Appendix 3

Test plan document

Test Plan Template - Pixel Wizard

Prepared by - Grace Keane

Student ID - G00359990

05/05/2020

Table of Contents

[1.0 INTRODUCTION 3](#_Toc40188482)

[2.1 Objectives 4](#_Toc40188483)

[2.2 Tasks 4](#_Toc40188484)

[3.1 General 5](#_Toc40188485)

[3.2 Tactics 6](#_Toc40188486)

[4.1 Unit testing 6](#_Toc40188487)

[4.2 System & integration testing 7](#_Toc40188488)

[4.3 Performance and stress testing 8](#_Toc40188489)

[4.4 Batch testing 9](#_Toc40188490)

[4.5 Automated regression testing 9](#_Toc40188491)

[PRE-SUBMISSION: 9](#_Toc40188492)

# 1.0 INTRODUCTION

**This test plan describes the testing approach and overall framework that will drive the testing of the “Pixel Wizard” 2D application.**

The customer wants a perfect game, which passes the full cycle of manual testing. Given the specificity of the application it is very important to have the same level of quality as portrayed in the appendix 2 documentation.

The test plan has been created to facilitate communication within the team members. It describes approaches and methodology that will apply to the unit, integration and system testing of the game design document. It includes the objectives, test responsibilities, entry and exit criteria, scope, schedule major milestones and approach. This document has clearly identified what the test deliverables will be and what is deemed in and out of scope.

*Brief overview of the application itself*

This goal is focused on creating a test plan document for a 2-Dimensional horizontal scrolling shooter game called Pixel Wizard. The main theme is set in a forest. The background is used repeatably to make the level seem longer. On the top left and right of game play there is health representations for both player and enemy. These are displayed by red and blue diamonds. Players have an opportunity to increase their health using a health pickup which is located on a platform in game play. There are three characters to this game, they are player, enemy and boss. These characters are programmed to attack. The main theme of this game is "kill or be killed". If the player is successful, they will progress to other levels and soon win the game overall.  
  
The game has various requirements which will allow this application to be of high quality and fit for purchase. The game has a start-up main menu which contains buttons for settings, load game, delete game, exit game and a play game button which takes the player from the title screen to the point that gameplay begins.  
  
An in-game pause menu is also included. Once the game has been paused using the space bar, the player can access settings or exit the game, a save game option will also be presented, allowing the player to save the game state.  
  
The way in which the player controls the game entities is relatively simple which will cause no confusion to the player. Control is different for both laptop and mobile phone.

2.0 OBJECTIVES AND TASKS

### 2.1 Objectives

1. Ensuring that the software under test is bug free before release.
2. Gaining confidence in and providing information about the level of quality.
3. To make sure that the result meets the business and user requirements.
4. To ensure that the application satisfies the client.
5. To gain the confidence of the customers by providing them a quality product.
6. Find as many software defects as possible.
7. Result should be a production-ready software.

**Document to be used**  
Appendix 2 – Game Design Document

### 2.2 Tasks

* Ideally one test must be carried out at a time to avoid tracking errors or bugs arising.
* Small, clear and non-complex test cases should be written.
* Exit criteria and test closure must be fully thought out.
* Testing must be carried out at an early stage to leave time for possible critical errors.
* Different independent test cases must be carried out to avoid pesticide paradox.
* Work must be evenly divided amongst testers.
* If problems arise, they must be recorded and solved quickly.
* Main components of the game must be tested individually and then tested again when combined with the rest of the application.
* Post testing must be fully done when application is complete.
* All in scope material must be fully tested by the test team.
* Regression testing must be carried out with every addition of code.

3.0 SCOPE

### 3.1 General

* Exploratory test opening main menu functionality and check if all buttons work as expected.
* Play game - should take the player into the game and the player should begin at Level 1.
* Settings - should navigate to another page which will allow the player to edit game settings. Game settings like sound and music volume will be tested also.
* Load game - this should allow the player to choose which level to load. Test can the user select a game and note does to game navigate to the correct one selected.
* Delete game - should allow players to delete a games history.
* Exit game – should quit the application.
* Exploratory test the in-game menu to check if it is working correctly with no bugs.
  + Once in game and user clicks the required key (Laptop – spacebar, Mobile – button on top right of screen). Program should navigate to pause screen.
  + Save game – should save position and status of current game being played.
  + Settings – should navigate to the settings page. Also allows for change of sound and music volume.
  + Exit game – allows users to exit the game being played.
  + Resume game –The player can resume the game by selecting the appropriate option or simply pressing the assigned button for pausing/ resuming the game. When key clicked it should enable the game to be played again at current position and status.
  + Restart level – should reset the entire level.
* Fully test in-game functionality and design using functionality testing, compatibility testing, regression testing.
  + Test player navigation with correct key specified in document.
  + Test crouch defence mechanism.
  + Test jump mechanism.
  + Test if player can shoot using key specified.
  + Note if enemy can shoot at player.
  + If player gets hit by an enemy’s bullet check if health score decreases.
  + If enemy gets hit by the players bullet check if health score decreases.
  + hit Note can a health increase be accessed by player touching the “+” symbol.
  + Test whether player can get hit and eventually die.
  + Test whether enemy can get hit and eventually die.
  + Test whether player can move onto next level when level one has been completed successfully.

### 3.2 Tactics

* Exploratory testing would be carried out once the build is ready for testing.
* All defects seen should come along with a snapshot JPEG format and they should be fully recorded.
* The test team assumes all necessary inputs required during test design and execution will be supported by development/ business analysts appropriately.
* Test case design activities will be performed by the QA group.
* Any defect fixes planned will be shared with test team prior to applying fixes to application.
* Project manager will review and sign-off on test deliverables.
* Project team has the required experience and knowledge.
* Test team will manage test planning, test design and test execution support.
* Be familiar with game rules and test the gameplay against these rules.

4.0 TESTING STRATEGY

### 4.1 Unit testing

**Description**

Unit testing is a level of software testing where individual units/ components of a software is tested. The purpose is to validate how each unit of the software performs as designed. A unit is the smallest testable part of any software material. It usually has one or a few inputs but usually only one output.

**Participants**

Grace Keane

Jane Murphy

**Popular unit testing framework**

One of the most popular C# unit testing frameworks is NUnit. It is an open-source unit testing framework for the .NET framework and Mono. It serves the exact same purpose as Junit does in the java world.

*NUnit features*

* Tests can be run from a console runner, within a visual studio app through a test adapter, or 3rd party runners.
* Tests can be run in parallel.
* NUnit has a strong support for data driven tests.
* It supports multiple platforms including .NET Core, Xamarin Mobile, Silverlight or Compact framework.
* Test cases can be added to one of more categories, this allows selective running.

**Methodology**

How unit test will be carried out

Typically, unit testing is utilized to ensure that the underlying logic does not change with refactoring and other changes such as new featured. Throughout development of this Pixel Wizard application all added functionality must be tested in small units to avoid bugs from appearing. Testing can be done manually, automated or by using NUnit. Unit testing is typically carried out unity 100% decision coverage is met.

Who will write test scripts?

Script writer – Grace Keane

Tester – Jane Murphy

How will testing activity take place

1. Scripts must be fully written.
2. Manual tests must be carried out on singular small units or functions defined in scope.
3. NUnit can be used to test score outcomes using the console.
4. After every change to the application, regression tests must test updated data.
5. Software must be tested in small unity before being system tested.

### 4.2 System & integration testing

**System testing**

System testing is testing conducted on a complete integrated system to evaluate the system's compliance with its specified requirements. System testing is one of the last tests to be carried out on a software project/ application.

**Integration testing**

Integration testing tests interfaces between components and interactions to different parts of a systems such as an operating system, file system and hardware or interfaces between systems. It occurs after unit testing and before validation testing.

**Participants:** Jane Murphy, Olivia Simpson

System testing - Jane Murphy

Integration testing - Olivia Simpson

Test scripts implementation & design – Grace Keane

**Methodology**

Integration testing should be carried out after unit testing. Once all the individual units are created and tested, then start combining those tested modules and start performing the integrated testing. The main goal here is to test the interfaces between the units/modules.

For integration testing the team must use Bottom-up testing. This is an approach to integrated testing where the lowest level components are tested first, then used to facilitate the testing of higher-level components. The process is repeated until the component at the top of the hierarchy is tested.

*How can Integration testing be carried out?*

1. Prepare the test integration plan
2. Decide on the type of integration testing approach
3. Design test cases, test scenarios and test scripts accordingly
4. Deploy the chosen modules together and get the integration tests running
5. Track the defects and record the test results of tests
6. Repeat the above steps until the complete system is tested

System testing of software is to be testing conducted on a complete and integrated system to evaluate the system's compliance with its specified requirements. It is the very last test to be carried out on this “Pixel wizard application”. System testing includes functional & non- functional testing and is performed by the testers. System Testing is done after Integration Testing. This plays an important role in delivering a high-quality product.

### 4.3 Performance and stress testing

**Stress testing**

Stress testing is the software testing approach for exercising a software system beyond its maximum design load. Stress testing causes defects to come to light faster.

**Participants**

Stress testing - Grace Keane

Test script implementation - Olivia Simpson

**Methodology**

For this Pixel Wizard application, performance testing can be used to test whether the game can be played at a certain speed, page refreshes are relatively fast and shoot speeds are as expected throughout the game.

*Steps when stress testing*

Stress Testing process can be done in 5 major steps:

1. Planning the Stress Test. Here you gather the system data, analyse the system, define the stress test goals.
2. Create Automation Scripts: In this phase, the Stress testing automation scripts should be designed and implemented as well as generating the test data for the stress scenarios.
3. Script Execution: In this stage, the Stress testing automation scripts should be ran results should.
4. Results Analysis: In this stage, you analyse the stress test results and identify bottlenecks.
5. Tweaking and Optimization: In this stage, the system should be fine-tuned, configurations changed, optimize the code with goals that meet the desired benchmark.

Stress testing can be done on separate units of the application or on the completed finished software.

### 4.4 User acceptance testing

**Definition**

User acceptance testing focus on functionality validating fitness for purpose. It is Performed by users and application managers. User acceptance testing Links tightly to system test and may overlap in time E.g. performance - load time may be an issue, usability of product.

**Participants**

User acceptance testing & implementation of test scripts - Grace Keane

**Methodology**

UAT is done by the intended users of the system or software. This type of Software Testing usually happens in Beta Testing. Once Entry criteria for UAT are satisfied, the defined tasks need to be performed by the testers:

1. Analysis of Business Requirements
2. Creation of UAT test plan
3. Identify Test Scenarios
4. Create UAT Test Cases
5. Preparation of Test Data (Production like Data)
6. Run the Test cases
7. Record the Results
8. Confirm business objectives

User acceptance testing is the last test to be done on a software product. It is carried out when the full application is complete. User acceptance testing fully tests the main components of the software product.

### 4.5 Batch testing

A Group of tests executing sequentially one by one is called Batch Testing. All automated test scripts are executed one at a time by keeping the other scripts in waiting mode because the end state of one test is base normally the start state to another test.

### 4.6 Automated regression testing

**Definition**

Regression testing is the selective retesting of a system or component to verify that modifications have not caused unintended effects and that the system or component still works as specified in the requirements. Regression testing is useful for detecting bugs at the beginning stages of product development.

**Regression testing participants**

Grace Keane

Jane Murphy

Olivia Simpson

**Methodology**

regression tests must be incorporated in the daily test cycle. After every change to code or the system, regression testing should take place. The application should be manually ran by the game platform. This ensures that the addition of extra features has not corrupted the functionality of the original game.

### 4.7 Beta testing participants

**Participants**

John Walsh

Sam Kelly

**Definition**

A beta test is a type of testing that is conducted prior to any sort of official release. It is considered the last stage of software testing. Beta testing involves releasing the application to sites or individual users outside Game Development International Ltd.

**Methodology**

Beta testing will be conducted using cross-platform users. The users send records of incidents with the system to the development organization where defects will be analyzed and repaired. Developers will not be present when beta testing has begun therefore it is not held in a controlled environment.

## PRE-SUBMISSION:

The game is sent to the manufacturer (e.g. SONY, Microsoft) so they will briefly look at it and provide us with some feedback. Pre-submission gets us some idea about the readiness of the game. This is sort of an informal review, if we don’t receive any problem, it doesn’t necessarily mean the game is bug free. We often have a couple of pre-submissions for a brand new game (e.g. our first action game of Jackie Chan Stunt master), and we may skip the pre-submission for the second generation of a previous game title.

EXTRA

* Must examine the entire screen and not just small parts of it.
* Be familiar with game rules and test the gameplay against these rules.
* Test for clipping (two or more objects overlapping or cancelling each other out).
* Test for incorrect and inappropriate collision.
* Move the character through all the available objects and levels and closely examine the effect.