Charlie Conneely – G00348887

Test Plan Template: The Pixel Wizard

Prepared by Charlie Conneely

Date: 8/5/20

Table of Contents:

1.0 Introduction

2.0 Objectives and 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 Batch Testing

4.6 Automated Regression Testing

4.7 Beta 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. **Introduction**

The Pixel Wizard is a 2D side-scrolling platform game with artwork inspired by games like Shovel Knight, which also use pixel art. The player/wizard will navigate through each round with increasing difficulty.

In each round the player must use magic to defeat the encountered enemies whilst collecting pickups which will replenish the wizard’s health. The “magic” refers to the wizard’s ability to shoot fireballs and freeze spells at the enemy.

Each round will also present a “boss” that the player must defeat to progress to the next round. The player will be able to see the boss’s health status on the screen as well as their own.

1. **Objectives and Tasks**

The main objectives of the testing process are:

* To achieve 100% decision coverage.
* To ensure that the product will be launched with no critical bugs.
* To ensure that all team members are aware of their responsibilities.
* To ensure that all team members are aware of the scope and the location of their tasks within.

The tasks involved:

* TO-DO

1. **Scope**

The testing lifecycle will take place between 1/6/20 and 5/8/20.

Throughout the testing lifecycle, the following functions/interfaces of The Pixel Wizard will be tested:

* Start Menu. Ensure that all buttons work correctly (including the sound and music levels inside Settings). We need to ensure that all “back” buttons work correctly i.e. that the user can always swiftly return to the menu.
* In-Game Menu. All buttons inside this menu should work essentially the same as in the start menu. However, in place of the “start game” option, the user should see a “back to main menu option”. All buttons should be thoroughly tested for bugs. The restart option should reset everything and take the user back to the beginning of the round. To open/close the game menu, simply press the escape button.
* Gameplay. All functions need to be tested (shooting fireballs/freeze spells).

-- The player should be presented with instructions at the start of the level.

-- The player should progress to the next round after each level is complete.

-- The game difficulty should increase with the rounds.

-- Upon completion, the user should be presented with the option to restart or exit the game.

Throughout the testing process. All bugs should be logged under the correct heading with their test case number in the documentation on GitHub, with information such as the severity of the bug, it’s location in the game, which stage of the test process the tester was in and timestamp.

Depending on the severity of the bug, the tester may need to announce in the teams group chat that the schedule should be re-examined as a result. The developer(s) responsible for this area of the game will subsequently be contacted and the information will be relayed. Then, a meeting may or may not be held to reorganise the testing process going forward.

1. **Testing Strategy**
   1. Unit Testing  
      Before we begin the testing process, there will be a meeting held on 2/6 at 10 am to conduct a static test of the various functions within the game’s software. An overview of the algorithms used will be presented and time will be allocated at the end to allow for questions and for more in-detail inspections of the code if necessary. Attendance is mandatory for all team members. Dynamic testing of the components will follow.   
        
      Participants: Mary Hanson, John McMahon, Andrew Gordon.  
        
      Each participant will be provided with their test conditions, test basis, test data and testware. All incidents are to be logged to the shared GitHub repository in the “Unit Testing” document along with all details of any incidents encountered. As most functions are only applicable when playing the game, the purpose of this testing is to ensure that there are no critical bugs found, that all areas of the code will be executed at least once and that these functions perform well when given certain test data.
   2. System and Integration Testing  
      During this time we will be testing how the program interacts with the database, how the different components of the program operate together and if the program meets the requirements defined in the scope.  
        
      Participants: Edward Minh, Pat O’Brien, Andrew Gordon.  
        
      This will entail testing:   
      -The load/save/delete game options and if they interact with the database correctly.  
      -How the program performs when navigating between the main menu and the game or game and the pause menu etc.  
      -How does the game hold up under different circumstances pertaining to hardware e.g. monitor with different frame rate.   
      -Does the game satisfy each requirement as specified by the scope and will it be satisfactory to the user.   
      -Are the choices for the control mechanisms consistent across different platforms/hardware.  
      All bugs found should be logged alongside their test case no. and their severity in the shared “System/Integration Testing” document.
   3. Performance and Stress Testing  
      During this period we will focus on the boundaries/limitations of the game and how the system will handle being under such pressure.  
        
      Participants: Edward Minh, Mary Hanson, Pat O’Brien  
        
      Activities involved:  
      -Rapid in-game button bashing – there will be an automated testing application supplied to put the buttons for firing/freezing and moving under stress to find limitations.   
      -Automated main/pause menu rapid decision making. A similar automated testing process will take place to put the navigation of the menu system under stress.  
      -The load function for different versions of gameplay will be tested in order to find the limit in relation to the number of versions that can be saved and to find if any errors occur when such stress testing takes place.  
      All bugs/limitations found should be logged alongside their test case no. and their severity in the shared “Performance/Stress Testing” document.