**Test Plan**

Pixel Wizard

**Prepared by:**

Noel Melia

2/5/20

Contents

[1.0 INTRODUCTION 2](#_Toc40225224)

[2.0 OBJECTIVES AND TASKS 3](#_Toc40225225)

[2.1 Objectives 3](#_Toc40225226)

[2.2 Tasks 3](#_Toc40225227)

[3.0 SCOPE 4](#_Toc40225228)

[General 4](#_Toc40225229)

[Tactics 4](#_Toc40225230)

[4.0 TESTING STRATEGY 5](#_Toc40225231)

[4.1 Unit Testing 5](#_Toc40225232)

[4.2 System and Integration Testing 6](#_Toc40225233)

[4.3 Performance and Stress Testing 7](#_Toc40225234)

[4.4 User Acceptance Testing 7](#_Toc40225235)

[4.5 Batch Testing 7](#_Toc40225236)

[4.6 Automated Regression Testing 7](#_Toc40225237)

[4.7 Beta Testing: 8](#_Toc40225238)

[5.0 TEST SCHEDULE 8](#_Toc40225239)

[7.0 FEATURES TO BE TESTED 9](#_Toc40225240)

[8.0 FEATURES NOT TO BE TESTED 9](#_Toc40225241)

[9.0 RESOURCES/ROLES & RESPONSIBILITIES 9](#_Toc40225242)

[10.0 SCHEDULES 10](#_Toc40225243)

[11.0 RISKS/ASSUMPTIONS 10](#_Toc40225244)

[12.0 TOOLS 11](#_Toc40225245)

# 1.0 INTRODUCTION

This test plan describes the strategy that will be carried out to verify that the product/game is working properly with minor errors from the design specification and other requirements. The document will assist the tester and staff to the testing that has been carried out. The game will be mainly tested on the Computer and the controls for the game will be used in this.

This is a Horizontal Shooter game that is one player. It has many levels with each level getting harder as the player moves through each level through the enemy health increasing and more enemies to kill in each level.

# 2.0 OBJECTIVES AND TASKS

# 2.1 Objectives

The objective of this test plan is to find out as many game defects as possible and get the game finished and ready to be created. These will be the main objects that we will test throughout the process. The Main Components are as follows:

1. Front End
2. In Game Menus
3. Control Mechanisms
4. The Game

These are only but a few of the main components. In the following Tests the team will testing these in detail and check to see if there are many defects or errors in the game.

The testing of this game will be white box testing as the team will be testing the internal structure and workings of the game. White Box testing is like the work of a mechanic who examines the engine to see why the car is not working.

Some of the Tests that will be carried out are Unit Testing, System and Integration Testing, User Acceptance Testing. The team will be broken into these different sections and be tasked to accomplish it within the time frame given.

# 2.2 Tasks

Most of the Tasks that need to be tested will be before the game will be released. The game has many scripts and the tests will verify that the functionality of the game is running properly and that there is no errors or bugs in the criteria of the components. These will be severity defects for the some of this testing which the teams will decide the level of defect throughout the test case.

These are the Tests that will be carried out during the test case:

Unit Testing will be tasked to 4 Individuals.

System and Integration Testing will be Tasked to 4 individuals.

User Acceptance Testing will be tasked to the manager.

# 3.0 SCOPE

# General

Here I will describe the tests that will be carried out throughout the game. Each Heading will have a test for every relevant topic under it.

1. Front End
   1. Opening the Game
   2. Menu Opens up properly
   3. Play Button is working
   4. Settings Button
   5. Quit the game
   6. Load the game
   7. Delete the game Saved
2. In Game Menus
   1. Save the game
   2. Pause the Game
   3. Settings can be adjusted
   4. Exit the game
3. Control Mechanisms
   1. Controls are connected to what they are said to do
   2. Player Moves Easily
4. The Game
   1. Enemies die when hit with bullets
   2. Player dies when hit with projectiles
   3. Menu Logo Pauses the game Easy to follow through levels
   4. Pickup items can be picked up

As stated above the various tests will be carried out checked to make sure of no bugs or errors.

# Tactics

Many Parts will be broken down into smaller parts so it will make the testing quicker and more efficient. Some of the parts are duplicated, so it make less work on testing purposes.

