THE PIXEL WIZARD TEST PLAN

Prepared By:

Fabricio Gentile

01/05/2020

TABLE OF CONTENTS

# **Introduction**

The product is a pixeled 2D side scrolling platform game. At first divided into three levels with a different range of enemies and bosses. To kill the enemies, the player can attack shooting a ball of flame from using the left mouse click. In the game scene, the player’s health bar and boss’s health bar will be represented by diamonds (red for the player and blue for the boss), the background will be used repeatedly, the player will be presented with some health pickups to increase his health.

The game 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 can 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’. OPTIONAL: Include a ‘Load Save’, ‘Save Game’, and ‘Delete Save’ option. This will allow the player to create multiple save files and be able to choose which one to load.

# **Objectives and Tasks**

## Test Objectives

The objective of testing is to assure that the system meets the full requirements, fit metrics for each quality requirement and satisfies the use case scenarios and maintain the quality of the product. The software test will ensure that our software is free from high and medium severity defects and all the game logic will function as intended. For example: After killing the boss, the player should go to the next level; Picking up the “health pickup”, the health of the player should increase.

## Tasks

* Tests
* Problem Reporting
* Change Requests
* Test Report

# **Scope**

## General

The scope of this document is testing the functions of the game, to ensure that all functionalities has been implemented correctly:

* The menu and the menu functions
* art (character model, platform, texture, objects, enemies, etc.)
* animation (when the player/ enemy is attacking)
* music, audio, and the sound effect
* game logic and flow
* increasing levels of difficulty
* the scoring
* the game options (game start or menu selection, game pause, pause menu options)
* health bar and health pickup

## Tactics

The test team will meet once every week to evaluate progress to date and to identify error trends and problems as early as possible. The test team leader will meet with development and the project manager once every week as well. Additional meetings can be called as required for emergency situations.

# **Testing Strategy**

This test plan relies on test automation, supplemented by test developers. Developers are expected to have run the same tests locally that will be run for them by continuous integration. This will execute unit tests, System and integration tests, Performance and stress tests, User acceptance tests, batch tests, automated regression tests and beta tests.

## Unit Testing

**Definition:** Unit Test is basically the test of the smallest testable part of the code. If one part of the game is buggy/broken, it might result in other areas of the game to fail.

**Participants:** John; Terry; Bobby.

**Methodology:** It will be done by the developer and will be approved by the development team leader. Proof of unit testing (test case list, sample output) must be provided by the programmer to the team leader before unit testing will be accepted and passed on to the test person.

## System and Integration Testing

**Definition:** System and Integration testing aim to find integration failures between the units, it consists of carrying out various types of tests that aim to determine whether the components of the game integrate well and perform the functionalities that have been specified to them.

**Participants:** Monica; Tim.

**Methodology:** It will be performed by the development team leader with assistance from the individual developers as required. Programs will enter System/Integration test after all critical defects have been corrected.

## Performance and Stress Testing

**Definition:** Performance and Stress testing is a form of deliberately intense or thorough testing used to determine the stability of a given system or entity. It involves testing beyond normal operational capacity, often to a breaking point, to observe the results.

**Participants:**Patrick; Jordan.

**Methodology:**It will be performed by the development team. The loading time is going to be tested and concurrent functions will be put into test to check the performance of the game.

## User Acceptance Testing

**Definition:** User Acceptance testing is to conform that system is developed according to the specified user requirements and is ready for operational use.

**Participants:** Ruby; Tessa.

**Methodology:** Testing will be performed by the actual end users with the assistance of the test manager and development team leader. The acceptance test will be done after completion of the System and Integration test process.

## Automated Regression Testing

**Definition:** Regression testing is a software testing technique that consists of applying the most recent versions of the software, to ensure that no new defects have appeared in components already analysed.

**Participants:** John; Bobby.

**Methodology:** Regression for fixed bugs will be done by respective the testers once it is resolved by respective developer and bug/defect status will be updated accordingly. The team will perform a comprehensive regression testing exercise during System Testing.

## Beta Testing

**Definition:** Beta testing is a product in its beta version is for users to use it for the purpose of reporting bugs and feedback to the developers and the company that develops it.

**Participants:** Trevor

**Methodology:** The beta test is implemented by users, usually with little or no management by the development organization (or another non-end user). Beta testing is the most subjective of all acceptance testing strategies.

