Test Plan Template:

The Pixel Wizard

Prepared by:

David Morton

G00351634

14/05

# Table of Contents

[Table of Contents 2](#_Toc40374609)

[Introduction 3](#_Toc40374610)

[Objectives and tasks 3](#_Toc40374611)

[Objectives 3](#_Toc40374612)

[Tasks 3](#_Toc40374613)

[Scope 4](#_Toc40374614)

[General 4](#_Toc40374615)

[Tactics 4](#_Toc40374616)

[Testing Strategy 5](#_Toc40374617)

[Unit Testing 6](#_Toc40374618)

[Integration testing 7](#_Toc40374619)

[System Testing 8](#_Toc40374620)

[Performance and Stress Testing 9](#_Toc40374621)

[User Acceptance Testing 9](#_Toc40374622)

[Participants 10](#_Toc40374623)

[Methodology: 10](#_Toc40374624)

[Batch Testing 10](#_Toc40374625)

[Automated Regression Testing 10](#_Toc40374626)

[Beta Testing 11](#_Toc40374627)

[Test Schedule 11](#_Toc40374628)

[Control Procedures 12](#_Toc40374629)

[Problem reporting 12](#_Toc40374630)

[Change requests 13](#_Toc40374631)

[Features to be tested 13](#_Toc40374632)

[Features not to be tested. 13](#_Toc40374633)

[Resources, Roles and Responsibilities 13](#_Toc40374634)

[Software engineers 13](#_Toc40374635)

[Quality assurance 14](#_Toc40374636)

[Mobile developers 14](#_Toc40374637)

[External software engineers 14](#_Toc40374638)

[Hardware engineer 14](#_Toc40374639)

[Locations 14](#_Toc40374640)

[Schedules 14](#_Toc40374641)

[Risks and Assumption 15](#_Toc40374642)

[Assumptions made for testing 15](#_Toc40374643)

[Risks 15](#_Toc40374644)

[Contingencies 15](#_Toc40374645)

[Tools 15](#_Toc40374646)

# Introduction

The aim of this test plan is to test ‘The Pixel Wizard’. This is a 2d side-scrolling rpg cross platforming game. There are several main areas that need to be tested for this game before it can go be released.

Each of the main areas that will be tested, will have sub sections that will also need their functionality to be tested. The main areas are:

* The main menu
* The pause menu
* Character controls
* Items
* Enemy functionality
* How the level functions

# Objectives and tasks

## Objectives

The objective of these tests is to verify the functionality of The Pixel Wizard. To ensure that it meets the specifications outlined in the appendix. The test will test the scripts and fix any bugs encountered and then test for non-script related bugs.

The results of the test should be that the game is ready to launch and be given to the client and that the scripts are re-usable for future projects.

## Tasks

In this test plan the tasks are to:

* Test the functionality of the parts of the game
* Test the game in full
* Report problems back to the game designer

# Scope

## General

The main sections that will be tested will be the main menu of the game, the in-game pause menu, the character controls, items and enemy and level functionality.

Within those areas there is sub areas that will also be tested. For the main menu if a save file feature is implemented in the game the load a file feature needs to be checked to ensure that it loads the correct file to the point is was saved. The options were the game settings are set will be tested to ensure the settings can be changed and once the game loads these settings carry into the game.

The range of controls for the character will be tested. Movement in each direction, jump, attack and controls to open pause menu. There are several different types of pickups in the game, when the player interacts with these either with attacks or walking over them there needs to be two tests. One to ensure on collision they interact as expected and that once they are triggered, they function correctly. This includes pickups such as potions which will replenish the players health.

Enemies will be tested to ensure they function correctly, these functions include dealing the correct amount of damage, having the right health settings and dropping or giving the correct pickups and health. The first build of the game has a set layout for enemies, if these were to be randomised the script would need to be checked so that there is a limit to how many enemies are spawned for two reasons. To keep the game balanced and not overpower the player and secondly the game would crash.

The last area to test is the levels. Testing the level includes testing that the level assets are solid, and the player will not fall through the level. The platforming aspects also need to be checked, if the gaps are to big and the player is unable to jump and make the gap the game will be unplayable.

