Prepared by:

Judita Kasiliauskaite  
G00348535

software test plan

“The Pixel Wizard”

Contents

[1.0 Introduction 2](#_Toc39589806)

[2.0 Objectives and Tasks 2](#_Toc39589807)

[2.1 Objectives 2](#_Toc39589808)

[2.2 Tasks 2](#_Toc39589809)

[3.0 SCOPE 2](#_Toc39589810)

[3.1 General 2](#_Toc39589811)

[3.2 Tactics 3](#_Toc39589812)

[4.0 Testing Strategy 3](#_Toc39589813)

[4.1 Unit Testing 3](#_Toc39589814)

[4.2 System Integration Testing 3](#_Toc39589815)

[4.3 Performance and Stress Testing 4](#_Toc39589816)

[4.4 User Acceptance Testing 4](#_Toc39589817)

[4.5 Automated Regression Testing 4](#_Toc39589818)

[4.6 Beta Testing Participants: 5](#_Toc39589819)

[5.0 Test Schedule 5](#_Toc39589820)

[6.0 Control Procedures 5](#_Toc39589821)

[6.1 Problem Reporting 5](#_Toc39589822)

[6.2 Change Requests 5](#_Toc39589823)

[7.0 Features to Be Tested 5](#_Toc39589824)

[8.0 Features Not to Be Tested 5](#_Toc39589825)

[9.0 Resources/Roles & Responsibilities 5](#_Toc39589826)

[10.0 Schedules 6](#_Toc39589827)

[11.0 Risks/Assumptions 6](#_Toc39589828)

[12.0 Tools 6](#_Toc39589829)

[13.0 References 6](#_Toc39589830)

# 1.0 Introduction

This test plan describes the testing approach and overall framework that will drive the testing of the 2-D game “The Pixel Wizard” which can be played on a PC and on a mobile phone. This document will assist the designated staff and personnel in testing in completing the tasks.

The product being tested is a 2D side-scrolling platformer game. The characters and the world will be created by using Asperite to create 2D sprites and animations. The game will have a wizard character (which is the player) navigating through levels which get more difficult each time. The character will use its magic to defeat the enemies in each level and each level will have a boss that must be defeated before progressing onto the next level. Each level will also have various pickups for the player such as health pickups to replenish the player’s health bar in the game.

This game will allow the player to control specific characters, it will have game statistics, relational attributes with other game objects like enemies, allow the player to navigate the levels easily and have obstacles that the player must overcome.

2.0 Objectives and Tasks

## 2.1 Objectives

The objective of the test is to verify the functionality and flow of ‘The Pixel Wizard’ and that the game works according to the specifications.

The test will execute and verify test scripts, identify and fix bugs, complete tests according to a plan and prioritise defects for future testing and fixing.

The final product of the test is to have a functioning and easy flowing 2-D side-scrolling platformer game which will run error-free.

## 2.2 Tasks

* Perform Unit Testing
* Perform System Integration Testing
* Perform Performance and Stress Testing
* Perform User Acceptance Testing
* Perform Automated Regression Testing
* Perform Beta Testing Participants

3.0 SCOPE

## 3.1 General

The purpose of this test plan is to achieve our goals that are described within the objectives and tasks. There will be a specific timeframe set to achieve that. This is to ensure that the best possible product is being presented and released to the customers. The testing will be done out as if the person is playing the game for the first time, to make sure that all glitches and errors are removed before releasing it. As well as that, it’s very important that the customer is able to easily navigate through the game and that the game is understandable in general. Therefore, all of the buttons and interfaces must be tested along with everything else.

## 3.2 Tactics

The tasks and objectives will be tested on a scheduled timeframe as mentioned above in the general part of the scope. It will be our priority to make sure that everything is tested efficiently and in a structured approach before releasing it to the public.

# 4.0 Testing Strategy

A testing strategy is vital for a test plan to be successful. Its aim is to provide the way we’re going to approach the test plan and to provide some sort of an outline. As well as that, it’s important for us to have a test strategy so that we follow the timeframe, keep up with the goals, and complete each step as efficiently as possible in order to remove any defects that could appear in the testing phase before releasing the product to the public.

4.1 Unit Testing

Definition:

Unit testing is when individual parts/components of a software are tested. A unit is the smallest testable part of any software. It may be an individual program, function or method, in this case it could be a function which allows the character to move. In unit testing it’s important to focus only on the tests that impact the behaviour of the system. Unit testing should have one or more inputs and one output.

