to the Development, Test Plan, SRS and Design documents of the GrateBox project. They can be found here, here, here, and here respectively.

1 Functional Requirements Evaluation

Functional requirements were evaluated through automated and non-automated testing.

1.1 Automated Testing

Automated testing for GrateBox was conducted using the QUnit software outlined in the Development document. The test cases executed can be found in the test.js file here. Each test case in the test.js file corresponds with an automated test case in the Test Plan document. The automated test cases in the Test Plan document are those found in sections 3.1.1 and 3.1.2. All automated test cases were properly implemented and executed, and the results returned were the results expected, indicating successful implementation of functional requirements by automated testing.

1.2 Non-Automated Testing

Non-Automated testing for GrateBox's functional requirements was conducted by the test team outlined in the Test Plan document. These include the tests outlined in sections 3.1.3, 3.1.4, and 3.1.5 of the Test Plan document. All test cases were exectued correctly and returned the results expected. They are broken down into more detail below.

Table 1: **Revision History**

Date	**Version**	**Notes**
Dec 6	1.0	Document creation
Dec 7	1.1	Document completion

1.2.1 Graphics

Test for graphics are as follows.

GR-1.1

Cars are generated as expected given numerical values. The position of vertices, the connections between vertices, the radius of wheels, and the placement of wheels all vary depending on given input. Comparison with BoxCar-2D also verifies successful implementation of graphics module. Test successful.

GR-1.2

Cars are not generated as expected given invalid numerical values. An error message is displayed when this occurs. Test successful.

GR-1.3

Cars are not generated as expected given invalid numerical values. An error message is displayed when this occurs. Test successful.

GR-2.1

Road created corresponds to algorithm input. Test successful.

GR-2.2

No road created and error message is displayed when this occurs. Test successful.

1.2.2 Fitness and Score

Test for fitness and scores are as follows.

FI-1

Values calculated correspond to values observed. Test successful.

FI-2

Values calculated correspond to values observed and value properly displayed in GUI. Test successful.

1.2.3 Other GUI elements

Test for other GUI elements are as follows.

GU-1

Health bars operate properly. Test successful.

GU-2

Text file is created and is accurate. Test succesful.

2 Nonfunctional Requirements Evaluation

2.1 Nonfunctional Tests

The exact details of nonfunctional requirements can be found in the Test Plan document in section 3.2.

2.1.1 Look and Feel

LF-1

Majority of users agreed that the visual aesthetic of the program rated favourable. Test successful.

LF-2

Majority of users agreed that the style of the program rated favourable. Test successful.

2.1.2 Usability

US-1

Users performed all tasks in alloted time. Test successful.

US-2

Users performed all tasks in alloted time. Test successful.

US-3

Majority of users agreed that the program's usability rated favourable. Test successful.

2.1.3 Performance

PF-1

Time restriction for tasks performed met. Test successful.

PF-2

Majority of useres agreed that the program's usability rated favourable. Test successful.

PF-3

Numerical values and equations determined to be accurate and valid. Test successful.

PF-4

Majority of users agreed that the program's usability rated favourable. Test successful.

2.2 Evaluation based on tests and user survey

3 Comparison to Existing Implementation

The original implementation contained no testing of its own. One test conducted by Grate (GR-1.1) required a comparison to the original implementation. This comparison helped validate the results of the test as seen in section 1.2.1.

4 Unit Testing

All unit testing for this project can be found in the test folder found here. The tests.java file contains the source code for all testing. The Test.html file contains the output of this code. All test were conducted using the third party testing software QUnit, outlined in the Design document.

5 Changes Due to Testing

6 Automated Testing

7 Trace to Requirements

8 Trace to Modules

9 Code Coverage Metrics