Pixel Wizard Test Plan

Prepared by: Arnas Steponavicius

May 2020

Contents

1 Introduction

This game will be a 2D side-scrolling platformer, inspired by the likes of '**Salt and Sanctuary**', '**Shovel Knight**', and '**Fancy Pants**', with elements from '**Skyrim**' (mainly in the way the player character and enemy characters attack). The artwork will be inspired mainly by Shovel Knight, which uses mainly pixel art to create its characters and world.
The gameplay will be inspired by '**Salt and Sanctuary**' and '**Dark Souls**' and '**Skyrim**', which will see the player navigate progressively difficult levels with a wizard type character that uses magic a lá '**Skyrim**'.
Each level will have several enemies that the player must defeat to progress. Each level will also have a boss that the player must defeat to progress to the next level. Each level will contain pickups for the player, such as health pickups to replenish the player's health.

Game Characteristics
As this will be a platformer, this game will:

- Allow the player to control a specific character, that has an important fictional/narrative role.
- Have game statistics and/or relational attributes with other game objects, enemies, and/or the player character.
- Allow the player to take on and navigate the levels using an easy-to-use user interface.
- Have obstacles that the player must overcome, such as enemies and bosses.

2 Objectives and Tasks

2.1 Objectives

The Objectives include:

- Verify that the functionality of "Pixel Wizard" works in the same way as outlined in the specifications.
- Using industry standard testing techniques to look for major and minor defects in the game.
- Ensure that if major defects are found they are reported and fixed before full-release of the game.
- Communication between the various testing teams through Microsoft teams meetings in the mornings on what tasks are to be done, as well as convey the completed tasks to each team.
- Track issues and progress on known defects through GitHub.

2.2 Tasks

1. Menu Testing
2. Gameplay Testing
3. Combat Testing
4. Game Exploitation
5. Item usage and collection

3 Scope

3.1 General

Using the above objectives and tasks, the scope of testing will cover the major features and most minor features of the game. Each feature tested will have a priority the most crucial ones being tested most extensively to ensure a high quality product. One of such features includes "Gameplay" as it is a vital feature.
Minor feature testing would include things along the lines of typos in quests and so on, as they are not game breaking and can therefore be fixed at a later stage.

3.2 Tactics

- Assign teams to certain features of the game to be tested.
- Morning meeting with all teams to go over previous days and current days workload.
- Meeting between each individual group to go over each persons task.
- Search for defects.
- Create an issue if a defect is found so it can be fixed.

4 Testing Strategy

4.1 Unit Testing

Unit testing will be used to test individual components by teams, such as the menu functionality, level navigation, defeat scenarios and player control to name a few. This type of testing can only be used in certain areas as the game state constantly changes and therefore wont return consistent data in randomly generated scenarios of the game.

4.2 System and Integration Testing

One specific team will be designated to test the interaction between the software and hardware of the game. The team will be responsible for monitoring hardware usage when the game is running and assessing it to check whether the game runs fine without using to much a systems resources.
The team will test on different machines with different specifications to see interactions. If defects are found they must be reported as the interaction with the system and hardware is vital.

4.3 Performance and Stress Testing

Similar to the testing in System and Integration testing section, A team will test the game on a variety of machines with varying specifications and observe results by pushing the game beyond a normal operating capacity to test the stability of the game. Some tests may include spawning a mass amount of items in a small area to check the stability of the game under a heavy load of item initialization and environment interaction.

4.4 User Acceptance Testing

This test will work in tandem with Beta testing to ensure the game can be accepted by an users upon release. Depending on results and responses of the Beta testing stage. It will determine whether the game will pass the user acceptance test.

4.5 Batch Testing

A team will be responsible for preparing scripts to run tests on the menu aspect of the game. This is to ensure all possible menu routes work correctly.

4.6 Beta Testing

The game will be released for a period of time to allow users to test out the product. Users will allowed to report defects that they find in the game and they will be closely monitored so they can be patched after the beta.

5 Test Schedule

- Gameplay (Weeks 1 - 3)
    1. Movement
    2. Object Interaction
    3. Enemy Behaviour
    4. Item usage
    5. Combat Mechanics
    6. Dialogue
    7. End Game Conditions

- Hardware (Week 4)
    1. Memory Interaction
    2. CPU Usage
    3. GPU Usage
    4. Interaction with System Applications
    5. Interaction with 3rd Party Applications

6 Control Procedures

Once a defect is found the individual who found the defect must create a GitHub issue on the private repository. This issue will then be brought up in the morning meetings where it will be assigned a priority. Based upon the priority of the defect it will be assigned to individual to be patched.

7 Features to be tested

8 Features NOT to be tested

9 Resources/Roles & Responsibilities

10 Schedules

11 Risks/Assumptions

12 Tools