The pixel wizard test plan

GMIT | Dublin Road, Galway

Software Testing project

DARRAGH LALLY – G00220290

2020

Table of Contents

[Introduction 2](#_Toc40028953)

[Objectives 2](#_Toc40028954)

[Tasks 2](#_Toc40028955)

[Scope 2](#_Toc40028956)

[General 2](#_Toc40028957)

[Areas of Testing 2](#_Toc40028958)

[Tactics 3](#_Toc40028959)

[Error Handling 4](#_Toc40028960)

[Testing Strategy 4](#_Toc40028961)

# Introduction

*‘Testing can only show the presence of bugs, never the absence.’*

This test plan is to outline a testing strategy for ‘The Pixel Wizard’, a 2D, cross platform game. As it is impossible to exhaustively test any project the aim here is to help expose any defects in the major components of the game and by doing so, determine if the product is ready for market. As the game is already produced, we will be taking a reactive testing approach.

# Objectives

* ~~Reactive testing because game is already produced~~
* Do risk bases analysis
* ~~Main obj is to identify any defect.~~
* As testing is independent, we reduce risk of bias
* M. O’Hammad
* Add testing environment ?
* Contingency plans – over time, budget,

The main objective is to identify any defects within the game. This will be accomplished by testing the main components. We have chosen four components to focus on, as highlighted in the design document, see below.

* In Game Menus
* Save Functionality
* Control Mechanisms
* Game Play

These four parts will be tested to ensure the game is in a playable state before release. Test cases will be drafted for each component to outline the tests to be conducted and their outcome logged.

If defects are found, they will be logged, categorized and the development team will be notified.

# Tasks

*List all tasks identified by this Test Plan, i.e., testing, post-testing, problem reporting, etc*.

There will be several tasks to be completed through the testing process, these include:

* Identifying components to be tested.
* Decide on a testing method appropriate for each component.
* Preform test under varying conditions.
* Logging test result.
* Categorizing errors. (Trivial to Blocker)
* Notifying development team with change request.
* Preform confirmation/regression test.

# Scope

## General

As mentioned above there are four areas where we will focus our testing. These components will be tested individually, where possible, and in conjunction with one another. The purpose is to test as much as possible to ensure the product is ready for consumption. This will include closed beta testing.

## Areas of Testing

1. **Individual functions:** Testing, where possible, each function on its own. Ensuring we test all possible code paths
2. **Combination of classes/functions:** Testing, where possible, classes and functions that interact with each other.
3. **Player Interfaces:** Testing all routes possible for a user to navigate on the in-game menus and game options. Testing game visuals and SFX.
4. **Game Play:** Testing the complete product to ensure all parts work together and do not cause any critical errors.
5. **Beta Testing:** When the testing team is satisfied with the in-house testing, we want to make available to a small public group to test the game in real world circumstances.

## Tactics

*List here how you will accomplish the items that you have listed in the "Scope" section. For example, if you have mentioned that you will be testing the existing interfaces, what would be the procedures you would follow to notify the key people to represent their respective areas, as well as allotting time in their schedule for assisting you in accomplishing your activity?*

We plan on having our testing (up to beta) complete in 6 days. We are allowing a further 6 days for beta testing. The team will be notified of their individual responsibilities at our initial meeting, these responsibilities will then be reiterated via email and issues will be opened on GitHub assigning each team member their tasks.

**Day 1:**

**Entry Criteria:** Test cases developed.

**Exit Criteria:** 100% completion of test cases, no major errors, 1-5 minor or low-level errors acceptable.

**Note:** Testing each function and any available code paths.

|  |  |  |
| --- | --- | --- |
| **Test** | **Member** | **Due** |
| Unit Testing | D. Lally | Day 2, @ 1300 |
|  | M. Hynes |  |
|  | J. Esus |  |

**Day 2 & 3:**

**Entry Criteria:** Unit Testing complete.

**Exit Criteria:** 100% completion of test cases, 1-3 low-level errors acceptable.

**Note:** Testing classes and functions working in tandem

|  |  |  |
| --- | --- | --- |
| **Test** | **Member** | **Due** |
| Integration Testing | D. Lally | Day 4, @ 0900 |
|  | M. Hynes |  |
|  | J. Esus |  |

**Day 4:**

**Entry Criteria:** Integration Testing complete.

**Exit Criteria:** 100% completion of test cases, 1-3 minor or low-level errors acceptable.

**Note:**

* Testing menus and in game options. Testing visuals i.e. aminations.
* Testing player movement/actions

|  |  |  |
| --- | --- | --- |
| **Test** | **Member** | **Due** |
| Interface Testing | D. Lally | Day 5, @ 0900 |
|  | M. Hynes |  |
| Avatar Testing | J. Esus | Day 5, @ 0900 |

**Day 5:**

**Entry Criteria:** Interface Testing complete.

**Exit Criteria:** 100% completion of test cases, 5 minor or low-level errors acceptable.

**Note:** Testing the game in its entirety

|  |  |  |
| --- | --- | --- |
| **Test** | **Member** | **Due** |
| Full Game Testing | D. Lally | Day 6, @ 0900 |
|  | M. Hynes |  |
|  | J. Esus |  |

**Day 6:**

Assuming there are no major bugs found, we will release the game into our beta test group. This group consists of volunteer community gamers. This period is important to the testing plan because we can simulate a large player base, and have the community try to break the game in ways that specified testing may not be able to pick up on. This period of testing is to last 6 days. During which the community will give feed back on any errors they may encounter.

## Error Handling

We will be working off a 5-level categorizing:

1. Blocker
2. Critical
3. Major
4. Minor
5. Trivial

All bugs must be categorized and logged in the centralised repository for the testing manager to review. After reviewing the manager will notify the development team and request changes to be made. When the bug is eliminated regression testing will be performed by the tester who highlight the issue in the first place. Once confirmation and regression testing has passed, we can move on to the next stage of the STLC.

# Testing Strategy

*Describe the overall approach to testing. For each major group of features or feature combinations, specify the approach which will ensure that these feature groups are adequately tested. Specify the major activities, techniques, and tools which are used to test the designated groups of features. The approach should be described in enough detail to permit identification of the major testing tasks and estimation of the time required to do each one*

**Unit Testing**

**Definition:**

Specify the minimum degree of comprehensiveness desired. Identify the techniques which will be used to judge the comprehensiveness of the testing effort (for example, determining which statements have been executed at least once). Specify any additional completion criteria (for example, error frequency). The techniques to be used to trace requirements should be specified.

**Participants:**

List the names of individuals/departments who would be responsible for Unit Testing.

**Methodology:**

Describe how unit testing will be conducted**, including a description of tests to be carried out**. Who will write the test scripts for the unit testing, what would be the sequence of events of Unit Testing and how will the testing activity take place?

**Entry & Exit Criteria:**

**4.2 System and Integration Testing**

**Definition:**

List what is your understanding of System and Integration Testing for your project.

**Participants:**

Who will be conducting System and Integration Testing on your project? List the individuals that will be responsible for this activity.

**Methodology:**

Describe how System & Integration testing will be conducted**, including a description of tests to be carried out** Who will write the test scripts for the unit testing, what would be sequence of events of System & Integration Testing, and how will the testing activity take place?

**4.3 Performance and Stress Testing**

**Definition:**

List what is your understanding of Stress Testing for your project.

**Participants:**

Who will be conducting Stress Testing on your project? List the individuals that will be responsible for this activity.

**Methodology:**

Describe how Performance & Stress testing will be conducted**, including a description of tests to be carried out** Who will write the test scripts for the testing, what would be sequence of events of Performance & Stress Testing, and how will the testing activity take place?

**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:**

Who will be responsible for User Acceptance Testing? List the individuals' names and responsibility.

**Methodology:**

Describe how the User Acceptance testing will be conducted**, including a description of tests to be carried out** Who will write the test scripts for the testing, what would be sequence of events of User Acceptance Testing, and how will the testing activity take place?

**4.5 Batch Testing**

**4.6 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:**

**Methodology:**

**4.7 Beta Testing Participants:**

**Methodology:**

**5.0 TEST SCHEDULE**

Include test milestones identified in the Software Project Schedule as well as all item transmittal events.

Define any additional test milestones needed. Estimate the time required to do each testing task. Specify the schedule for each testing task and test milestone. For each testing resource (that is, facilities, tools, and staff), specify its periods of use.

**6.0 CONTROL PROCEDURES**

**Problem Reporting**

Document the procedures to follow when an incident is encountered during the testing process. If a standard form is going to be used, attach a blank copy as an "Appendix" to the Test Plan. In the event you are using an automated incident logging system, write those procedures in this section.

**Change Requests**

Document the process of modifications to the software. Identify who will sign off on the changes and what would be the criteria for including the changes to the current product. If the changes will affect existing programs, these modules need to be identified.

**7.0 FEATURES TO BE TESTED**

Identify all software features and combinations of software features that will be tested.

**8.0 FEATURES NOT TO BE TESTED**

Identify all features and significant combinations of features which will not be tested and the reasons.

**9.0 RESOURCES/ROLES & RESPONSIBILITIES**

Specify the staff members who are involved in the test project and what their roles are going to be (for example, Mary Brown (User) compile Test Cases for Acceptance Testing). Identify groups responsible for managing, designing, preparing, executing, and resolving the test activities as well as related issues. Also identify groups responsible for providing the test environment. These groups may include developers, testers, operations staff, testing services, etc.

**10.0 SCHEDULES**

Identify the deliverable documents. You can list the following documents:

- Test Plan

- Test Cases

- Test Incident Reports

- Test Summary Reports

**11.0 RISKS/ASSUMPTIONS**

Identify the high-risk assumptions of the test plan. Specify contingency plans for each (for example, delay in delivery of test items might require increased night shift scheduling to meet the delivery date).

**12.0 TOOLS**

List the Automation tools you are going to use. List also the Bug tracking tool here.