Test Plan

**Product Name:** The Pixel Wizard

**Prepared by:** Gearóid Reilly

**Date:** 12/05/2020

Contents

[Introduction 1](#_Toc40187975)

[Objectives & Tasks 1](#_Toc40187976)

[Objectives 1](#_Toc40187977)

[Tasks 1](#_Toc40187978)

[Scope 2](#_Toc40187979)

[Front-End Menu 2](#_Toc40187980)

[In-Game Menu 2](#_Toc40187981)

[Game Controls 2](#_Toc40187982)

[Gameplay Features 2](#_Toc40187983)

[Testing Tactics 3](#_Toc40187984)

[Testing Strategy 5](#_Toc40187985)

[Unit Testing 5](#_Toc40187986)

[System and Integration Testing 6](#_Toc40187987)

[Performance and Stress Testing 6](#_Toc40187988)

[User Acceptance Testing 7](#_Toc40187989)

[Batch Testing 7](#_Toc40187990)

[Automated Regression Testing 8](#_Toc40187991)

[Beta Testing 8](#_Toc40187992)

[Test Schedule 9](#_Toc40187993)

[Control Procedures 9](#_Toc40187994)

[Problem Reporting 9](#_Toc40187995)

[Change Requests 9](#_Toc40187996)

[Features to be Tested 9](#_Toc40187997)

[Features not to be Tested 10](#_Toc40187998)

[Roles & Responsibilities 10](#_Toc40187999)

[Schedules 11](#_Toc40188000)

[Risks/Assumptions 11](#_Toc40188001)

[Tools 12](#_Toc40188002)

# Introduction

This document describes the test plan for the game project “The Pixel Wizard”, this game is a side scrolling platformer where the player controls a wizard who must navigate through the levels by defeating all of the enemies in each level. The player can move through the levels, shoot magic at enemies, avoid obstacles, and find pickups such as health pickups that can restore health.

The game contains a menu where the player can start the game, open up game settings and exit the game, the player will also be able to load their previous progress from a save file and they can delete their save file.

# Objectives & Tasks

This section will detail all the objectives of this test plan and the tasks which will be taken to accomplish these goals.

## Objectives

The objective of this test plan is to ensure the functionality of “The Pixel Wizard” works according to the specifications detailed in the game requirements document. This test plan will test the functionality of the front-end menu components, the in-game screens such as the pause menu, the player input controls and game features such as enemy combat and level progression.

This test plan will also list all the tasks and responsibilities needed to complete the testing for “The Pixel Wizard”. This document will be used to communicate the testing that is required for this product and to provide a series of reusable test scripts that can be utilized for test execution.

## Tasks

Below are all the tasks that will be involved in the test plan for this product:

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

The contributors to the test plan are as follows:

* Gary Garland
* Penny Quarters
* Frank Frankton
* Jack Jackson
* Amy Hendricks
* Hugh Mann

# Scope

This section will describe all the components of “The Pixel Wizard” that will be tested with this plan.

## Front-End Menu

The front-end menu is what the player will see upon starting the game, the menu will provide 3 options to select from, “Play”, “Settings” and “Exit Game”. By selecting “Play”, the player will begin the game at the first level, selecting “Settings” will allow the user to edit game settings such as sound and music volumes, selecting “Exit Game” will quit the application.

## In-Game Menu

The player can pause the game during gameplay and a menu with similar options to the front-end menu will appear, no other game features will be active while this pause menu is open. The pause menu will provide 4 options to choose from, “Resume”, “Settings”, “Restart” and “Exit Game”.

If the player selects “Resume”, the game will resume, and the player will continue from when they paused the game. Selecting “Settings” will open game options like those in the main menu and allow the player to change sound and music volume. Selecting “Restart” will restart the current level, the player will be sent back to the starting point of the level, the player’s health will be reset and all enemies and pickups will be set back to their initial points in the level. If the player selects “Exit Game”, they will exit the game level and be brought back to the main menu.

## Game Controls

The game has 2 input controls, one for the computer version of the game and one for the mobile version of the game. The controls for the PC version are managed through keyboard inputs and the controls for the mobile version are managed through on-screen buttons that the player can touch.

|  |  |  |
| --- | --- | --- |
| **Action** | **Computer** | **Phone** |
| Move Forwards | Right Arrow Key/D Key | Right Arrow on Screen |
| Move Backwards | Left Arrow Key/A Key | Left Arrow on Screen |
| Jump | Up Arrow Key/W Key | Up Arrow on Screen |
| Crouch | C Key | Down Arrow on Screen |
| Attack | Left Mouse Click/R Key | Dedicated Button on Screen |
| Pause/Resume | Spacebar | Top Right Button on Screen |

## Gameplay Features

**Player Controls:**

The player can move the character forwards and backwards, jump and crouch using the mention controls above. The player should be able to navigate through the levels by moving and jumping on platforms when needed. The player can also shoot projectiles with a delay time between each shot.

**Player Health:**

The player has health is displayed on the top left side of the screen represented by red diamonds. The player can lose health by taking damage from enemy attacks. If the player loses all their health, the player loses the game.

**Enemies:**

Each level contains enemies that can move towards the player and shoot their own projectiles at the player, enemies will be damaged when the player’s projectiles collide with them. Enemies have their own health and will be removed from the level if they lose all their health. At the end of each level there is a boss enemy whose health is displayed in the top right corner of the screen represented with blue diamonds, like the player’s health.

**Levels:**

There are multiple levels that the player must progress through, when the player completes 1 level, they move to the next one until they reach the end of the last level where they win the game. Platforms, enemies, and pickups are in each level which add different challenges to the player.

**Pickups:**

When the player interacts with a health pickup, it should restore some of the player’s health.

## Testing Tactics

Below will list how we intend to test some of the features listed in the scope. Some features may be tested simultaneously because of 1 test case. We will follow our test schedule to ensure that the features are tested, and the product is suitable for release for customers.

|  |  |
| --- | --- |
| **Test Item** | **Test Method** |
| Front End Menu | Click each menu item to test if they work. Log messages to test functionality before implementing other components.  Verify the “Play” button brings the player to level 1  Verify that the player can change the sound and music volume in the main menu after clicking “Settings”  Verify the application closes after the player clicks the “Exit Game” button. |
| In Game Menu | Verify the player can pause the game by pressing the pause button.  Verify that neither the player character nor enemies can move while the game is paused.  Verify that the options function by clicking each button. Add debug logs to test each button before implementing other features.  Verify that clicking the “Resume” button will resume the game.  Verify that the player character and enemies can move again after the game is resumed.  Verify that the player can adjust sound and music volumes after clicking “Settings”.  Verify that the player is brought back to the beginning of the level after selecting “Restart”. Ensure that player health is reset, enemies and pickups are added back to their original points in the level.  Verify that the player is brought back to the main menu when they click the “Exit” button. |
| Game controls | Verify that the game responds to player input on computer and mobile. Add test logs to check if input is received before adding gameplay features.  Verify that the player moves when pressing the movement keys.  Verify that the player can shoot projectiles using the dedicated buttons. Check that there is a delay between each shot. |
| Player Health System | Verify that the player loses health when they collide with an enemy or enemy projectile.  Verify that the health UI updates when the player gains or loses health. |
| Pickup System | Verify that the player can touch the pickup.  Verify that the health pickup increases the players health.  Ensure that the pickup is destroyed when the player collides with the pickup. |
| Enemy System | Ensure that enemies can shoot projectiles.  Verify that enemies can take damage from the player’s projectiles, ensure that the player projectiles are destroyed on collision with the enemy.  Verify that an enemy is removed from the level when lose all their health.  Test that the level boss health updates when they take damage.  Verify that the enemies can move on platforms. |
| Level System | Verify that the player moves to the next level after the boss is defeated.  Ensure that the player wins the game when they finish the last level. |

# Testing Strategy

Below is our strategy and approach to all the different phases of the testing process and who will be responsible for that phase.

## Unit Testing

**Definition:**

Unit testing is where the smallest components of a software program are tested individually, these tests are designed to ensure that each independent part of a program are functional before incorporating the piece of the software together. These tests are separate for each feature or component and must not rely on any other part of the software to perform tests on.

**Participants:**

Gary Garland and Jack Jackson

**Methodology:**

Both Gary and Jack will test each of the individual features of the game, Gary will test controls and game interactions on the computer while Jack will test game interactions on the phone. The testing scripts will contain stubs to manage interactions with separate components that will be integrated at a later point. Debug logs will also be used to verify that certain inputs work for certain features like control input.

## System and Integration Testing

**Definition:**

Integration testing is where the individual components of a software are combined and tested together. Some of the stubs that would be present before have functionality added to them referencing other parts of the system that were added. This testing also exposes the faults in the interfaces between components.

System testing is where the whole system has been compiled and tests are run to ensure that it complies with specified requirements

**Participants:**

Penny Quarters and Jack Jackson

**Methodology:**

We plan to use a bottom up approach to integration. Penny and Jack will work on gradually combining the components of the game together, Penny will test on the computer while Jack will test on mobile. Both will record what changes occur and continue until the whole system is created. At that point, Penny will run tests on the whole system to ensure that it meets specifications provided to her by Hugh.

## Performance and Stress Testing

**Definition:**

Performance testing is determining the speed, responsiveness, and stability of a program while it is under a workload. This form of testing is mainly used to see if the software can function while under a heavy workload and to expose any errors that could appear as a result.

Stress testing is where the software is put under an extreme workload beyond what it is capable of to determine how it will respond, this is mainly used to see how badly the system will crash or identify any other errors that could arise in this situation.

**Participants:**

Frank Frankton

**Methodology:**

Frank will test the performance of the game by measuring how fast the game responds to his inputs, he will also try running the game at different resolutions and having extra programs running at the same time to determine how those aspects could affect how the game performs. Frank will also run tests increasing the load on the game to identify the breaking point of the game.

## User Acceptance Testing

**Definition:**

User acceptance testing is a process to verify the usability of a software for the user, this is done to ensure that the software is ready for operational use. Real world software users operate the software to verify that it meets requirements for the intended users.

**Participants:**

Amy Hendricks and Alpha Testers selected by Amy

**Methodology:**

Amy will enlist the help of 5 testers for Alpha testing, she will be present during the testing procedures to answer any questions regarding the intended functionality of the game. She will also record issues that the testers face during the testing process.

## Batch Testing

**Definition:**

Batch testing is a group of tests executed sequentially; these tests consist of multiple dependent test cases where every end state of a case is the base state of the next. If one script fails or passes, it will lead to the whole batch passing or failing.

**Participants:**

Frank Frankton

**Methodology:**

Frank will identify test cases relating to game progression such as defeating the boss to progress to the next level and see how the cases are connected and run those tests. Some features of the game cannot be tested independently so batch testing is required to test a sequence of events.

## Automated Regression Testing

**Definition:**

Regression testing is running previous tests after changes have been made to the software to confirm that those changes have not affected any other part of the software. This process is necessary to identify any unexpected changes which could occur because of changes.

**Participants:**

Gary Garland

**Methodology:**

This process will be automated as these tests will likely be run several times over the testing period. Gary will identify what parts of testing the game can be automated and create test cases which are designed to be used with an automated testing software. Once this process is complete, any time changes are made to the game, the tests will be run to identify any potential errors.

## Beta Testing

**Definition:**

Beta testing is an acceptance testing process to evaluate the level of customer satisfaction with the product. This is done to evaluate if the software is ready for operational and commercial use. The testers are provided a beta version of the software and they operate independently it in a real-world scenario and encouraged to provide feedback which will be used to improve the product further.

**Participants:**

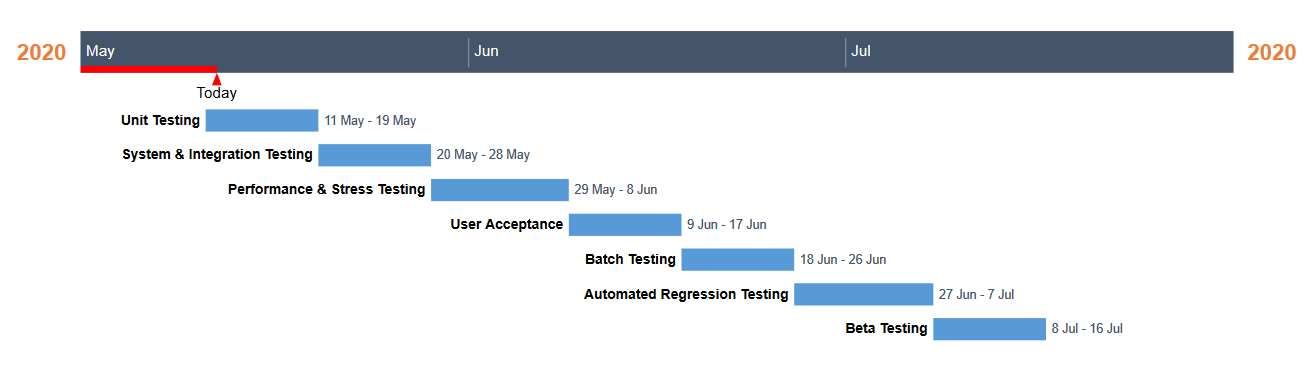
Amy Hendricks and the selected beta participants

**Methodology:**

Amy will have selected several beta testers who will all operate the game without her there to guide them. Amy will send the testers a link to an online form to provide feedback on the game. The beta testing period will last for a week after which Amy will gather the results from the forms and create a report which she will send to Hugh and he will organise changes that need to be made to the game and the testing team will write tests to verify the changes work correctly.

# Test Schedule

We devised a test schedule which assumes that each task will take roughly a week to accomplish. We feel that this should be enough time given that this game is a relatively small project.



# Control Procedures

Below are the procedures that should be followed in the event of a problem occurring or if a change to the project is required.

## Problem Reporting

In the event of a problem occurring during testing, the tester must write down the issue that occurred regardless of the size of the issue to maintain the best level of quality throughout testing. Issues should be recorded on a problem sheet form which must be filled out in these circumstances, the form will be sent to the project manager and they will send the details of the problem to the relevant department.

## Change Requests

Any change requests should be written on a submission form and sent to the project manager, only the project manager can sign off on changes made to the product as they must discuss any changes that would be made to the client.

# Features to be Tested

* Front-End Menu
* Game Exit Features
* Pause Menu
* Restart Level
* Player Movement
* Player Health
* Enemy Movement
* Enemy Attacks
* Boss Health
* Level Transition
* Pickup Interactions
* Sound and Music settings
* Game Performance
* Operation on both Computer and Mobile versions

# Features not to be Tested

* Saving and Loading System
* File Deletion System
* Application interruption, e.g. Player clicks away from the game.
* Game over system

# Roles & Responsibilities

|  |  |  |
| --- | --- | --- |
| **Name** | **Role** | **Responsibilities** |
| Gary Garland | Computer Tester | Unit Testing  Testing features specifically on computer  Managing Automated testing tools |
| Penny Quarters | System Tester | Testing system while the game is active  Integration testing |
| Frank Frankton | Performance Tester | Performance testing  Batch Testing  Checking responsiveness of controls. |
| Jack Jackson | Mobile Tester | Unit Testing  Testing Features specifically on the mobile  Integration testing for mobile |
| Amy Hendricks | User Tester | Manages group for Alpha testing.  Recording results from beta testing. |
| Hugh Mann | Project manager | Manages the overall operations of the project.  Receives reports and relays it to the relevant members.  Main contact between client and testers. |

# Schedules

Below are the identified deliverables for the test plan documents along with the planned delivery dates.

|  |  |
| --- | --- |
| **Deliverable** | **Expected Delivery Date** |
| Test Plan | 12/05/2020 |
| Test Cases | 18/05/2020 |
| Test Incident Reports | 16/07/2020 |
| Test Summary Reports | 16/07/2020 |

# Risks/Assumptions

For this test plan, we have identified some risks that could appear during testing and we have defined strategies should those issue appear during the testing process.

|  |  |
| --- | --- |
| **Risk** | **Solution** |
| Data Loss or Corruption | Have files backed up on the local computer to minimise game data being lost permanently.  Save files every day on the cloud to ensure that data can be accessed from any computer. |
| Hardware Crash or Malfunction | Have spare computers to use in the event of a crash to use for testing while the hardware is being fixed.  Files are backed up on the cloud to allow the spare computer to run the tests. Spare computers have necessary programs installed.  Software installers are saved for quick hardware restoration. |
| Delay in Tasks | Pinpoint where the delays are occurring and solve the problem.  Have testers sign task completion forms to indicate what progress has been made which will be compared to the planned schedule.  If deadlines are approaching, testers will likely have to work night shifts to meet mandatory test requirements according to the schedule. |
| Insufficiently Skilled Testers | All Testers will undergo training to ensure a sufficient base level of quality. |

# Tools

|  |  |
| --- | --- |
| Hardware | Computers of varying specifications, we can establish minimum hardware required for the pc version of the game.  Smartphones of varying specifications, like with computers we can determine the minimum specifications for the phone. |
| Bug Tracking Tool | **Bugzilla:**  Easily accessible as it is a web-based bug tracking tool. The tool is open source which reduces costs of tools. |
| Test Case Management Tool | **Testopia:**  An extension to Bugzilla providing a convenient management tool for writing test cases. |
| Automated Testing Tool | **T-Plan:**  Automated testing that can be used for both desktop and mobile games. Tool can be utilized for both versions of the game. |