The Pixel Wizard

TEST PLAN

Michael Whelan
G00351979 | 13/05/2019

Contents

1.0 Introduction

This document is a test plan for the 2D side-scrolling platformer “The Pixel Wizard”. The game will be developed by Game Development International Ltd for PC and mobile platforms.

The game will allow the player to control a wizard and navigate through several levels that get more difficult as the player progresses. The player can move, jump, crouch their way through each level, attacking enemies with magic as they go. The player will also be able to pick up health packs to help them through the level.

The game contains menus that the player can navigate through such as the main menu and pause menu. These menus will allow the player to play or exit the game and change the settings such as the sound levels and music levels. When the player begins the first level, they will be shown some text on screen which shows them what the controls are.

2.0 Objectives and Tasks

2.1 Objectives

The primary objective of this test plan is to ensure that the game meets the specifications detailed in the requirements document. At the end of the game development cycle, the user should find that the project has met all expectations as detailed in the requirements document.

The test plan will test the functionality of the various components such as the front end, the in-game menus, the control mechanisms, and game features.

The secondary objective of this test plan is to identify issues and bugs, communicate all known issues to the development team and ensure that all issues are addressed in an appropriate matter before release.

2.2 Tasks

These are the tasks that will be used in this test plan:

- Unit Testing
- System & Integration testing
- Performance & Stress Testing
- User Acceptance Testing
- Automated Regression Testing
- Beta Testing

3.0 Scope

Below is a list of the components that will be tested:

Front End

- Ensure all menu items are clickable
- Play button should load the first level
- Settings button should open the settings sub menu
- Settings sub menu contains music and sound level options
- Music and Sound level options work correctly
- Load Game should open a sub menu with previous save points
- Previous saves should be clickable and bring user to correct save state
- Delete game should open a sub menu with previous save points
- Previous saves should be clickable and allow user to remove them
- Exit game should close the game and return the user to their desktop/home screen

In-Game Menu

- Resume option should resume the game from the point the player paused
- Save option should save the users progress through the level
- Settings should open a sub menu
- Settings sub menu should have options for music and level sounds
- These sub menu items should increase and decrease their respective volumes
- Restart option should reset the player and enemies' positions to the start and reset pickups
- Exit game should bring the user back to the main menu outlined in the front-end section of this document

Control Mechanisms

- Test control inputs on keyboard and mouse (PC)
- Test control inputs on onscreen controller (Mobile)

Gameplay

- On load, user should be at the expected position. If new game is selected the user should be on level one. If loading a game, the user should be set to that particular location
- Ensure that player cannot go beyond the boundary of the level (if they immediately turn left upon loading)
- Ensure if beginning a new game, the onscreen text displaying the controls appears
- Ensure that difficulty of the game increases with each completed level (more enemies, difficult jumps etc)
- Player should be able to damage enemies which will kill them at different rates depending on difficulty
- If the player takes damage from an enemy, they should take damage based on enemy difficulty – this should correspond with the player health bar in the upper left corner
- Ensure that the damaged player can collect health pickups which is reflected in the health bar
- If the player's health bar is depleted a game over screen should appear
- Ensure that boss fight is triggered at the end of each level
- The end level boss should have a health bar in the upper right corner that functions like the player's health bar outlined above
- Once the boss is defeated, the player should be brought to the next level
- On final level completion the player should be brought to a screen with options to restart the game or exit

4.0 Testing Strategy

This section of the document will explain how we will approach testing, the strategies involved, and who is responsible for each test.

Unit Testing

Definition:

Unit testing is where individual components of any software are tested. The purpose of unit testing is to validate that each unit works as designed. Each test case should be tested independently to isolate issues that may arise. We will be using solitary tests here as it ensures that every unit is working as expected and isn't dependent on another unit to function.

Participants:

Jim Jones & Luke Freeman

Methodology:

While we would encourage the development team to use a test-driven development approach, we will produce some unit tests to ensure that the final product is as reliable as possible. Jim and Luke will be doing manual testing here. They will be assigned a platform each (PC or Mobile) which will allow them to test the controls for the respective platforms. Test cases will be created where the expected result is written, the tester then performs the inputs and finally the actual result is recorded. Individual test cases will be created for menu functions, gameplay, controls, level progression and the UI.

System & Integration Testing

Definition:

Integration testing tests the interface between modules of the application. The different modules are first tested individually and then combined to make a system.

System testing is testing of the software application as a whole to check if the system is compliant with the user requirements. System testing is a type of black box testing thus knowledge of the internal code is not required. It is a high-level testing that is always performed after integration testing.

Participants:

Jenny Blackman & Sam Uther

Methodology:

We will be using an incremental approach for this section. It makes fault localisation easier than if we were to use the big bang approach. Specifically, we will be using a bottom-up approach. Here the lowest level modules are brought together to form clusters. A driver is made to get the input and output of the test case. The cluster is then tested and once that testing is complete, the driver is removed so the cluster can be combined with the upper level. Jenny and Sam will continue this testing until the whole system is tested.

Performance & Stress Testing

Definition:

Performance testing checks the speed, response time, reliability, resource usage, scalability of a software program under their expected workload. The purpose of Performance Testing is not to find functional defects but to eliminate performance bottlenecks in the software.

Stress testing verifies the stability and reliability of the system. It is used to measure the system on its robustness and error handling capabilities under extremely heavy load conditions. Stress testing tests beyond the normal operating point and evaluates how the system works under extreme conditions.

Participants:

Jim Jones & Leslie Kilmer

Methodology:

Leslie will be performance testing the game by checking the response time whenever a certain action is called such as the delay between pressing the attack button and when the player character does the action. This performance testing will be done for all similar player actions and for level load times, menu navigation times etc.

Jim will stress test the system by testing different scenarios that are designed to use a lot of resources. Stress testing will entail spawning in many 2D objects and effects to see how far the game can be pushed until the system reaches an unacceptable loss of performance. They will also try to emulate real world scenarios where many applications are open at the same time once the game is running to see how the game copes with this.

Different types of hardware will be used for this section. Participants will test the game on hardware ranging from high end to low end to see how gameplay and performance is affected.

User Acceptance Testing

Definition:

User Acceptance Testing is a type of testing performed by the end user or client to verify the software system before moving the software application to the production environment. UAT is carried out in a separate testing environment with a production like set up.

Participants:

Bill Buxby & Ellen Shortall

Methodology:

Bill and Ellen will be given different platforms and test the game from start to finish. They will record any issues that appear which will be communicated to the development team.

Automated Regression Testing

Regression testing is a type of software testing to confirm that a recent program or code change has not affected existing features. In general, regression testing is a full or partial execution of already completed test cases to ensue existing functionalities are done.

Participants:

Luke Freeman

Methodology:

Luke will be called upon when a major update to the system occurs. He will make use of the test cases in vTest that have been executed previously and run them again, automating the process where appropriate. This section will be run automatically to allow the testing team to focus on other aspects of the testing.

Beta Testing

Definition:

Beta testing is a type of Acceptance testing, which adds value to the product as the intended user validate the product for functionality, usability, reliability, and compatibility. The beta testers are given a beta version of the product and use it on their own systems and on their own time. This type of testing relies primarily on feedback from the testers which will be communicated to the development team.

Participants:

Beta testers will be selected by Jim Gavin who will monitor all feedback

Methodology:

Jim will select beta testers from online forums who will be given a beta version to test on their own time. These users will be given a feedback link that they can use to send information on issues to Jim. At the end of the beta period Jim will gather the results and share them with the development team.

5.0 Test Schedule

Week 1

Task Name	Task Description	Start Date	End Date	Estimated Effort (Hrs/Days)	Participants
Unit Testing	Installation on Platforms & Menu Navigation	04/05/2020	04/05/2020	2 Hrs	Jim Jones, Luke Freeman
Unit Testing	Audio levels and changes	04/05/2020	04/05/2020	1 Hr	Jim Jones, Luke Freeman
Unit Testing	Character Movement/ Enemy AI/ Level Progression	04/05/2020	07/05/2020	2.5 Days	Jim Jones, Luke Freeman
Unit Testing	Run through of all components again	08/05/2020	08/05/2020	6 Hrs	Jim Jones, Luke Freeman

Weeks 2 & 3

Task Name	Task Description	Start Date	End Date	Estimated Effort (Hrs/Days)	Participants
System Integration Testing	-	11/05/2020	12/05/2020	1 day	Jenny Blackman & Sam Uther
Performance & Stress Testing	-	12/05/2020	13/05/2020	1 day	Jim Jones & Leslie Kilmer

User Acceptance Testing	-	13/05/2020	14/05/2020	1 day	Bill Buxby & Ellen Shortall
Automated Regression Testing	-	14/05/2020	No fixed date	-	Luke Freeman
Beta Testing	-	15/05/2020	22/05/2020	7 days	Jim Gavin & Beta Testers

6.0 Control Procedures

Problem Reporting

Software problems are expected, and it is crucial to the overall development that these are documented and communicated to the development team. Each tester will be given a issue tracker form (see appendix). These forms are to be used for the duration of that particular testing phase. Each tester must fill out the specific details of the issue as well as a priority level. The forms will be submitted each day and a weekly report will be written up. The team lead will then send the report to the development team along with the forms.

Change Requests

Change requests should be brought to the project manager who will then need to sign off on it before being brought to the development team or client.

7.0 Features to Be Tested

- Front End
- In-Game menus incl. play, sound, load, save, pause
- Player movement
- Player progression
- Player Health
- Enemy Movement
- Enemy Health
- Boss Spawn
- Boss Health
- Installation on both PC and Mobile
- Game boundaries

8.0 Features Not to Be Tested

All Features of the game will be tested

9.0 Resources/Roles & Responsibilities

Name	Role	Responsibilities
Michael Whelan	Project Lead	• Overseeing all testing teams • Create weekly reports • Communication between testers and developers/client
Jim Jones	Unit Tester / Stress Tester	• Unit Testing • Overloading the system to test system
Luke Freeman	Unit Tester / Regression Tester	• Unit Testing • Create Automated Regression tests • Manual Regression tests if necessary
Jenny Blackman	System Tester	• Create tests for system integration
Sam Uther	Integration Tester	• Create tests for system integration
Bill Buxby	UA Tester	• Use the product as a user would • Report issues
Ellen Shortall	UA Tester	• Use the product as a user would • Report Issues
Jim Gavin	Beta Team liaison	• Select individuals for beta test • Remain in contact with individuals • Obtain feedback about product • Compile feedback and report to project lead

Leslie Kilmer	Performance	• See how product performs under usual load • Report issues to lead

10.0 Schedules

Deliverable	Expected
Test Plan	01/05/2020
Test Cases	To be completed before each phase
Test Incident Reports	To be completed end of each day of testing
Test Summary Reports	To be completed end of each week

11.0 Risks

Risk Description	Probability	Effect on Cost/Schedule/Quality	Contingency
Tight time limits that influence testing flow	Medium	Cost, Schedule, Quality	- Add more testers if needed -Ensure all testers are working efficiently
Data Corruption/Loss	Medium	Cost, Schedule	- Ensure backups are made regularly on local and cloud platforms
Staff absences	Low	Cost, Schedule	- If staff are absent, substitute other members onto that team to ensure test flow is kept
Hardware Crash	High	Cost, Schedule	-Ensure that each system is

			operational before use -Have spare hardware on hand in case needed

12.0 Tools

Hardware	PC & Mobile devices	Different specifications of these hardware's will be used to test the system as much as possible
Automation Tools	Tosca TestSuite, Selenium	Plan and design test cases
Bug Tracking	Bugzilla/JIRA	Simple to use system, also contains Incident management tools if needed

Appendix

Issue Tracker form

Test Phase	**Action Description**	**Date Reported**	**Priority (L/M/H)**	**Tester**	**Tools used**

Signature by team lead:

Date: