Test Plan Template:

Pixel Wizard

Prepared by:

Sinéad Howard

27-04-2020

TABLE OF CONTENTS

1. INTRODUCTION

2.0 OBJECTIVES AND TASKS

2.1 Objectives

2.2 Tasks

3.0 SCOPE

4.0 Testing Strategy

4.1 Unit Testing

4.2 System and Integration Testing

4.3 Performance and Stress Testing

4.4 User Acceptance Testing

4.5 Automated Regression Testing

5.0 Test Schedule

6.0 Control Procedures

7.0 Features to Be Tested

8.0 Features Not to Be Tested

9.0 Resources/Roles & Responsibilities

10.0 Schedules

11.0 Risks/Assumptions

12.0 Tools

1.0 INTRODUCTION

A brief summary of the product being tested. Outline all the functions at a high level.

Pixel Wizard is a 2d side scrolling platform game. In the game the user controls the main character. This character has basic game mechanics that will need to be tested, which are:

**PC**:

* Right arrow/D key to move forward
* Left arrow/A key to move backword
* Up arrow to jump
* Left mouse click/R for action
* C to crouch
* Spacebar to pause or resume the game

**Mobile:**

* Arrow on screen to move forward
* Arrow on screen to move backward
* Arrow on screen to jump
* Arrow on screen to crouch
* Button on screen for action
* Button on top right of screen to pause or resume the game

As well as the game mechanics each separate scene will have to be tested. The first scene is the main menu screen. This consists of 5 option for the player:

* Play game
* Settings
* Load game
* Delete game
* Exit game

**Play Game**

Play Game will bring the user to the first level of the game where all the mechanics and some assets will be tested

**settings**

Settings will load a scene where the user can change some of the settings of the game which will need to be tested for functionality these include:

* Adjust music level
* Adjust sound effects level

**Load Game**

Pressing the load game button will allow users to pick up where they last saved the game

**Exit game**

This button allows the user to exit without playing

The pause button (spacebar for PC, Button for mobile) will load another pause screen. The buttons on this screen will be:

* Save game
* Settings
* Exit game

**Exit and Settings**

Exit game and settings should load the same screen as they do in the main menu.

**Save Game**

The save game allows the user to load the game at any stage. This game should then be available in the load games option in the main menu.

**Game Complete Scene**

Game complete scene gives the user the option to either replay or exit the game.

**Game levels**

There are 3 levels of the game which increase in difficulty. All levels will have the same assets which include:

* Background image
* Player Character sprite
* Enemy character sprite
* Enemy and player fireball
* Ground
* Platform 1
* Platform 2
* Health
* Rock
* Menu logo

Not all of the assets will need to be tested. This will be discussed further in the document.

2.0 OBJECTIVES AND TASKS

2.1 Objectives

Describe the objectives supported by the Master Test Plan, eg., defining tasks and responsibilities,

vehicle for communication, document to be used as a service level agreement, etc.

The main objectives of this test plan are as follows:

* to prevent bugs from happening in the game at as early a stage as possible.
* Sometimes it is not possible to prevent all bugs from happening in software. This means it is important to detect bugs at the earliest stage possible so that they are not carried through every production stage.
* To ensure user enjoyment by reducing bugs
* Software quality will be maintained by keeping bugs at a low level
* Identify testing standards and procedures that will be used on the project
* Used to define how test cases and documentation are written
* Document regression testing guidelines
* Define an exit strategy for testing
* Define the procedure to be undertaken when bugs are found
* Define the measurements that will determine if the test document is successful
* Ensure all tests are repeatable

2.2 Tasks

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

* Identify all software features that will be tested
* Identify items not to be tested
* Identify participants
* Create a testing strategy for the game
  + Define the documentation for unit testing
  + Define the documentation for system and integration testing
  + Define the documentation for stress testing
  + Define the documentation for user acceptance testing
  + Define the documentation for regression testing
  + Define the documentation for beta testing
* Create a test schedule for the game
* Create documentation on how to report problems
* Create documentation for change requests
* Create a measurement for success
* Create an exit strategy

3.0 SCOPE

General

T

This section describes what is being tested, such as all the functions of a specific product, its existing

interfaces, integration of all functions.

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?

4.0 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 sufficient detail to permit identification of the major testing

tasks and estimation of the time required to do each one.

4.1 Unit Testing

Definition:

A unit test tests one small unit at a time. A unit is defined as the smallest testable part of the code and varies depending on language. Unit tests should be designed to determine that a particular piece of code does exactly what it is meant to do when it is meant to do it. It is done using the white box testing method i.ie the item is known to the tester as they choose the inputs and know the expected outputs. Unit testing is the first test to be carried out.

There are many benefits to unit testing including:

* Makes it easier to detect flaws when changing or rewriting code
* Makes code easier to reuse
* Writing the tests in advance makes the testing process faster in the long run
* Catching a defect at a lower level cost less to repair than if it is caught at a later stage of testing

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:

Unit testing is to be carried out by the game testers.

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?

Unit testing will be conducted using a testing framework using the same language that the game will be written in. The testing team will write the scripts for each unit test which test all the functions outlined below.

**Game Level**

A unit test will be written to test:

* Unit scrolling script
* Game audio script

**Playable Character**

A unit test will be written to test:

* Move forwards
* Move backwards
* Jump
* Crouch
* Attack/action
* Pause/resume
* Health script
* Player death script
* Player audio script

**Enemy Character**

A unit test will be written to test:

* Enemy movement pattern
* Enemy attack pattern
* Enemy health
* Enemy death script
* Enemy audio script

**Enemy/player Projectile**

A unit test will be written to test:

* Projectile movement
* Projectile damage

**Main Menu**

A unit test will be written to test:

* Play game button
* Setting button
* Load game button
* Delete game button
* Exit game button
* Intro music script

**Settings Menu**

A unit test will be written to test:

* Adjust music level script
* Adjust sound effects level script

**Pause resume menu**

A unit test will be written to test:

* Save game function
* Settings button
* Exit game button

**End Game Menu**

A unit test will be written to test:

* Replay game script
* Exit game script

These tests can be repeated for each level of the game

It would be recommended to use JIRA which is a software management tool. This will be used to define task for testing track progress and also for bug reporting.

4.2 System and Integration Testing

Definition:

Tests the interactions between separates units when they are integrated together. Tests should be written that detects defects that occur when separate components interact with each other and therefore is usually conducted on a complete system. It is also done after the individual components have been tested themselves. Each integration is tested as soon as it is added to the system until the system is complete.

Some of the benefits of system and integration testing include:

* Helps to detect bugs early
* The integration testing can be carried out during development
* Finds errors in the interface

Participants:

Integration testing to be carried out by the software tester

Methodology:

Integration should only be carried out after unit testing has been completed. The testers will create test cases that test all levels of integration and procedures.

Integration tests should be carried out on the following systems:

The interface between the player character and the enemy character. Do they interact the way they are meant to interact? Does the player health decrease as the enemy attacks? Does the enemy health decrease as the player attacks? Does the enemy character react as expected when the player character appears on the screen?

The interface between the main menu and the first level. Does the menu link to the level correctly? Does every element load when it needs to load?

The interface between the pause/resume menu and each level. Each level needs to be tested separately. Does the menu link to the levels correctly? Does the level change from play mode to pause mode correctly?

The interface between each level. Does the level change to the correct level? Does the level link to the next level correctly?

The interface between the final level and the complete scene. Does the final level link to the scene correctly?

The save game function should be checked to see if the data has reached the database correctly. Storage procedures can be checked.

System integration testing should be conducted each time a new module is added to the game.

The bottom up approach should be used. In this method drivers can be used to manage the test input and output which can be removed once the cluster is tested and combined with the next level.

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:

Performance testing sets the standards for the game. It is used to determine how the game will react under normal circumstances.

Stress testing determines how stable the system is under pressure. It tests how the system reacts to extreme loads and how it recovers from it. Stress testing is done to ensure that the game does not crash under extreme circumstances

Some of the benefits of stress testing include:

* Helps check the games performance under various circumstances
* Allows testers to monitor the game at time of failure
* Can check for data security issues and privacy issues during stress test

Participants:

Who will be conducting Stress Testing on your project? List the individuals that will be responsible

for this activity.

Methodology:

The stress testing will be done by the developer. It Should be completed after at lest one level is completely finalized. Co CPU or GPU testing tools need to be used as the game does not use much processing power.

Stress testing can be carried out by adding enemies to the scene in large numbers to see if this causes the game to lag or crash. All enemies added have to be fully functioning i.e. have the ability to walk and attack the playable character. This stress testing should be done on many different handheld devices with different specifications. This should also be done for the pc version on different operating systems.

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:

While all the other tests have been created under the developer guidelines, User acceptance testing is used to evaluate the game in accordance the user’s guidelines. It is carried out in respect to the users needs. It is the last level of testing before a system is released publicly. Also known as alpha and beta testing

Some of the benefits of user acceptance testing include:

* Reduces the risks of bugs making it through to production
* Improves user experience
* Removes the chance of testing bias

Participants:

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

Methodology:

User acceptance test should be carried out using a beta test. Before the beta test is carried out the game should first be reviewed under the following criteria:

* All components of the game are ready to be tested
* All documentation for end users is complete. This should include:
  + Set-up instructions
  + Installation instructions
  + How to play
  + How to uninstall
* Alpha testing/systems testing should already be undertaken
* The instruction on how to identify bugs, feedback and software used should be given to testers

Beta testing should be monitored by the team manager who chooses a user experience team. The team manager should also decide what reward or incentive should be given to the beta testers.

Beta testing should focus on the following areas:

* Functionality
* User interface
* Response
* User enjoyment

It is recommended the beta testing software ReQtest is used to document the bugs recorded to keep all reporting concise and consistent. All beta testers should be versed on how to use the beta testing software before the testing commences.

Between 10-20 beta testers should be used for testing. The test should cover as many devices as possible to give a good range of testing coverage. Installation packages should be delivered to the testers before product launch.

One test cycle should be completed that lasts four weeks.

All bugs reported by should be sent to the developers to be fixed.

All feedback should be sent to the test leader for evaluation. Feedback should be analysed under the above criteria.

The exit criteria for beta testing:

* Game breaking bugs should no longer be documented on any device or platform.
* Any major bugs that have been reported should be fixed
* The beta summary report should be produced
* Awards/incentives have been distributed to testers
* Beta testing has been signed off by QA

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 Automated Regression Testing

Definition:

Automated regression testing ensures that any changes made to the software , for example, to fix bugs after release, does not have an impact on code that are not associated with it. New tests are not created for regression testing. Older tests are re-executed. It would be ideal to run a full test if the system after every change however, this would be costly and time consuming so an impact analysis can be done to test which area has the most chance of being affected.

Some benefits of automated regression testing include:

* Can be done with automation tools
* Ensures a better quality of game
* More reliable code

Participants:

Methodology:

Because manual execution of test cases increases time as well as cost, It would be recommended to use Ranorex Studio. This is an all in one regression test automation for desktop web and mobile app with built in Selenium WebDriver.

An automated suite of test scripts can be built using previously defined test cases in section 4.1. this automated testing should take place anytime new functionality or a bug fix has taken place.

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.

* Unit scrolling script
* Game audio script
* Move forwards
* Move backwards
* Jump
* Crouch
* Attack/action
* Pause/resume
* Health script
* Player death script
* Player audio script
* Enemy movement pattern
* Enemy attack pattern
* Enemy health
* Enemy death script
* Enemy audio script
* Projectile movement
* Projectile damage
* Play game button
* Setting button
* Load game button
* Delete game button
* Exit game button
* Intro music script
* Adjust music level script
* Adjust sound effects level script
* Save game function
* Settings button
* Exit game button
* Replay game script
* Exit game script
* The interface between the player character and the enemy character
* The interface between the main menu and the first level.
* The interface between the pause/resume menu and each level
* The interface between each level
* The interface between the final level and the complete scene.
* the interface between the save game and the database
* new module testing
* stress test enemy characters
* beta test:
  + Functionality
  + User interface
  + Response
  + User enjoyment
* Regression testing on unit test cases

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

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Risk | Impact | Contingency Plan |
| 1 | Financial risk- risk of going over budget during game development | high | pinpoint all essential items and team members and determine what can be discarded |
| 2 | Design risk- risk of a lack of user interest | high | Analyse beta findings and add or improve features |
| 3 | Market risk- risk that the market for the game will change before the game is released | medium | Analyse the market and use the findings to improve game for current climate |
| 4 | Technology risk- risk that any of the software the game is reliant on will fail | medium | Create back-ups of all files, Delegate member of team to be in charge of recovery testing |
| 5 | Schedule risk- risk of a delay in the release date | medium | Developers may need to work overnight or weekend shifts to catch up on the work |
| 6 | Quality risk- risk of game breaking bugs | high | All test cases should be re-run and all code should be tested by delegated staff until bug is found and fixed |

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.

Below is a list of all the recommended software tools for testing:

* Jira – Unit Testing
* ReQTest – Beta Testing
* Ranorex Studio - Automated Regression Testing