Some benefits of unit testing:

* Code is more reusable
* Development is faster
* Debugging is easy

One example of unit testing is white box testing which allows a software to be tested beyond the user interface into the depth of a system.

Participants:

The developer and/or development team is responsible for unit testing. This is to allow a more efficient way of performing the test as the developer knows and understands the code better than any other department.

Methodology:

Unit testing can be done manually but automating the tests can speed up the process. The development team at this level will test things such as character movement within the game, the various controls for PCs and mobile phones as well as the overall functionality of the game. It’s important to test every stage of the game step by step to avoid any glitches or bugs appearing when the game is already released to the public. It will be tested based on the objectives and tasks as stated above.

4.2 System Integration Testing

Definition:

System integration testing (SIT) is a type of software testing which allows to test a system in order to verify the behaviour of the complete system. It also tests the interface between modules of the software application to make sure that they’re interacting between each other properly. In this scenario, it would be the testing of the overall system of the game, from start to finish.

Some benefits of SIT:

* Helps detect defects early
* Earlier feedback on acceptability of modules

Participants:

The development team will be responsible for system integration testing.

Methodology:

Describe how System & Integration testing will be conducted, including a description of tests to be

carried out Who will write the test scripts for the unit testing, what would be sequence of events of

System & Integration Testing, and how will the testing activity take place?

## 4.3 Performance and Stress Testing

Definition:

List what is your understanding of Stress Testing for your project.

Participants:

Who will be conducting Stress Testing on your project? List the individuals that will be responsible

for this activity.

Methodology:

Describe how Performance & Stress testing will be conducted, including a description of tests to be

carried out Who will write the test scripts for the testing, what would be sequence of events of

Performance & Stress Testing, and how will the testing activity take place?

## 4.4 User Acceptance Testing

Definition:

The purpose of acceptance test is to confirm that the system is ready for operational use. During

acceptance test, end-users (customers) of the system compare the system to its initial requirements.

Participants:

Who will be responsible for User Acceptance Testing? List the individuals' names and responsibility.

Methodology:

Describe how the User Acceptance testing will be conducted, including a description of tests to be

carried out Who will write the test scripts for the testing, what would be sequence of events of User

Acceptance Testing, and how will the testing activity take place?

4.5 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.

Participants:

Methodology:

4.6 Beta Testing Participants:

Methodology:

5.0 Test Schedule

Include test milestones identified in the Software Project Schedule as well as all item transmittal

events.

Define any additional test milestones needed. Estimate the time required to do each testing task.

Specify the schedule for each testing task and test milestone. For each testing resource (that is,

facilities, tools, and staff), specify its periods of use.

6.0 Control Procedures

## 6.1 Problem Reporting

Document the procedures to follow when an incident is encountered during the testing process. If a

standard form is going to be used, attach a blank copy as an "Appendix" to the Test Plan. In the

event you are using an automated incident logging system, write those procedures in this section.

## 6.2 Change Requests

Document the process of modifications to the software. Identify who will sign off on the changes

and what would be the criteria for including the changes to the current product. If the changes will

affect existing programs, these modules need to be identified.

7.0 Features to Be Tested

Identify all software features and combinations of software features that will be tested.

# 8.0 Features Not to Be Tested

Identify all features and significant combinations of features which will not be tested and the

reasons.

9.0 Resources/Roles & Responsibilities

Specify the staff members who are involved in the test project and what their roles are going to be

(for example, Mary Brown (User) compile Test Cases for Acceptance Testing). Identify groups

responsible for managing, designing, preparing, executing, and resolving the test activities as well as

related issues. Also identify groups responsible for providing the test environment. These groups

may include developers, testers, operations staff, testing services, etc.

10.0 Schedules

Identify the deliverable documents. You can list the following documents:

- Test Plan

- Test Cases

- Test Incident Reports

- Test Summary Reports

11.0 Risks/Assumptions

Identify the high-risk assumptions of the test plan. Specify contingency plans for each (for example,

delay in delivery of test items might require increased night shift scheduling to meet the delivery

date).

12.0 Tools

List the Automation tools you are going to use. List also the Bug tracking tool here.

# 13.0 References

<http://softwaretestingfundamentals.com/unit-testing/>