1. Buttons on Main Menu
   1. For this to be tested, the Tester will have to Start the game and check that the buttons are linked correctly to the corresponding fields, such as Play Button will bring the player to Level 1 of the game.
   2. Settings Button will be brought to adjust the game settings like the volume and music levels.
2. Controls of the Player
   1. The game begins and the player will be presented with a pop-up menu showing the player the controls in how to control and move the player. The player if using the PC/Mobile Device, will then press the corresponding keys to see if the game will respond to the directory.
3. Levels Linked to Each Other
   1. The player will have to play the game and finish the first level by defeating all the enemies and getting the through the levels with a small bit of health. Once completed the player will exit and be brought to the next level.
4. Player Dies and What Happens
   1. Player is hit by projectiles and killed, or damage taken,
5. Enemy Dies and What Happens
   1. Player kills the enemy and what exactly happens to the enemy. Does he explode or what exactly happens to him?

Throughout the Project I will go through in detail how the tests will be broken down and divided into different testing systems and scenarios.

# 4.0 TESTING STRATEGY

Test Strategy presents the recommended approach to the testing of the game. Here describe how the tests will be carried out. The main considerations for the test strategy are the techniques to be used and the criterion for knowing when to testing is completed. Each Test will be tested individual and will take a certain amount of time to complete. In here the Test Schedule will be issued for each test. The game is created on Unity Platform and the code is written in C# Language The game.

In each section the plan has stated the definition, who will be assigned the tasks and what they will do in the testing.

# 4.1 Unit Testing

**Definition:**

The most 'micro' scale of testing; to test functions or code modules. It requires detailed knowledge of the internal program design and code. Not always easily done unless the application has a well-designed architecture with tight code: may require developing test driver modules or test harnesses.

**Participants:**

Joe Byrne, Jade Smith, Susan Newton, Paddy Flynn

**Methodology:**

Here we will test the way the player moves and the movement of enemies through there scripts that have been written. The test will be carried out among test a team of five people and each will be given a relevant part of the testing. The movement of the player will be set up in C# Language and will be given the direction movement that will appear on the screen before the player begins the level.

The team will begin by looking through the code of the player which will have a movement script attached to the asset of the player. This will be done by using the unity platform and using the unit test software that has come with Unity to assist the run of a smooth game without bugs or defects here. After they have looked to see if there are any errors in the code, they will start up the game.

In this test which will be set up by the team, there will be a pass and fail result. Below I will set up a step by step.

1. Tester has made a Unity Test in Unity within the Game.
2. Run the game.
3. Select the Play Button on Main Menu.
4. The Level will be presented with a Pop-Up Message showing the Instructions of the game and Controls.
5. Tester will Test the game by pressing left key to see if the game reacts to the key being pressed.
6. Success

This is only but one test that the team will produce. They will test the functions, classes and methods of the scripts and begin to see if it successful. The player movement is reasonable and realistic, so that the player of the game will be satisfied and produce expected results and actual results. The assert keyword will be used in the code to create a test case to compare if true or false.

# 4.2 System and Integration Testing

**Definition:**

System Testing is a level where the application is tested for its compliance to functional and non-functional requirements. Whereas, in integration testing, we test the interfacing between the modules which are interconnected with each other

**Participants:**

Niall Byrne, Noel Melia, Darragh Lally, Anthony Moore

**Methodology:**

There are two different approaches that will be taking and separate at the beginning and come together towards the end. The team will be broken into 2 groups to get the work done quicker and more stable. For the integration testing, the team will be using Bottom up testing which will integrate individual components in the levels until the complete system is created.

System testing will be investigating both functional and non-functional requirements of the code. To do this the team will be given code to test and perform checks on to make sure that program is running correctly with no bugs.

The test will be like the unit test above which each team will be giving section to as assess and test in detail. The test will be and what happens to player when he moves into an object. Below I will describe the procedure of the function that will be taken:

As stated, like the unit test, the team will test the behaviour of what will happen to the player when he moves into an obstacle or object. The object will have a script attached to it and the player will have a trigger activate when it moves into the object. Like before I will describe below:

1. Tester has looked at code within the Game.
2. Run the game.
3. Select the Play Button on Main Menu.
4. The Level will be presented with a Pop-Up Message showing the Instructions of the game and Controls.
5. Tester will Test the game by pressing left key to see if the game reacts to the key being pressed.
6. A box will be in front of player and the box will not move.
7. Success

This only but one test that the team have carried out. Then the System test will begin when the whole team will work together to test the entire system to see if the function is in all the objects.

After each test is complete it is added to the test case success or failure.

# 4.3 Performance and Stress Testing

**Definition:**

Used to describe such tests as system functional testing while under unusually heavy loads, heavy repetition of certain actions or inputs, input of large numerical values, large complex queries to a database system. So, in general it is the speed of the game.

**Participants:**

Joe Byrne

**Methodology:**

The tester will issue a performance check before running the game to check the speed of everything. The game then will begin, and a timer will be set. The Tester will use the asset in unity called NeoLoad which will be performed before the game starts.

Once complete then the check will be added to the test case and recorded for future checks. The only thing is once the more information and asset are added to the game the slower the performance will be.

# 4.4 User Acceptance Testing

**Definition:**

The purpose of acceptance test is to confirm that the system is ready for operational use. During acceptance test, end-users (customers) of the system compare the system to its initial requirements.

**Participants:**

* Michael Mulholland
* Public Testers

**Methodology:**

This is the final testing Section to determine if the requirements of the game are meet. This usually performed by the Project Manager of the Team and Testers that are brought in to do external work for testing. It is also to check if the Game will be user friendly for the customer. Its main function is to test the overall game to check if it is worthy of putting it on the shelf or not.

The test is done by playing the game and seethe games usability and user experience of the user.

# 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 base state to next case.

# Automated Regression Testing

**Definition:**

Regression testing is the selective retesting of a system or component to verify that modifications have not caused unintended effects and that the system or component still works as specified in the requirements.

**Participants:**

* Jade Smith, Susan Newton

**Methodology:**

The software that the team are using is [Game Driver](https://www.gamedriver.io/). This software is an external testing software that writes scripts. This Game Driver API provides tools for simulating input, moving, or querying object properties and works against the game engine’s native objects, giving unprecedented external control and access to the game inner workings.

**Running the tests in the game:**

Test assets are complied, and the game is started. Game Drivers technology hooks itself onto in game objects, giving the tests script complete control over the game state. Tests can be run against the ISE, a standalone build or as a continuous integration pipeline.

**Results:**

Once completed testers and developers can view whether each test has passed or failed and determine how to resolve any regressions. Unlike with manual testing, Game Driver test are constituent and repeatable, significantly reducing user error from testing and shrinking defect resolution cycles.

# 4.7 Beta Testing:

**Methodology:** testing when development and testing are essentially completed, and final bugs and problems need to be found before final release. Typically done by end-users or others, not by programmers or testers. Over time the game will be made and released to the public. It see the future is impossible but a prediction can be made and new versions of the game can be updated and releases when there is an outdated error that occurred.

# 5.0 TEST SCHEDULE

In this section each item will be given a time frame of how long it will take and be given. Additional time will be given to relate in case problem arise across the testing. As well as time schedule, there will be test cases updated every time there is a new one passed or failed. There are several stages that have been taken in the above report.

|  |  |  |
| --- | --- | --- |
| **Stage** | **Time Period** | **Resources** |
| Unit Testing | 2 – 3 Weeks | Unity, Visual Studio Code, Assets from Unity |
| System Testing  Integration Testing | 4 weeks  2 weeks | Visual Studio Code, Unity, Asset from Unity, Helper |
| Performance and Stress Testing | 2 weeks | External Software NeoLoad, Unity, Visual Studio Code |
| Automated Regression | 2 weeks | [GameDriver](https://www.gamedriver.io/), Trilleon, Unity, Visual Studio Code |
| User Acceptance Testing | 4 - 5 weeks | Unity, Game Itself |

# 6.0 FEATURES TO BE TESTED

* Pc Platform
* Front End
  + Opening the Game
  + Menu Opens up properly
  + Play Button is working
  + Settings Button
  + Quit the game
  + Load the game
  + Delete the game Saved
* In Game Menu
  + Save the game
  + Pause the Game
  + Settings can be adjusted
  + Exit the game
* Control Mechanism
  + Controls are connected to what they are said to do
  + Player Moves Easily
* The Game
  + Enemies die when hit with bullets
  + Player dies when hit with projectiles
  + Menu Logo Pauses the game Easy to follow through levels
  + Pickup items can be picked up