# **Test Schedule**

|  |  |  |  |
| --- | --- | --- | --- |
| **Task Name** | **Start** | **Finish** | **Effort** |
| Test Planning | 05/05 | 09/05 | 40 man-hours |
| Review Requirements documents | 07/05 | 09/05 | 24 man-hours |
| Unit test | 10/05 | 11/05 | 16 man-hours |
| System and Integration test | 11/05 | 12/05 | 16 man-hours |
| Performance and Stress test | 12/05 | 13/05 | 16 man-hours |
| User Acceptance Test | 13/05 | 14/05 | 16 man-hours |
| Bach test | 14/05 | 15/05 | 16 man-hours |
| Automated Regression test | 15/05 | 16/05 | 16 man-hours |
| Beta test | 16/05 | 17/05 | 16 man-hours |
| Problem Report | 18/05 | 19/05 | 13 man-hours |
| Change requests | 19/05 | 20/05 | 13 man-hours |
| Automated Regression test | 21/05 | 22/05 | 16 man-hours |
| Beta test | 22/05 | 23/05 | 16 man-hours |

# **Control Procedures**

See Appendix 1 and Appendix 2

# **Features to be tested**

|  |  |
| --- | --- |
| **Music/ Sounds** | ON/OFF sound & background music |
| Verify if sound effects are in sync with action |
| **User Interface** | Check for animation, movement of character, graphics etc |
| Test whether one object overlaps with another |
| Verify if loading indicator is displayed wherever required |
| Character should not move out of the screen/specified area |
| Character should not move out of the screen/specified area |
| Check for screen title |
| Font displayed (color, size etc) |
| Check other objects (platform, pickup items, tree on background, etc) |
| **Performance** | Check the loading time of a game (starting the game and changing levels) |
| **Score** | Score calculation |
| Check for level completion syncs with the score |
| **Multitasking** | Switch between different apps and play game, check for sound, score, UI |
| **Pause** | Check if game is paused when multitasking or spacebar is pressed |
| **Functionality** | Check game area, game logic |
| Play till last level |
| Check for bonus score |
| Check the score hike when level gets increased |

# **Features not to be tested**

|  |  |
| --- | --- |
| **Share Options** | Post score via mail/FB/Twitter |

**Reason:** Not to be included in this release of the Software.

9.0

|  |  |  |
| --- | --- | --- |
| Role | Staff Member | Responsabilities |
| Project Manager | Joe | In charge of recruiting, staff supervision, and staff training.  Responsible for test budgeting and test planning and the cohesive integration of test and development activities.  Coordinate meetings and keep track of the progress of the testing as well as ensuring that test-product documentation is complete. |
| Lead Designer / Team | Smith | Develop all testing scenarios and procedures.  In charge of training new testers, the procedures for bug and status reporting.  Be able to identify the best ways to leverage a test tool on the project and to review test reports. |
| Designers | Patrick  Jordan  Carol  Pheobe | Execute automated test cases using test scripts designed by the programmers as well as manual tests.  Prepare test reports which will be reviewed by the Lead Designer. |
| Lead Programmer | Tim | In charge of the technical aspect of leadership for the testing.  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. |
| Programmers | John  Terry  Monica  Bobby | Maintaining test environment and creating automated scripts.  Responsible for executing security, load and performance stress test.  Preparing test reports which will be reviewed by the Lead Programmer. |
| Public Testers | Ruby  Tessa  Trevor | Public Testers will be introduced to the project to provide external feedback from the consumers and usability testing. |

10.0

|  |  |  |
| --- | --- | --- |
| Deliverable Documents | Start | Finish |
| Test Plan | 05/05 | 09/05 |
| Test Cases | 10/05 | 17/05 |
| Test Incident Reports | 18/05 | 19/05 |
| Test Summary Reports | 19/05 | 20/05 |

11.0

*Risk and Mitigation*

|  |  |  |
| --- | --- | --- |
| 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 are 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 |
| Insufficient skilled tester | Low | - Make sure testers go through tester training briefing |

12.0 TOOLS

The only test tools to be used are the standard AS/400 provided utilities and commands.

A. The Program Development Manager (PDM) will be used as the source version configuration management tool in conjunction with the in-house check-in/check-out control utility. The check-in/out utility is part of each developers standard AS/400 access menu.

B. The initial prototypes for the new screens will be developed using the AS/400 Screen Design Aid (SDA). The initial layout and general content of the screens will be shown to the sales administration staff prior to proceeding with testing and development of the screens.

C. All editing, compiling and debugging will be done using the Source Entry Utility (SEU).

D. Data acquisition will be from actual production files where available using the AS/400 data base copy command CPYF and it's various functions. Additional data will be created and modified as needed using the Data File Utility (DFU). No changes will ever be made to actual production files under any circumstances.

E. Initial data for EDI testing will be done using one or two beta sites and replicating the data at the mailbox location or locally in the control files, to create high volume data and to simulate multiple distributors sending in data.

|  |  |
| --- | --- |
| Process | Tool |
| Bug tracking | Jira |
| Test case execution | **Trilleon** |
| Test reporting | PDF |
| Check list creating | Microsoft Excel |
|  |  |

Appendix 1

PROBLEM REPORT

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test ID** | **Test Name** | **Issue** | **Responsible** | **Date** | **Action Taken** | **Status** |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

Appendix 2

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Module** | **Request** | **How is now** | **How it will be** | **Reasons for change** | **Positive Impact** | **Negative Impact** |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |