Prepared by:

Judita Kasiliauskaite  
G00348535

software test plan

“The Pixel Wizard”

Contents

[1.0 Introduction 2](#_Toc39592576)

[2.0 Objectives and Tasks 2](#_Toc39592577)

[2.1 Objectives 2](#_Toc39592578)

[2.2 Tasks 2](#_Toc39592579)

[3.0 SCOPE 2](#_Toc39592580)

[3.1 General 2](#_Toc39592581)

[3.2 Tactics 3](#_Toc39592582)

[4.0 Testing Strategy 3](#_Toc39592583)

[4.1 Unit Testing 3](#_Toc39592584)

[4.2 System Integration Testing 3](#_Toc39592585)

[4.3 Performance and Stress Testing 4](#_Toc39592586)

[4.4 User Acceptance Testing 4](#_Toc39592587)

[4.5 Automated Regression Testing 4](#_Toc39592588)

[5.0 Test Schedule 5](#_Toc39592589)

[6.0 Control Procedures 5](#_Toc39592590)

[6.1 Problem Reporting 5](#_Toc39592591)

[6.2 Change Requests 5](#_Toc39592592)

[7.0 Features to Be Tested 5](#_Toc39592593)

[8.0 Features Not to Be Tested 5](#_Toc39592594)

[9.0 Resources/Roles & Responsibilities 5](#_Toc39592595)

[10.0 Schedules 6](#_Toc39592596)

[11.0 Risks/Assumptions 6](#_Toc39592597)

[12.0 Tools 6](#_Toc39592598)

[13.0 References 6](#_Toc39592599)

# 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 cases, 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.

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:

Mary McDonagh and John Healy.

Methodology:

Unit testing can be done manually but automating the tests can speed up the process. John will write up the test cases. Mary and John 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. Unit testing has to be done first before continuing onto any other testing phases.

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.

We are going to follow the big bang approach as this approach will be integrated only when all of the modules of the application are completely ready.

Participants:

James Lally.

Methodology:

SIT will help with the integration of the system as it allows defects to be detected early which allows to have these solved asap. James Lally will be in charge of writing the cases and the steps will be done in a step by step sequence for this testing phase. SIT can only be done once unit testing has been finished.

## 4.3 Performance and Stress Testing

Definition:

Performance and stress testing is performed to determine the speed of a device or network, in this case it will be performed for the speed of the game and the devices – PC and mobile phone. It’s also performed to check the stability and robustness of the system (game in this scenario).

Participants:

Naomi Flood and Daniel McHugh.

Methodology:

Naomi and Daniel will be in charge of conducting this testing at the highest level possible. Naomi will oversee the writing of the test cases while working on the actual testing alongside Daniel. They will make sure to overload the game and check how it reacts to that overload. They will also be responsible to test how long does it take for a specific control to execute once brought into action, e.g. if the pause button is pressed, how long will it take for the game to actually pause. Testing will be done according to its importance and it can only be done once SIT is finished.

## 4.4 User Acceptance Testing

Definition:

The purpose of acceptance test is to confirm that the system is ready for releasing to the public. In this case, beta-testers will test the game against the game requirements and to check if it is fit for its purpose.

Participants:

Team of 5 beta-testers that are familiar with the business/game requirements.

Sam Heany

Methodology:

Beta-testers will be brought on site and this way we will monitor their gaming experience. We will make sure that each beta-tester tests every aspect of the game from start to finish and we will have them complete a survey answering questions about response time, speed, functionality, etc. Sam Heany will create the survey and he will be in charge of collecting the answers from each beta tester. This testing can only be completed after all of the above tests are completed as this test is conducted to make sure that the game is ready to be released.

4.5 Automated Regression Testing

Definition:

Automated regression testing is performed to confirm that any changes of the code hasn’t affected the functionality of the system. In this case, the game test-cases will be re-run to check that none of the prior code changes affected any of the game features.

Participants:

Sean Howard

Methodology:

Sean Howard will have to re-run existing test cases on the game and make sure that none of the functionality or features have been affected in any way. Automated tests are far more cheaper to execute and it’s faster. This test will be executed as the last testing phase before the release of the game to the public.

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

* Check sound effects/background music (in settings from main menu)   
  On/off sound & background music  
  Verify if sound effects are in sync with actions  
  On/off device sound and check

Check sound & background music settings in pause menu

* User Interface  
  Check in landscape/portrait mode  
  Check animation/movement of character, graphics  
  Check the various control mechanisms  
  Check side-scrolling  
  Character should not move out of the specified area   
  Test whether one object overlaps another  
  Text displayed (font, size, etc)  
  Check new level message
* Performance  
  Check the loading time of the game  
  Game flow should be fast – check that various actions aren’t taking too long to respond
* Multitasking  
  Check what happens when switching between apps while playing game
* Health-bar  
  Check if health-bar reduces for character & enemy
* Pause  
  Check if game is paused and resumes from same spot when clicked to pause the game  
  Check pause menu
* Save settings (if implemented)  
  Check if settings have saved after device has been turned off and back on

Check save menu

* Restart   
  Check if level has reset itself – character/enemy/pickups
* Functionality  
  Check game area/logic  
  Play until last level to check for defects  
  Check for pick-up assets  
  Check if boss character appears  
  Menu options

# 8.0 Features Not to Be Tested

* Interruption  
  When game is running, check the behaviour of game when things like text messages/calls are interrupting game  
  Check what happens when phone dies while playing

9.0 Resources/Roles & Responsibilities

* Mary McDonagh – responsible for executing unit testing and checking functionality of the overall game.
* John Healy – responsible for writing the test cases in unit testing and also checking the character movement, controls within the game.
* James Lally – responsible for writing the test cases for system integration testing and performing the testing of the system.
* Naomi Flood – responsible for writing the test cases for performance and stress testing while also testing the functionality of the game when the game is overloaded.
* Daniel McHugh – responsible for the control execution response times when stress testing.
* Sam Heany – responsible for preparing the survey for beta-testers when conducting user acceptance tests.
* Sean Howard – responsible for re-running test cases one last time before releasing the game to the public during the automated regression testing phase.
* Lisa O’Nally – responsible for resolving/fixing any bugs or defects of higher importance/difficulty.
* Beta-testers (5) – responsible for testing the functionality of the game while also overloading it, and making sure it meets the requirements of the game at the same time.

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

|  |  |  |
| --- | --- | --- |
| **Risk** | **Probability** | **Mitigation Plan** |
| Schedule Backlog/ Task Delay | High | * Figure out the origin of the problem and solve immediately * Require testers to sign off task completion report lists |
| Data Corruption/Loss | High | * Files must be backed up to local storage every hour * Also, files must be backed up to online cloud storage every day |
| System/Hardware Crash | High | * Files are stored and backed up on numerous systems |
| Hardware Malfunction/Breakdown | High | * Keep spare parts for temporary use while hardware is being fixed * If there’s no spare parts, get it fixed and get the hardware running asap * Software installation files should be kept to avoid any more delays when re-installing hardware |
| Unexpected Absence of Staff | Medium | * Have temporary substitutes employees enlisted * All staff should document their weekly work in case substitute needs to fill in for them * If not, director must fill in their spot for the time being |
| Insufficient Testers | Low | * Project leaders should temporarily fill in the spots if needed * Hire 3rd party tester or freelancer to help out |

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/>

<https://www.softwaretestinghelp.com/system-integration-testing/>

<https://www.guru99.com/performance-vs-load-vs-stress-testing.html>

<https://www.softwaretestinghelp.com/what-is-user-acceptance-testing-uat/>

<http://softwaretestingfundamentals.com/test-plan/>

<https://www.guru99.com/regression-testing.html>

<http://templatelab.com/test-plan/>

<https://www.softwaretestinghelp.com/wp-content/qa/uploads/2014/02/Live_Project_Test_Plan_SoftwareTestingHelp.pdf?fbclid=IwAR2gL760W1R94RWNaGNQrQ893rL5D2W2CSFTcqObtsvsQG58fH5GygPvt_Q>