Test Plan Project

Software Testing

Product: The pixel Wizard

Test Plan ID: 130520

STUDENT: aOIFE kILKELLY- g00342834

dATE: 13/05/2020



# TABLE OF CONTENTS

# Introduction

# 2.0 Objectives and Tasks

# 2.1 Objectives

# 2.2 Tasks

# 3.0 Scope

# 4.0 Testing Strategy

# 4.1 Unit Testing

# 4.2 System and Integration Testing

# 4.3 Performance and Stress Testing

# 4.4 User Acceptance Testing

# 4.5 Batch Testing

# 4.6 Automated Regression Testing

# 4.7 Beta Testing

# 5.0 Test Schedule

# 6.0 Control Procedures

# 7.0 Features to Be Tested

# 8.0 Features Not to Be Tested

# 9.0 Resources/Roles & Responsibilities

# 10.0 Schedules

# 11.0 Risks/Assumptions

# 12.0 Tools

# Introduction

The product that is being tested is called “The Pixel Wizard”. This is a 2D side-scrolling platformer game. This game allows the player to control a specific character, that has an important fictional/narrative role. There are statistics and relational attributes with other game objects, enemies, and the player character. The player takes on and navigates through the levels using an easy-to-use user interface. There are also obstacles that the player must overcome, such as enemies and bosses. Each level contains pickups for the player, such as health pickups to replenish the player’s health.

# Objectives and Tasks

## Objectives

The Objective of this test plan is to define the various testing strategies and testing tools used for a complete testing of the life cycle of this project. The objectives are:

* Finding defects which may get created by the programmer while developing The Pixel Wizard.
* Gaining confidence in and providing information about the level of quality.
* To prevent future defects.
* To make sure that the final result meets and user and business requirements.

## 2.2 Tasks

* Determining the scope and the risks that need to be tested and that are NOT to be tested.
* Documenting Test Strategy and Methodologies
* Ensuring that the testing activities have been included.
* Evaluating the tests estimate.
* Planning when and how to test and deciding how the test results will be evaluated and defining test exit criterion.
* The Test artefacts delivered as part of test execution.
* Defining the management information, including the metrics required and defect resolution and risk issues.
* Ensuring that the test documentation generates repeatable test assets.

# Scope

1. GUI
2. Control Mechanisms
3. System interfaces
4. Testing environments

### Tactics

I will be testing the games GUI to check all the elements for size, position, width, length, and acceptance of characters or numbers. I will also ensure the control mechanisms in the game work with no errors. The system interface will be tested by performing a System and Integration Test along with a User Acceptance Test. I plan to then test The Pixel Wizard environments by conducting a unit test. Below is the testing strategy I will follow.

# 4.0 Testing Strategy

This project includes the following testing tasks:

1. Unit testing
2. System and Integration Testing
3. Performance and Stress Testing
4. User Acceptance Testing
5. Batch Testing
6. Automated Regression Testing
7. Beta Testing

### Participants (Team Members)

Mary Forde- Test Team Lead and project Leader

Amy Smith – Tester

Jack Davis- Lead Developer

Mike Whatts– Developer

Rebecca Black- Lead Quality Assurance (QA)

Ethan Fahy- Quality Assurance (QA)

Shaun Purcell- Front end Developer

## 4.1 Unit Testing

### Definition:

Is a level of software testing where individual units/ components of a software are tested. The purpose is to validate that each unit of the software performs as designed. A unit is the smallest testable part of any software.

### Participants:

Mary Forde and Amy Smith

### Methodology:

Unit Testing is the first [level of software testing](http://softwaretestingfundamentals.com/software-testing-levels/). It is performed by using the [White Box Testing](http://softwaretestingfundamentals.com/white-box-testing/) method. **White box testing** is based on an analysis of the internal structure of the. The Pixel Wizard game. This can start at an earlier stage. I do not need the GUI to be available.

## 4.2 System and Integration Testing

### Definition:

The purpose of this level of testing is to expose faults in the interaction between integrated units. Test drivers and test stubs will be used to assist in Integration Testing. Integration Testing is the second level of testing performed after Unit Testing and before System Testing.

### Participants:

Jack Davis and Rebecca Black

### Methodology:

Black Box Testing, White Box Testing and Gray Box Testing methods can be used. I will be using black box testing as it will help me solve:

* Incorrect or missing functions of the game
* Interface errors
* Errors in data structures or external database access
* Behavior or performance errors of The Pixel Wizard
* Initialization and termination errors

## 4.3 Performance and Stress Testing

### Definition:

**Performance Testing** is a type of software testing that intends to determine how a system performs in terms of responsiveness and stability under a certain load.

**Stress Testing** is a type of performance testing conducted to evaluate the behavior of a system at or beyond the limits of its anticipated workload.

### Participants:

Mike Whatts and Shaun Purcell

### Methodology:

* **Always Increasing Load**: load is increasing during the whole test until the servers being tested fall apart
* **Stepping load**: it is good practice to increase load with steps so that I can maintain a stable level of load for some time before going to the next one. This will make analysis of server behaviour much easier for each step
* **Various Durations**: the test duration only depends on the rate of load increase you want to apply on the game system

## 4.4 User Acceptance Testing

### Definition:

Formal testing with respect to user needs, requirements, and business processes conducted to determine whether a system satisfies the acceptance criteria. It is also to enable the user, customers, and other authorized entity to determine whether to accept the system.

### Participants:

Mary Forde

### Methodology:

Acceptance Testing is the fourth and last [level of software testing](http://softwaretestingfundamentals.com/software-testing-levels/) performed after [System Testing](http://softwaretestingfundamentals.com/system-testing/) and before making the system available for actual use. [Black Box Testing](http://softwaretestingfundamentals.com/black-box-testing/) method is used in Acceptance Testing. I will be using ad-hoc when carrying out this part of black box testing for the game in order to randomize results that a user could receive.

## 4.5 Batch Testing

Definition:

Group of tests executing sequentially one by one is called Batch Testing. Every test Batch consists of multiple dependent test cases. In those batches every end state is a base state to next case.

Participants:

Rebecca Black

## 4.6 Automated Regression Testing

### 

### Definition:

**Regression Testing** is a type of software testing that intends to ensure that changes (enhancements or defect fixes) to the software have not adversely affected it .It is essential that regression testing is conducted to make sure that fixing one thing has not broken another thing. During regression testing, new test cases are not created but previously created test cases are re-executed.

### Participants:

Ethan Fahy

### Methodology:

Regression testing can be performed during any level of testing. It is essential to do an impact analysis of the changes to identify areas of the game that have the highest probability of being affected by the change and that have the highest impact to users in case of malfunction and focus testing around those areas.

## 4.7 Beta Testing

### Definition:

**Beta Testing** is one of the Acceptance Testing types. It adds value to the product as the end-user validates the product for functionality, usability, reliability, and compatibility. Inputs provided by the end-users helps to improve the quality of the product further and leads to its success. Since Beta Testing happens at the end user’s side, it cannot be the controlled activity.

### Participants:

Users, Mike Whatts, Shaun Purcell

### Methodology:

This type of testing is completed in five different steps

1) Planning

2) Participants Recruitment

50 – 250 users

3) Product Launch

* share the link from where they can download and install the game

4) Collect and Evaluate Feedback

* Feedbacks are evaluated to analyse and make out the customer to satisfy the game.
* Suggestions are considered to improve the product in its next versions.

5) Closure

* Once a certain point is reached and when all the features are working, no bugs are arising, and exit criteria are met then decide to conclude Beta Testing Phase.

# 5.0 Test Schedule

|  |  |
| --- | --- |
| Task | Duration |
| Review Test Document | 3 days |
| Train staff | 7 days |
| Create test cases for unit testing | 1 week |
| Run unit test cases | 1 week |
| Verify unit test | 2 days |
| Create tests for system integration | 1 week |
| Run tests for system integration | 5 days |
| Verify system test integration | 2 days |
| Create stress testing environment | 2 days |
| Run stress test | 2 days |
| Verify stress test | 1 day |
| Create beta testing requirements | 1 day |
| Write up documentation for beta testers | 5 days |
| Release game to Beta Testers for testing | 3 weeks |
| Collect and analyse data from beta testing | 1 week |
| Return data to developers for any errors | 2 weeks |
| Release game | 2 days |

# 6.0 Control Procedures

## Problem Reporting

Bug reporting will be completed using Jira bug reporting software. Jira uses a bug report process template which ensures that all bugs are reported and can be fixed.

The process for Jira consists of the following order:

1. STR- Steps to reproduce.
2. AR- Actual result
3. ER- Expected result
4. Platform
5. Browser
6. Software version
7. Comments

### Change Request Form

Jira will be used to manage change requests. A new issue type called Change request will be made in Jira. It should be created with the following steps:

* Project
* Issue type- Change request
* Summary of the request
* Priority
* Name of reporter
* Description of change request

The change request will be assigned to Mary Forde as she is the only team member with the ability to sign it off.

When Mary Forde has approved the change request, the tester will be notified and can now make the changes to the code and can close the request.

# 7.0 Features to be Tested

|  |
| --- |
| Unit scrolling script |
| Move forward |
| Move backward |
| Jumping |
| Attacking |
| Pause/Start |
| Health |
| Player death script |
| Player audio script |
| Enemy movement |
| Enemy attack |
| Enemy health |
| Enemy death script |
| Enemy audio script |
| Projectile movement |
| Play game button |
| Load game button |
| Exit game button |
| Music |
| Adjust sound effects level |
| Save game function |
| Settings button |
| The interface between the player character and the enemy |
| Interface between the main menu and level 1 |
| Interface between the pause/resume menu and each level |
| The interface between each level |
| Interface between the final level and the Final screen |
| Interface between the save game and the database |
| Beta testing |
| Regression testing on unit test cases |

# 8.0 Features Not to Be Tested

Features that will not be tested are the background image and the logo.

# 9.0 Resources/Roles & Responsibilities

|  |  |  |
| --- | --- | --- |
| Team Members | Role | Responsibilities |
| Mary Forde | Test Team lead and Project Manager | Write Test design conditions  Sign off on the exit of every stage  Write user acceptance guidelines  Run unit test cases |
| Amy Smith | Tester | Writes test cases for unit testing |
| Jack Davis | Lead Developer | Write test cases for system integration |
| Mike Whatts | Developer | Create stress test environment  Documentation for beta testing |
| Rebecca Black | Lead Quality Assurance | Run system integration testing  Run Batch Testing |
| Ethan Fahy | Quality Assurance | Run automated regression testing |
| Shaun Purcell | Front End Developer | Run stress testing  Analyse beta test results |

# 10.0 Schedules

### The deliverables before testing:

|  |
| --- |
| * Test plan document |
| * Prewritten test cases |
| * Test design conditions for test cases |

### The deliverables during testing:

|  |
| --- |
| * Test scripts |
| * Test incident reports |
| * Change request forms |
| * Bug reports |

### All documentation for beta testers:

|  |
| --- |
| 1. Set-up instructions |
| 1. Installation instructions |
| 1. How to play the game |
| 1. How to uninstall / delete the game |

### The deliverables provided after testing:

* Test result summary report

# 11.0 Risks/Assumptions

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Risk | Impact Level | Contingency Plan |
| 0.1 | Financial risk  risk of going over budget | high | pinpoint all essential items and team members and determine what can be discarded |
| 0.2 | Design risk  risk of a lack of user interest | high | Analyse beta findings and add or improve features |
| 0.3 | Market risk  risk that the market for the game will change before the game is released | Low | Analyse the market and use the findings to improve game for current climate |
| 0.4 | Technology risk  risk that any of the software in the game will fail | medium | Create back-ups of all files, Delegate member of team to oversee recovery tests |
| 0.5 | Schedule risk  risk of a delay in the release date | medium | Developers may need to work overtime |
| 0.6 | Quality risk  risk of game crashing with bugs and errors | high | All test cases should be re-run and all code should be tested by delegated staff until bug is found and fixed |

# 12.0 Tools

Below is a list of all the recommended software tools for testing:

* Jira – Unit Testing
* ReQTest – Beta Testing
* Ranorex Studio - Automated Regression Testing