
# Software Testing Project

Mark Reilly
Github

May 12, 2020

## Contents

# 1 Introduction

## 1.1 Document Purpose

This document is a high-level overview defining our testing strategy for Game Development Ltd new 2D side-scrolling plat-former.The objective of the game is for the player to navigate through progressively difficult levels.The test team will verify the quality of the product prior to release. The document also lists the different resources that are needed for a successful testing of the project.

## 1.2 High Level Functions:

1. Unit Testing
2. Integration Testing
3. Smoke Testing
4. Regression Testing
5. Sanity Testing
6. User Acceptance Testing

## 1.3 Document Audience

Business owner(s), stakeholders and project team members are the intended audience for this document. Stakeholders will be requested to provide feedback on the overall scope of the test effort and any omissions.

# 2 OBJECTIVES AND TASKS

## 2.1 Objectives

The objective of our test plan is to find and report as many bugs as possible to improve the integrity of our product. We will use a broad range of tests to achieve our goals. The test plan document supports the following objectives:

1. Identify existing project information and the software that should be tested.
2. List the recommended test requirements (high level).
3. Recommend and describe the testing strategies to be employed.
4. Identify the required resources and provide an estimate of the test efforts.
5. List the deliverable elements of the test activities.

## 2.2 Tasks

1. Identify what particular tests will be used to test each module.
2. Identify the expected results for each test.
3. Perform the tests.
4. Document the test data, test cases and test configuration used during the testing process.
5. Successful unit testing is required before it is eligible for system and integration testing.
6. Unsuccessful testing will require a Bug Report to be generated.
7. Test documents and reports should be submitted when all is complete.

# 3 Scope

### 3.0.1 General

The main components of the game that need to be tested are as follows:

1. Main Menu:

   The Main Menu will have three options on start-up: 'Play', 'Settings', and 'Exit Game'. Selecting 'Play' will take the player into the game and the player will begin at Level 1. If a save system is able to be implemented, the player will begin at their last saved point. 'Settings' will allow the player to edit game settings, such as sound level and music level. 'Exit Game' will quit the application'.

2. Player Movement:

   Player Movement will be tested to see if the player is able to move the character, also to see if the player is confined to the screen and not able to go out of view and disappear.The players jump, crouch and attack abilities will be tested under this section.

3. Player Health Counter:

   The players health counter will be tested to see if the player loses a life after being hit by an enemy. When the player collects an extra health, the health counter should go up by one.If the player loses all of its lives, it should be destroyed.

4. Enemy Health Counter:

   The enemies health counter will be tested to see if the enemy loses a life after being hit by the players bullet.When the enemy loses all its lives, the enemy should be destroyed. If the enemy is the boss enemy at the end of each level and it is destroyed , the player should be brought to the next level.

5. In-Game Menu:

   The game will include a number of options when the game is paused, similar to those available on start up. The player will be able to resume the game, access settings, restart the level, and exit the game. All the in-game menu options will be tested to see if the selected options does its intended function.

### 3.0.2 Tactics:

| Scenario | Strategy to test scenarios |
|---|---|
| Main Menu | Functionality testing |
| Player Movement | Functionality, Integration testing |
| Player Health Counter | Functionality, Integration, System, regression testing |
| Enemy Health Counter | Functionality, Integration, System, regression testing |
| In-Game Menu | Functionality, Integration, System, regression, and automation testing |

# 4 Testing Strategy

## 4.1 Unit Testing

### 4.1.1 Definition

UNIT TESTING 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.[6]

### 4.1.2 Participants

Developers

### 4.1.3 Methodology

Unit test case has been processed by Developers and they will write the test scripts for the unit testing if game.[6]

| Unit test Scenario | Expected Outcome | Actual Outcome |
|---|---|---|
| Play game | User should able to play a game | User is able to play a game. |
| Pause Game | Pause button functionality should be work | Pause is working. |
| Health counter for Main player | Health should be increased for main player | Health is increasing. |
| Health Counter for Enemy Player | After attacking enemy player health should decrease | Health is impacted for enemy player |
| Setting options | All setting options should work | All setting options are working. |
| Stage complete validation | Once Main player complete the stage it should display one stage complete. | Stage is completed. |
| Next stage validation | Once completion of stage main player should be able to play in next stage | User move to next stage once complete previous stage. |
| Exit game validation | If exit button press, then user should be able to exit from game | Exit game working properly. |

Figure 1: Unit Testing

## 4.2 System and Integration Testing

### 4.2.1 Definition

System Integration Testing is defined as a type of software testing carried out in an integrated hardware and software environment to verify the behaviour of the complete system. It is testing conducted on a complete, integrated system to evaluate the system's compliance with its specified requirement. System Integration Testing (SIT) is performed to verify the interactions between the modules of a software system. It deals with the verification of the high and low-level software requirements specified in the Software Requirements Specification/Data and the Software Design Document.[5]

### 4.2.2 Participants

Testing is performed using test data created by the testers.

### 4.2.3 Methodology

Once unit testing is completed then as a tester; they should test whole game from start to end with all the components from game. Tester will perform the testing and will create the test scripts.

| Unit test Scenario | Expected Outcome | Actual Outcome |
|---|---|---|
| Play game | User should able to play a game | User is able to play a game. |
| Pause Game | Pause button functionality should be work | Pause is working. |
| Health counter for Main player | Health should be increased for main player | Health is increasing. |
| Health Counter for Enemy Player | After attacking enemy player health should decrease | Health is impacted for enemy player |
| Setting options | All setting options should work | All setting options are working. |
| Stage complete validation | Once Main player complete the stage it should display one stage complete. | Stage is completed. |
| Next stage validation | Once completion of stage main player should be able to play in next stage | User move to next stage once complete previous stage. |
| Exit game validation | If exit button press, then user should be able to exit from game | Exit game working properly. |

Figure 2: System and Integration Testing

## 4.3 Performance and Stress Testing

### 4.3.1 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 or device. STRESS TESTING is a type of Software Testing that verifies the stability reliability of the system. This test mainly measures the system on its robustness and error handling capabilities under extremely heavy load conditions.[4]

### 4.3.2 Participants

Testers

### 4.3.3 Methodology

For checking the game performance and at a time how many users can play a game we have to perform Stress and performance testing. Using Jmeter we can create n number of virtual users and check the performance of the game. Tester will create the test scripts.[4]

| Scenario | Expected Outcome | Actual Outcome |
|---|---|---|
| N number of users' validation at the same time | Game speed should not impact | Game is running smoothly. |
| Verifying response(movement, health counter, attack, play, pause) from different options | It should be a quick response | All response from different options are working |

Figure 3: Performance and Stress Testing

## 4.4 User Acceptance Testing

### 4.4.1 Definition

User Acceptance Testing (UAT) is a type of testing performed by the end user or the client to verify/accept the software system before moving the software application to the production environment. UAT is done in the final phase of testing after functional, integration and system testing is done.[1]

### 4.4.2 Participants

End User OR Client

### 4.4.3 Methodology

To ensure a successful product delivery, we should follow 5 phase UAT process i.e. Planning, Coverage, Execution  Tracking, Reporting, and Reuse. As this has been executed by End user or client so tester will create either script or it can be executed manually.

| Scenario | Expected Outcome | Actual Outcome |
|---|---|---|
| Verify the Main Menu | All options inside the main menu should work | All options are working |
| Validation of Player Movement | Player should move Forward/Backword | Player movement is working fine |
| Verify Health counter for both player | If attack to Enemy player then health should be decrease and for collect the health then increase | Health is working fine for both player |

Figure 4: User Acceptance Testing

## 4.5 Batch Testing

### 4.5.1 Definition

Group of tests executing sequentially one by one is called Batch Testing.[2]

### 4.5.2 Participants

Testers

### 4.5.3 Methodology

Batch testing can be conducted once we have all the test cases for the game. We will arrange all the test cases sequentially and will execute in a batch. For Example, if we have to test game like start  Pause  Resume then Exit. So we have to arrange all the test cases according to that and then we will execute all the test cases in Batch. Tester will create the scripts for batch execution.[2]

## 4.6 Automated Regression Testing

### 4.6.1 Definition

Regression tests ensure that code that was previously functioning correctly does not regress when changes are made. Having a comprehensive suite of unit level regression tests provides a safety net, making sure that code changes do not break existing functionality.[3]

#### 4.6.2 Participants

Tester and Developer

#### 4.6.3 Methodology

Regression testing ensures that recent changes to the code leave the rest of the code intact, thereby preventing software regression. For Example, In Game If Main Player attack Enemy Player then Health should be impacted for enemy player. If main player health is less and if he/she picked health, then life should be increased. So if we are testing these scenario then it should work properly.[3]

### 4.7 Beta Testing Participants

#### 4.7.1 Participants

End Users/Real Users

#### 4.7.2 Methodology

- All the components of the Game are ready to start this testing.
- Documentation that must reach the end users should be kept ready – Setup, Installation, Usage, Uninstallation should be detailed out and reviewed for correctness.
- Product Management team should review if each key functionality is in good working condition.
- Procedure to collect Bugs, feedback etc should be identified and reviewed to publish.