# 7.0 FEATURES NOT TO BE TESTED

* Boss Level
* Different sounds for Game
* Mobile Platform

# 8.0 RESOURCES/ROLES & RESPONSIBILITIES

In this section will be shown the members of the team/resources and responsible

**Project Manager**

* Michael Mulholland

The Project Manager oversees recruiting, staff supervision, and staff training. The Project Manager will also be responsible for test budgeting and test planning and the cohesive integration of test and development activities. Any acquisition of hardware and software for test environment must also be approved by the Project Manager. The Project Manager is also required to coordinate meetings and keep track of the progress of the testing as well as ensuring that test-product documentation is complete.

**Lead Programmers**

* Joe Byrne, Jade Smith, Susan Newton, Paddy Flynn

The Lead Programmer oversees the technical aspect of leadership for the testing. The Lead Programmer will need to be able to verify the quality of the requirements, including testability, requirement definition, test design, test-script and test-data development, test automation, test-environment configuration; test-script configuration management, and test execution. The Lead Programmer will help train new testers to use existing test tools.

**Lead Designers**

* Niall Byrne, Noel Melia, Darragh Lally, Anthony Moore

The Lead Designer design and develop all testing scenarios and procedures. The Lead Designer will oversee training new testers the procedures for bug and status reporting. The Lead Designer will also need to be able to identify the best ways to leverage a test tool on the project and to review test reports.

**Programmer**

* Joe Byrne

Programmers will oversee maintaining test environment and creating automated scripts. Programmers will also be responsible for executing security, load, and performance stress test. Programmers will also oversee preparing test reports which will be reviewed by the Lead Programmer.

**Designers**

* Jade Smith, Susan Newton

Designers will execute automated test cases using test scripts designed by the programmers as well as manual tests. Designers will also need to prepare test reports which will be reviewed by the Lead Designer.

**Public Testers**

* Joe Brolley, Eamon McGrath, Paul Kilgarriff

Public Testers will be introduced to the project to provide external feedback from the consumers and usability testing.

Also, there is a Gantt Chart State Roles, Schedule, and Responsibilities

# 9.0 SCHEDULES

This stage states the deliverable document as shown below:

* Test Plan
* Test Cases
* Test Incident Reports
* Test Summary Reports
* Gantt Chart

# 10.0 RISKS/ASSUMPTIONS

Some of the Worst fears always with a Test Project are running out of Money or Delay in Time. As the Plan states above and in detail, there is a sufficient allocation of resources and schedule give to the test procedure.

There is always an extra allocation of 2 weeks in case of problems arising in the plan

Below I got

***Risk and Mitigation***

|  |  |  |
| --- | --- | --- |
| **Risk Description** | **Impact if Risk Occurs** | **Mitigating Action** |
| Data corruption / Loss | High | - Files are automatically backed up in our local storage every hour.  - Files are synced to online cloud storage every day. |
| System crash / hardware crash | High | - Files are backed up and stored in multiple systems, networked. |
| Hardware malfunction or breakdown | High | - Have spare hardware’s for temporary use while it gets repaired.  - If no spare, get it replaced and have it back to running immediately.  - Software’s installation files are kept for quick hardware restoration. |
| Task Delay | High | - Pinpoint problem origin and solve it immediately  - Require tester to sign task completion report form |
| Unexpected absence of staff | Medium | - Enlist a temporary substitute employee through specialist or freelance.  - If unable, director is to temporarily fill in the work.  - All staff are to weekly document their work in preparation for their substitutes. |
| Insufficient tester | Low | - Project leads are to temporarily fill in the work  - Hire 3rd party tester or freelance |

# 11.0 TOOLS

The Automation Tools that will be used are as Follows: [GameDriver](https://www.gamedriver.io/) and Trilleon which the link are on the text.

* Writing Test scripts.
* Development of Test suites.
* Execution of scripts.
* Create result reports.
* Identify any potential bug or performance issue

The Bug Tracking Tool that is used in the testing is an asset to unity which can be added to the package via download. The tool is called Trello Bug Tracker.

<https://forum.unity.com/threads/what-options-do-i-have-for-automation-and-unit-testing-in-unity.682720/>

<https://assetstore.unity.com/packages/tools/integration/trello-bug-tracker-pro-75613>

# 12.0 REFERENCES