## Tactics

There will be several different testing phases carried out. There first will be unit testing. Then integration testing then system testing, then Performance and Stress Testing followed by User Acceptance Testing and finally batch testing.

The first thing to be checked will be the start menu. This alongside the pause menu will take the lowest amount of time to test as they do not interact with other elements of the game.

The time frame for testing the game menu will be two days. It will be tested in order it is displayed on the screen. During this time the sub menu of options/settings will be tested for functionality.

The next item on the list to be tested is the pause menu. There will be one day assigned to the pause menu. It has a smaller list of options and if the main menu is working a lot of the script is transferrable and applicable to the pause meu.

Three days will be assigned to testing the player and the control system. The main function of these tests is to ensure the player functions correctly. An example of this would be that when the move button is pressed, they move. This will also test that the health bar is working.

Two days will be assigned to testing the enemies. What will be tested is that their health decreases on impact of the players attack, that they are attacking correctly, dropping the right pickups and that they move around the scene correctly.

One day will be spent testing the level. This includes that the colours work correctly, that the terrain is correctly spaced, and the player is able to traverse the landscape correctly. Scene progression also needs to be tested that when a player completes the level they are brought to the next level, and also the save game function will be tested. Pickups will also be tested during this time.

Each of these sections or units will be tested on a computer for functionality. They will be tested on a phone as part of a system test. Before system testing can go ahead integration testing will take place. Integration is testing two or more units together to check they work and function correctly.

Seven days will be spent on integration testing overall. This will be broken down into sections, where two components will be tested and once the tests are successful another component will be added, and this will be repeated until the game has been tested in full.

Once integration testing has been completed, there will be six days allocated for system testing. This will include testing the game on a computer and mobile device. Then there will be five days allocated to performance and stress testing and four days allocated to user acceptance and batch testing.

After acceptance testing, five days will be given to conduct regression tests. Three weeks will be allocated to beta testing and then a week assigned for the handover process to give the software to the company.

The last test will be the beta test which is expecting to take four days and once this has been completed the game will be ready for launch on both devices.

The way these tests will be tracked and check of will be both physically and digitally tracked. There will be a chart set up in the main office with a breakdown of each section that is to be tested and a check list accompanying it to mark of if it has been completed, if the test is a fail or a pass, if the test has not been done yet and how many times it has been tested. If it has failed there will be a chart recording the changes that are necessary to fix the game.

Digitally the way this checklist will be done is by a shared excel sheet. This will be shared on google docs. Every member on the team will have access to this and they will be able to add the entries for their section.

# Testing Strategy

## Unit Testing

### Definition

Unit testing is testing the individual units or components of a piece of software. The aim is to validate that each unit of the code performs as expected. Each line of code will be run at least once to ensure that it functions. This statement coverage ensures that an overhaul of the code is not needed and only changes the broken line of code. Branch testing will also be carried out, these is the testing of each branch of code a function may run through but does not necessarily run through. Such as an if or if else statements. These types of statements have various conditions that can be met resulting in a variety of outcomes. For any existing branches each branch will be tested to determine if the desired outcome is given for each possible outcome for the branch.

The initial unit tests will have a binary pass system of fail or pass. A line of code that fails a test will be altered and re-ran until it works. Once the unit works in full it will then need to have a pass rate of nine out of ten times to be considered passed.

### Participants

There will be four participants for unit testing.

Mary Smith, Joe Blogs and Mark Anthony who are all software engineers. Mary and Joe will be responsible for writing and carrying out the tests. Mark will be responsible for making the necessary changes to the script after tests have failed.

Ann McDermott from the quality assurance team will fill out the charts with the results and figures of the tests. She will document the work as it is being carried out and work in constant communication with the team.

### Methodology

The main units to be tested are main menu, pause menu, character, enemy and level.

Mary and joe will write the tests and the scripts to conduct them. Mark will re-write any scripts that need to be altered if they fail. Ann will be documenting the progress and carrying out the write up for each unit.

The tests for the menus will be similar where the button will be pressed and the outcome noted.

A menu test would look like:

|  |  |  |  |
| --- | --- | --- | --- |
| Button | Expected result | Actual result | Pass/fail |
| Option | Options menu loads | Option menu loads | Pass |

The assumption on the buttons is that they have an onclick function that triggers the script. Each function would have a key word that makes it recognizable to the button on the menu that would identify it in the script. The second assumption is that the save/load feature is set up to save to local memory and that the test is to make sure that the buttons work first and then that the data is being saved.

The next unit to be tested is the player. The main aspect to be tested for the player during unit testing is the control system as the other tests that need to be conducted need other units to be carried out.

A control test would look like

|  |  |  |  |
| --- | --- | --- | --- |
| Button | Expected result | Actual result | Pass/fail |
| Left arrow | Player moves left | Player moves left | Pass |

For this test the assumption is that the testers have access to the control system and that once the unit works on a computer it can be tested and de-bugged on a phone during system testing.

The unit testing for the enemies and most of the level design will be minimal as many of their functions depend on relying on other units. Another assumption is that the enemies have code to generate them at specific locations and a specific number of them. So there will be no need to take into account random generation tests. The final assumption is that without a player in the level the enemies will walk towards where they would come from, and that if the player is in the scene the A.I. directs the enemy towards the player.

A unit test for an enemy would look like

|  |  |  |  |
| --- | --- | --- | --- |
| Enemy | Expected behaviour | Actual behaviour | Pass/fail |
| Zombie | Appear centre screen/walk left | Appear centre screen/walk left | Pass |

The unit testing for the levels will check that pick-ups and colours work as expected, the remaining level tests will be carried out during integration testing. The assumption for the scene is that the pickups are placed in the level and are not generated using code.

## Integration testing

### Definition

Integration testing is where individual units are combined and tested as groups. This is to check for faults in the interaction between tested groups. This will test the parts of the game that were unable to be tested during unit testing.

The test will be determined to be a success/pass if the units that are being tested work together. As with unit testing once the test has worked initially the second stage of testing will need to pass nine out of 10 tests.

The type of integration testing that will be implemented is Big Bang testing. The reason that this method has been chosen is that the team have received all the software at once and are able to combine multiple or all units at once to test them

### Participants

Mary and Mark from unit testing will be involved with the integration testing and will be joined by Sarah Doyle. Sarah is a software engineer. Sarah and Mary will be doing the writing and testing of the scripts with Mark making adjustments to any statements that fail testing.

Ann will document the tests and do the paperwork as she has done for the unit testing.

### Methodology

This will be varied from the unit testing as multiple units will be tested at the same time and if a test fails both units need to be inspected to check for errors. This again will be conducted Statement by statement.

A test including the menu and game testing the start button will look like the following:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test | Unit 1 | Unit 2 | Expected behaviour | Actual behaviour | Pass/fail |
| Start button | Menu | Levels | Level one loads | Level one loads | Pass |

A test inclusive of more units will have more aspects to be tested, an example of a player dying in the game with the expected out-come of returning to the main menu will require most of the units to be set up and would look like the following:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Test | Unit 1 | Unit 2 | Unit 3 | Unit 4 | Expected behaviour | Actual behaviours | Pass/fail |
| Health goes to 0 | Player | Enemies | Level | Menu | Player goes to menu | Player goes to menu | pass |

Integration will continue until all the units are integrated successfully together. Once integration testing is deemed to be a success the next phase will be system testing.

## System Testing

### Definition

This will be testing the software on a computer and a mobile device as these were specified as the devices that the game will be released on. Success will be defined as the game runs start to finish with a maximum of six graphical glitches and one code glitch.

### Participants

Mary and Sarah will be writing and conducting the tests on a computer. Steve Bigsby and Tom Bill who are both mobile app developers will conduct the tests on an android device and an Iphone.

Ryan Slater and Aine Potter are also software developers who will work on the code to fix and failed tests for the computer version and the mobile version respectively.

As with all the tests and testing phases Ann will document all the results.

### Methodology

The first thing that will be tested is that the game runs and loads on both devices. There will be an assumption here that the devices being used meet the minimum requirements to run the game.

Once the game loads the testers will go through every iteration of possibilities on each device to check for functionality. This will include testing the control system and the control system for mobile devices is onscreen buttons.

Once all the functions have been tested some non-functional elements will be tested. These will be mainly on the mobile devices to check what happens to the game if a message or phone call comes through while the player is playing.

Once system testing has been deemed a success the next stage will be user acceptance testing.

## Performance and Stress Testing

### Definition:

Performance testing is a software test to determine how the system performs under a certain workload

Stress testing is testing the system and causing a spike in activity to create a heavy load.

Performance testing will be deemed a success if the software can run from start to finish under normal circumstances on both desktop and mobile devices without crashing. Another result of the test is to record the ram, cpu and other processing power.

There will be no pass or fail on the stress test the stress test will be used to determine at what point the game crashes or caused the system to crash.

### Participants

Mary and Steve will conduct the performance testing as it will have similar tests to that of the system tests that they have carried out.

Sarah and Tom will conduct the stress tests on the systems they carried out their system tests on.

Mark and a hardware engineer named Lisa Bond will carry out the changes necessary to either the code or to the systems.

Ann will record the results, these will include the processing units the device has used to run the game, if the device crashes and what causes the device to crash.

### Methodology:

To run the performance, test the game will be run from start to finish. The tester will use MSI afterburner during the game. This is a screen overlay that shows the computers current process power there is also a mobile one available that will be used on the mobile devices. The testers will make note of the current performance levels during different stages of the game and tests.

These will range from simple actions such as pressing a button or moving the character to more complicated and power consuming operations such as fighting the enemies and changing level.

For stress tests the testers will create spikes in heavy workloads to determine what the system can handle and find out what would cause the game to crash. The tests will be simulated artificially such as adding more enemies to the level or screen then would usually be expected or let the score continually run up.

The results that will be documented will be the minimum number of enemies onscreen to cause the game to crash other numerical data that is relevant to any crashes in the software or system.

## User Acceptance Testing

### Definition:

This is the last phase of the software test. This is where software users are invited to test out the software in real world scenarios. This will be the most critical test in the software plan.

Success will look like the software matches or exceeds the initial requirements. Failure would be the software not meeting the initial requirements.

## Participants

For this test four external software users will be invited to test the game. Michael James, Michelle Jones, Adam North, Alison West. They will be responsible for testing the game on computers and mobile devices.

Software engineers Mary and Tom will be taking part to make adjustments to the game if any bugs are discovered.

Ann will document the results of the tests.

## Methodology:

Michael and Michelle will test the software on computers and Adam and Alison will test the software on mobile devices.

An example of a test for user acceptance will look like:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test | Step | Expected Results | Actual result | Pass/fail |
| Software runs as expected | Running software from start to finish | Software meets initial expectations | Software meets initial expectations | Pass |

The users will test the game and report bugs to Mary and Tom who will work on the code to fix the bugs. Once the bugs have been fixed the software users will test the game again. Once the software meets the initial requirements of the game it will be tested for reliability.

## Batch Testing

Batch testing is performed by running the entire test set again on the software. This is to ensure that the software that is being tested passes all sequential test in order. Where one test will not start until the software has passed the last test.

There was consideration for batch testing in this test plan. Batch testing was determined to be redundant due to the small size of the software being tested as passing each other test would ensure the software would meet the requirements.

## Automated Regression Testing

### Definition

Regression testing is testing the software to confirm that a recent change to the program or code has not affected existing features

### Participants

Software engineers Mark, Mary and Tom will conduct regression testing.

Ann will document the results of the tests.

### Methodology

There are two ways to conduct regression testing. Retesting all and regression test selection. Retesting all re-runs all the tests in the test suite, however this is very expensive and time consuming. Test selection is where the software engineers categorise tests into two lists, obsolete and reusable. Then all the reusable tests are run.

Due to the size of the project and the resources available the latter method will be used.

The engineers will set up the software in full on both a computer and mobile device. Then each test that has been deemed reusable will be ran.

## Beta Testing

### Definition

Beta testing is carried out by members of the target audience for the software. The software is sent out to a number of users who install and run it as it is intended. They then fill out a report of how the software preformed and report any bugs and suggestions to make to the software.