# 5 Testing Schedule

| Roles | Assign To | Start | End |
|---|---|---|---|
| Create Test Plan | Mark R. | <<Date>> | <<Date>> |
| Preparation of Test cases | | | |
| Unit Testing | | | |
| System and Integration Testing | | | |
| Performance and Stress Testing | | | |
| User Acceptance Testing | | | |
| Batch Testing | | | |
| Automated Regression Testing | | | |
| Defect Retesting | | | |
| Test Summary Report | | | |

Figure 5: Test Schedule

**Tools:**

| Tool Name | License Version | Valid from | Valid Till | Approver |
|---|---|---|---|---|
| HP ALM | 12.20 | <<Date>> | <<Date>> | <<Name>> |
| Appium | | | | |

Figure 6: Tools

**Facilities:**

| Facility Name | Version | Valid from | Valid Till |
|---|---|---|---|
| Monitor | <<VersionNo.>> | <<Date>> | <<Date>> |
| Processor | | | |
| Graphics Card | | | |

Figure 7: Facilities

# 6 Control Procedures

## 6.1 Problem Reporting

Any defect during game testing will be reported in below manner.

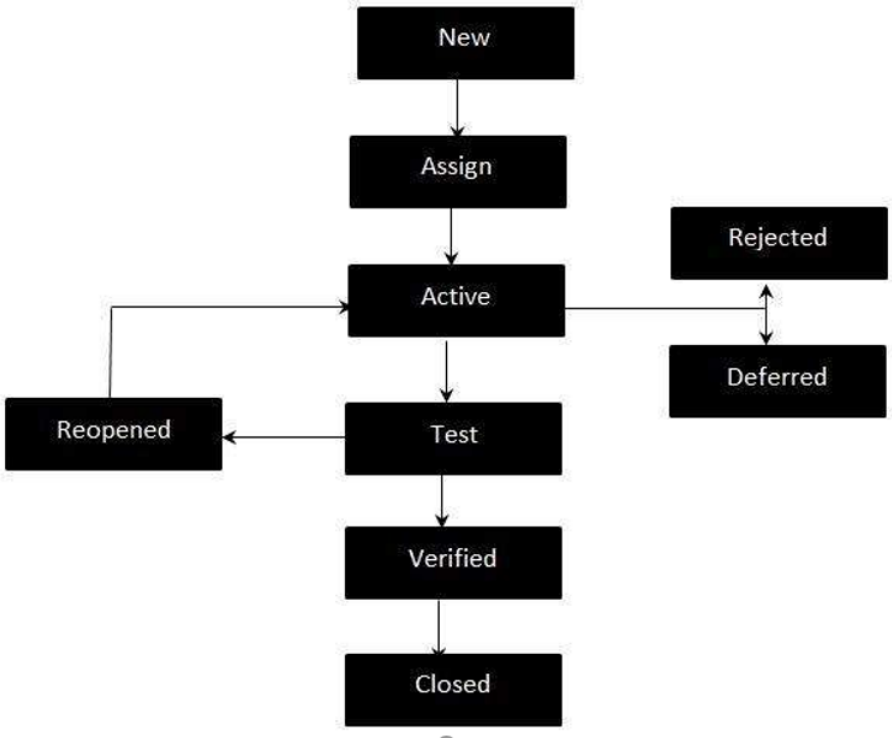


Figure 8: Control Procedure

## 6.2 Change Requests

Below is the steps for change requests.

| Steps | Actions |
|---|---|
| Change request identification | Identify the need for a change and describe it on the project change request form |
| Change request assessment | • If the change is not valid, it must be deferred or rejected<br>• Determine appropriate resources required to analyse the change request<br>• Perform quick assessment of the potential impact and update the change request form<br>• At this stage, rejected change request should stopped |
| Change request analysis | • For analysis assign the change request to an authorized member<br>• Deferred change re-enter this analysis step<br>• At this stage, rejected change request should stopped |
| Change request approval | • Identify change risk and complexity level before approval<br>• Identify the impact level of the change before approval<br>• Review impact of Change Request to authorized person for approval<br>• At this stage, rejected change request should stopped |
| Change request implementation | • Update project procedure and management plans<br>• Inform about the changes to the team<br>• Monitor progress of change request<br>• Record the completion of change request<br>• Close change request |

Figure 9: Change Requests

# 7 Features to be Tested

Below is the list of features which will be tested as part of this release:

- Background
- Character Moving forward and Backwards
- Character Jump/Attack
- Game Pause/Play
- Selection of Player/Enemy Character
- Player/Enemy Projectile
- Ground Asset
- Health Pickup
- Rock Asset
- Menu Options (Play, Load, Delete Games)
- Settings
- Exit Game

# 8 Features not to be Tested

- Hardware Performance
- Installation of any packages
- Network performance
- Server response
- Processors and memory constraints

# 9 Roles Responsibilities

- Game producers are responsible for setting testing deadlines in coordination with marketing and quality assurance.They also manage many items outside of game testing, relating to the overall production of a title. Their approval is typically required for final submission.
- Lead tester, test lead or QA lead is the person responsible for the game working correctly and managing bug lists. A lead tester manages the QA staff. Lead tester works closely with designers and programmers, especially towards the end of the project. The lead tester is responsible for tracking bug reports and managing that they are fixed. They are also responsible that QA teams produce formal and complete reports.
- Testers are responsible for checking that the game works, is easy to use, has actions that make sense, and contains fun game play. Testers need to write accurate and specific bug reports, and if possible providing descriptions of how the bug can be reproduced. Testers may be assigned to a single game during its entire production, or brought onto other projects as demanded by the department's schedule and specific needs.

| Roles | Assign To | Start | End |
|---|---|---|---|
| Create Test Plan | Mark R. | <<Date>> | <<Date>> |
| Test Plan Review and Rework | <<Name>> | | |
| Preparation of Test cases | | | |
| Unit Testing | | | |
| System and Integration Testing | | | |
| Performance and Stress Testing | | | |
| User Acceptance Testing | | | |
| Batch Testing | | | |
| Automated Regression Testing | | | |

Figure 10: Roles

# 10 Schedules

The following table highlights the tasks that will be carried out during this level of testing. These dates are accurate at the time of writing this document but can change as the project progresses.

| Test Task | Start | End |
|---|---|---|
| Release Kick off Meeting | <<Date>> | <<Date>> |
| Create Test Plan | | |
| Test Plan Review and Rework | | |
| Sign-off Test Plan | | |
| Preparation of Test Condition, Test Scripts for Changes | | |
| Review Test Condition, Test Scripts for Changes | | |
| Test Execution | | |
| Defect Retesting | | |
| Preparation for Test Summary report | | |
| GO-LIVE | | |

Figure 11: Schedule

# 11 Dependencies, Issues, Risks and Assumptions

## 11.1 Dependencies

The table below describes the dependencies related to system testing for this project:

| S.NO. | Dependencies |
|---|---|
| 1. | Game testing team shall have access to all relevant databases. |
| 2. | Server support is readily available for continuous execution during Game Testing to verify the connections between different servers. |
| 3. | Uninterrupted environment is available till closure of Test execution. |
| 4. | Availability of Test environment with all required packages installed and patches updated and ready for testing at the start of Game Testing. |

Figure 12: Dependencies

## 11.2 Issues:

The table below describes the open issues related to Application testing for this project:

| S.NO. | Issue Description | Owner/Assigned to |
|---|---|---|
| 1. | NA | |
| 2. | NA | |

Figure 13: Issues

## 11.3 Risks/Assumptions:

The table below describes the assumptions related to testing:

| S.NO. | Assumptions |
|---|---|
| 1. | Game is ready with all the required packages and the interface are set up, configured, and validated |
| 2. | Document has the required information for testing. |
| 3. | Any new enhancement identified as part of Game execution will need to be addressed via Change Management System. |

Figure 14: Assumptions

The table below describes the risks related to testing:

| S.NO. | Risks |
|---|---|
| 1. | The game does not create compelling experiences for the targeted audience. |
| 2. | The game does not have a player-centric design |
| 3. | Late submission of information, delays in document approval by the Customer |
| 4. | Game not unique, competitive, fast paced. |
| 5. | The fun factor and addictive gameplay missing in the games. |
| 6. | Game development cost goes over budget |
| 7. | The game should have simple aesthetic design and the gameplay. |
| 8. | The game fails because of technical issues, broken features, critical bugs, bad music sound, and poor video. |

Figure 15: Risks

# 12 Tools

Defect Management Tool – HP ALM (Application Lifecycle Management), JIRA, BUGZILLA Automation Tool – APPIUM, MAuto

# References


[1] *Acceptance Testing.* http://softwaretestingfundamentals.com/acceptance-testing/.

[2] *Batch Testing.* https://www.slightbook.com/Subjects/Software-Testing/Automation-QTP/Interview-Questions/What-is-Batch-testing.

[3] *Regression Testing.* https://www.guru99.com/regression-testing.html.

[4] *Stress Testing.* https://www.guru99.com/stress-testing-tutorial.html.

[5] *System Integration Testing.* https://www.guru99.com/system-integration-testing.html.

[6] *Unit Testing.* http://softwaretestingfundamentals.com/unit-testing/.