1 TEST PLAN IDENTIFIER

Team PIK Third Development Cycle (Second deployable from Team DUS, first by Team FVS)

2 REFERENCES

(Files located in appendix folder)

- FVS Requirements Specification
- DUS Testing
- IEEE Test Plan Template

3 INTRODUCTION

This is a master plan to cover the Third cycle of development (Assessment 4). The main aim in this testing cycle is to test the new requirements. And of course conduct regression testing on all the previous features and requirements (assess then run, update and extend the original tests for the games features).Then, where necessary, write and extend the testing to cover all non required features added by Team PIK.

Testing will be conducted alongside development but also re-ran on completion to ensure a stable, deliverable software free from system critical bugs and issues.

4 SOFTWARE RISK ISSUES

As before we assume our third party libraries and frameworks (most notably libGDX) are free from bugs. Although this is probably not the case, they are so widely used and supported that they are unlikely to include system critical flaws and we are unlikely to encounter them. Also any prospect of resolving these issues is likely to lie outside the scope of our knowledge base and this project.

We are using Eclipse which has been used previously on this project, is well trusted and well maintained so we are unlikely to face issues there.

5 FEATURES TO BE TESTED

Critical features to test during this assessment are:
1. Reply Function
2. Add Track
3. Remove Track

Also for regression testing the features tested in the previous stage should be retested:
4. Graphical user interface
5. Game map (Cities and junctions)
6. Resource management
7. Routing system
8. Absolute goals system

Finally the optional added features:
9. GUI changes
10. Start screen
11. Instructions screen
12. Variable Turn Limits
13. Background music

6 FEATURES NOT TO BE TESTED

All implemented features in the game should be play tested to some degree.
Features relating to the requirements will be covered most thoroughly however.
Standard features implemented through libgdx can be assumed to be functional in the backend such as label drawing, buttons graphics etc.

7 APPROACH (STRATEGY)

As in the previous section two forms of testing will be used; System Testing and Unit Testing. Followed by extensive Regression Testing of the features from previous cycles.

Regression testing will include both forms, all tests used in the previous phase will be adapted and re-run. New System tests will be written however to cover the new features in this development cycle (listed in section 5). The Unit tests will have to be updated as code is adapted and additions can be made to cover the critical features. And new unit tests for Reply and Track modification (the new customer requirements) will be added.

8 SUSPENSION CRITERIA AND RESUMPTION REQUIREMENTS

If a test detects a defect with large scope and critical implication with no simple fix, this test should be halted and development on that feature resumed. Once the defect has been resolved testing should be resumed with extra diligence on the previously defective code.

If the bug can be resolved easily and has low scope, a patch can be quickly added and noted, and then testing may proceed as normal. Also if the defect is not easily resolved but not completely critical to stability or functional requirements, it should be raised and documented but given low priority. If the schedule is kept all bugs should be resolved or removed.

9 TEST DELIVERABLES

- Test Plan
- Test Implementation

10 REMAINING TEST TASKS

As this is the final development cycle all testing should be covered. If functions are omitted or the testing strategy changed the reasons for this should be discussed and explained in the testing evaluation.

11 ENVIRONMENTAL NEEDS

A critical requirement of the game is that it must run on the University Computer Science Undergraduate software laboratories computers. These are capable PC's but this should at least be informally tested due to potential resolution differences or other unforeseeable issues.

12 STAFFING AND TRAINING NEEDS

All members of the team involved in testing need general training in the supporting libraries (mainly libGDX). Even if not as hands on with the code, general understanding of the architecture and intended functionality of the game is vital to carrying out meaningful tests. Finally every member of the team should have basic knowledge of testing procedures but is least critical as they can refer to the SEPR presentation on Testing for this information.

13 RESPONSIBILITIES

Setting risks, and selecting the features to test was brainstormed by the entire team and frequently reviewed in SCRUM meetings so key items are not missed and to ensure a thorough report.

Also one member has testing as their main responsibility, communication with development oriented members of the team is critical for them to select the highest risk areas and thus select features that need testing most thoroughly. Equally during development dynamic play testing should be carried out by all members. Also the member assigned testing will need to collect information together and document the unit tests.

14 SCHEDULE

Following a test driven development method tests should be created and updated as the features are added to the game.

We plan to finish all development on code a week before the deadline (22/04/2015) so the next three days can be devoted to running and rerunning tests, and the remaining days devoted to bug fixing and evaluation.

Previous unit tests that are part of previous cycles should be checked at every commit and kept up to date.

15 PLANNING RISKS AND CONTINGENCIES

The biggest risk to testing was identified by our experiences in the last assessments as features coded near to the deadline were not tested as thoroughly due to lack of time. To mitigate this this time around we have set an internal deadline for coding a week before the deadline, giving time to expose and rectify bugs.

Another risk is different testing strategies between cycles may cause features to be overlooked to avoid this we have studied previous testing documentation in detail and tried to match our strategy as closely as possible.

16 APPROVALS

Our testing and testing documentation should demonstrate the stability and reliability of the deliverable to fellow students with similar technical knowledge and the customer who in this case also has considerable technical expertise.

17 GLOSSARY

Regression Testing - Testing previous functionality of code after changes or enhancements have been made.

Unit Testing - Software testing for individual blocks of code

System Testing- Testing the system as a whole using test cases (lists of actions)

JUnit - is a unit testing framework for the Java programming language. Accredited for its contribution to test driven software development as it provides high capability for flexible repeatable testing.