### Participants

The participants of the beta test will be members of the general public. There will be fifty people testing the software on a computer and fifty people testing the software on a mobile device.

### Methodology

The software will be advertised on the relevant software marketplace for computers and mobile devices for users to download for. The software will be available for three weeks, during this time the users will be asked to fill out a form that is provided with each download of the software.

This form will have sections to report bugs found and recommend any suggestions to make for the software.

# Test Schedule

The testing will start once the software and requirements are passed to the software engineers from Game Development International Ltd. The main milestones will be completing each type of test with the final milestone completing the brief and providing Game Development International Ltd with software that has passed all the tests.

The first milestone will be completing unit testing. The estimated time to complete unit testing is nine days. Unit testing along with most testing will be conducted in a computer lab. There is four people involved in completing this milestone, Mary Smith, Joe Blogs and Mark Anthony who are all software engineers and Ann McDermott from the quality assurance team who will document the testing. The nine days of unit testing have been broken down into individual time frames for each separate task.

The second milestone will be completing integration testing. The estimated time frame to complete this stage of testing is seven days. Integration testing can only be started once each unit has passed unit testing as this is the stage when the units are built together and tested to confirm that they can function together. This will also take place in the computer lab. Involved in integration testing are Mary and Mark from unit testing and will be joined by Sarah Doyle who is a software engineer, Ann will be completing the documentation for this test.

The third milestone will be the completion of system testing. The estimated time to complete system testing is six days. This will be testing the software on both mobile and computer devices. This will take place in two computer labs one designed for computer testing and the other set up for mobile software testing. Mary and Sarah will be writing and conducting the tests on a computer. Steve Bigsby and Tom Bill who are both mobile app developers will conduct the tests on an android device and an Iphone. Ryan Slater and Aine Potter are also software developers who will work on the code to fix and failed tests for the computer version and the mobile version respectively. As with all the tests and testing phases Ann will document all the results

The next milestone is the completion of performance and stress testing. The estimated time to complete these sets of tests is five days. They will be testing how the software affects the performance of the system and how the software responds under a heavy load. This will take place in the same computer labs as system testing. Mary and Steve will conduct the performance testing as it will have similar tests to that of the system tests that they have carried out. Sarah and Tom will conduct the stress tests on the systems they carried out their system tests on. Mark and a hardware engineer named Lisa Bond will carry out the changes necessary to either the code or to the systems. Ann will do the documentation.

Four days will be allocated to the completion of acceptance and batch testing. The completion of these tests will be the fifth milestone during the software testing. For acceptance testing, four external software users will be invited to test the game. Michael James, Michelle Jones, Adam North, Alison West. Ann will document the results.

The three remaining milestones are regression testing, beta testing and the handover of software to the company. Regression testing is estimated to take five days to complete. Software engineers Mark, Mary and Tom will conduct regression testing. Ann will document the results of the tests. These will take place in the two labs as the tests will need to be conducted on both systems.

Beta testing is the longest testing phase as this is where to software is distributed to members of the target audience to test the software how it would be tested under general use. The beta testers will fill out a form to document the results. This test will take place in user’s homes. There will be one hundred participants, fifty to test on each system.

The final milestone will be finishing and presenting the software to Game Development International Ltd. A week will be spent finalizing the details to hand the software back with all the accompanying paperwork.

# Control Procedures

## Problem reporting

During each phase of testing there is a document which Ann, a member of the quality assurance team will fill out and document the results of each test. The form will have several sections, test that is being carried out, what the expected result is, what the actual result is, who performed the test, where and when it was preformed and if it is a pass or fail. Depending on the type of test there will be more sections on the document.

Bugzilla is a bug tracking piece of software that will be used throughout the software testing. The reason that this software has been chosen is that it has minimal impact on the system that is testing the software while utilising an advanced search system and a comprehensive reporting tool that generates charts and reports.

In unit testing the document will include the unit being tested, similarly to integration testing the names of the units being tested will be included. For the remainder of the tests the device will be listed in the document whether it be Computer or mobile device will be listed and this will also require the operating system to be filled out.

An example of this document

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Date: | Location | Tester | Test | Unit | Expected result | Actual result | Pass/fail |
| 11/03 | Lab 1 | Mary | Play on start menu brings to first level | Main menu | Loads first level | Changes scene to first level | Pass |

## Change requests

During each phase of testing there is a software engineer who will be responsible for making changes to the code if a test fails. Each engineer will make a change to the code and make a note of it. The final documentation will include the successful changes to reduce redundancy in the documentation. The document will include what unit the code is not working in, which specific code is not working, and the changes made to make the code work. The engineer will also include an update in the script file as a comment to show what has been changed and what the change is.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Engineer | Code | Change needed | Change successful | Date updated | Unit | Testing phase |
| Mark | Change scene to level 1 | Wrong name in code | Yes | 11/03 | Menu | Unit testing |

# Features to be tested

* Every option of main menu
* Every option of sub menu
* Player controls
* Pause menu
* Health of players and enemies
* Enemy actions
* Pickups
* Level, graphics, layout, and intractability
* Save/load feature

# Features not to be tested.

* Interference of notifications on mobile devices
* Background programs on computer
* Internet connection being present

# Resources, Roles and Responsibilities

## Software engineers

* Mary Smith- Unit testing – Integration testing – System testing- Regression testing – Writing code- Testing code- Fixing reported bugs
* Joe Blogs – Unit testing – Writing code- Testing code
* Mark Anthony – Unit testing – Integration testing- system testing- Performance testing - Stress testing - Regression testing- Writing code- Testing code- Changing code that fails tests
* Sarah Doyle – Integration testing – System testing – Performance testing – Stress testing - Writing code- Testing code
* Ryan Slater - System testing- Writing code- Testing code – Correct code on failed tests
* Aine Potter - System testing - Writing code- Testing code - Correct code on failed tests

## Quality assurance

* Ann McDermott, will document the results and complete the final documents for each phase of testing

## Mobile developers

* Steve Bigsby – System testing – Performance testing – Writing code – Testing code
* Tom Bill - System Testing - Regression testing – Stress testing- User acceptance testing – Writing code – Testing code

## External software engineers

* Michael James – User acceptance testing – Writing code – Testing code
* Michelle Jones – User acceptance testing – Writing code – Testing code
* Adam North – User acceptance testing – Writing code – Testing code
* Alison West – User acceptance testing – Writing code – Testing code

## Hardware engineer

* Lisa Bond - Performance and Stress Testing – adjusting and fixing Hardware

## Locations

* Computer lab – Ten computers
* Mobile system lab – Five Android devices – Five Ios devices

# Schedules

The documents that need to be delivered include test plan, test cases, test incident reports and summary reports.

The test plan will be created and presented before testing starts to lay out the steps that will be taken to complete the test.

Test cases will be created during each stage of testing. A test case will be created before each individual test to outline what the test case will be testing and what the expected results will be. Once the test case has been carried out it will be completed to show if the test was a success or failure.

Test incident reports will be created at the end of each test. These will document any incidents that need to be reported on. Including, systems failing and breaking, any mis conduct from employees that arise.

Test Summary reports will be a synopsis of what occurred during each testing phase. What worked, what failed and what had to be changed to ensure that the software that was being tested passed.

# Risks and Assumption

## Assumptions made for testing

* Each system used met the minimum requirements for running the software
* The testers have been given the control system for the game and other relevant information to help complete testing
* That all the buttons have an onlick() trigger function
* That the save/load feature works with local storage
* That enemies are not generated randomly and will always generate in the same location
* That when a player is not in a scene, enemies will move towards where the player will come from
* The term game and software are interchangeable terms in this test plan

## Risks

* Lack of personal resources
* Lack of required hardware
* Late delivery of software
* Changes to requirements or designs
* Complicated code

## Contingencies

* Additional staff available from other departments
* Overtime and night shifts will be made available
* If the requirements change a new delivery date will be implemented
* An extra computer lab will be made available for testing

# Tools

* Bugzilla
* MSI Afterburner
* TestingWhiz – used for mobile game testing
* TestComplete – used for automated testing
* Tosca Testsuite – another automated testing tool