# Platform Perils

## Test Plan

Steven Palmer
⟨palmes4⟩
Chao Ye
⟨yec6⟩

April 26, 2016

# Contents


**1 Overview** **1**
- 1.1 Test Case Format . . . 1
- 1.2 Automated Testing . . . 2
  - 1.2.1 Testing Tools . . . 2
- 1.3 Manual Testing . . . 2
- 1.4 List of Constants . . . 2

**2 Proof of Concept Testing** **3**
- 2.1 Significant Risks . . . 3
- 2.2 Demonstration Plan . . . 3

**3 System Testing** **5**
- 3.1 Game Mechanics Testing . . . 5
  - 3.1.1 Automated Testing . . . 5
  - 3.1.2 Manual Testing . . . 10
- 3.2 Game Design Testing . . . 12
  - 3.2.1 Game World Testing . . . 12
  - 3.2.2 Graphics Testing . . . 13
  - 3.2.3 Audio Testing . . . 14
  - 3.2.4 Miscellaneous Testing . . . 14

**4 Future Testing** **16**
- 4.1 User Experience Testing . . . 16

**5 Trace to Requirements** **17**
- 5.1 Functional Requirements . . . 17

**6 Timeline** **19**

**7 Appendix A: Testing Survey** **20**


# List of Tables


1 List of testing tools . . . 2
2 List of constants . . . 3

3 **Requirements Traceability** . . . . . 18
4 Testing timeline . . . . . 19

# List of Figures

1 Proof of concept sketch . . . . . 4

**Revision History**

| Date | Version | Notes |
|---|---|---|
| October 25, 2015 | 1.0 | Created document |
| October 31, 2015 | 1.1 | Major additions to all sections |
| November 1, 2015 | 1.2 | Final version for rev 0 |
| April 25, 2015 | 1.3 | Final version for rev 1 |

# 1 Overview

The purpose of this document is to provide a detailed plan for the testing of our game. The following brief outline gives an overview of what is covered in this document:

- A proof of concept test is described in §2.
- The set of tests that will be used in testing the system is described in §3.
- The set of tests that will be used to ensure that the software requirements specifications are met is described in §5.
- A timeline of the test plan is given in §6.

## 1.1 Test Case Format

The description of the tests that will be carried out are formatted in the following way throughout the document:

| | |
|---|---|
| **Test #:** | **Test name** |
| **Description:** | A description of what is being tested |
| **Type:** | The type of test |
| **Tester(s):** | The people who will run the test (**manual only**) |
| **Initial State:** | The initial state of the system being tested (**unit test only**) |
| **Input:** | The input that will change the state of the system (**unit test only**) |
| **Output:** | The relevant output that is checked (**unit test only**) |
| **Pass:** | The pass criteria for the relevant output in the case of unit tests, or a description of the pass criteria for other tests |

## 1.2 Automated Testing

Automated testing will be used for testing of the game mechanics system and will be carried out via unit test suites. Stubs and drivers will not be used due to the heavy use of side-effects in game code. To ensure that unit testing can be run while the game mechanics are in development the test cases have been grouped into sections that can be fully implemented once certain subsets of the game mechanics systems are completed.

### 1.2.1 Testing Tools

The software tools that will be used to carry out the automated testing are listed in Table 1.

**Table 1:** List of testing tools

| **Tool** | **Description** | **Use** |
|---|---|---|
| gUnit | Unit testing framework | Unit testing |

## 1.3 Manual Testing

Manual tests will be used for testing all aspects of the game that are not covered by the automated unit testing suite (the time that would be required to develop automated tests for these test cases would be prohibitive). The game development team will carry out these tests as the game is developed.

## 1.4 List of Constants

Constants used in this document are listed in Table 2.

**Table 2:** List of constants

| Constant | Value | Description |
|---|---|---|
| $\sigma$ | 60 | Frame rate target |
| $\delta$ | 3 | Number of people in user testing group |
| $\Theta$ | 6 | User testing entertainment target |
| $\Psi$ | 2 | User testing challenge range |
| $\Omega$ | 8 | User testing controls target |
| $\Phi$ | 250 | Dynamic object limit |

# 2 Proof of Concept Testing

Before any serious development of the game begins, a proof of concept test will be carried out to show that the undertaking is feasible. The remainder of this section describes the proof of concept test in detail.

## 2.1 Significant Risks

The successful completion of the project depends on overcoming the following significant risks:

1. In order to use the Chipmunk2D library it must first be successfully compiled. Since we intend for the game to be compatible with Windows 7, Mac OS X, and Ubuntu, there is a significant risk for the project to fail if compilation is not achieved on all three operating systems.

2. Chipmunk2D is a large library and its use is not straight forward. Successful implementation of the library features is crucial to the success of the project and the failure of this poses another significant risk.

## 2.2 Demonstration Plan

For a proof of concept test we will produce a working prototype that can be run on Windows 7, Mac OS X, and Ubuntu. The prototype will consist of a game demo that implements gravity and collision detection provided by the Chipmunk2D library. Rudimentary graphics will be used for the prototype since the scope is limited only to demonstrating that the identified risks can be overcome.

The prototype will consist of a small room in which a hero character and enemies exist. The room will be bounded by a floor below and walls on the left and right, all of which the hero and enemies cannot pass through. The room will contain platforms which the hero and enemies cannot pass through from above, but may pass through from below when jumping. A rough idea of the room is given in Figure 1.

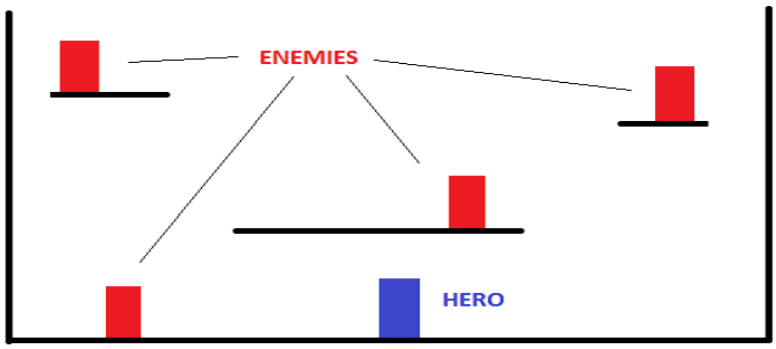


**Figure 1:** Proof of concept sketch

The hero character will be represented by a blue rectangle and will be controlled by the user in the following ways:

- The hero moves left and right using the 'a' and 'd' keys respectively
- The hero jumps by pressing the 'space' key
- The hero shoots a projectile in the direction of the mouse cursor by left-clicking

Enemies will be represented by red rectangles and will not have any programmed AI (they will not move or attack). The hero will be able to attack enemies with a projectile, which will knock them back when they are hit. The hero and all enemies will be subject to gravity and will free-fall when there is no platform or boundary under them.

The proof of concept is given in test case format to adhere with the presentation of the other tests in this document.

| | |
|---|---|
| **Test #1:** | **Proof of Concept** |
| **Description:** | Tests whether significant risks to the completion of the project can be overcome |
| **Type:** | Proof of Concept (manual) |
| **Tester(s):** | Game developers |
| **Pass:** | Successful development of a small demonstration which makes use of the Chipmunk2D physics engine and runs on Windows 7, Mac OS X, and Ubuntu |

# 3 System Testing

The testing of the system is broken down into game mechanics testing and game design testing phases. The game mechanics will be developed and tested first. Once the game mechanics systems are in place, the game design development and testing phase will begin.

## 3.1 Game Mechanics Testing

### 3.1.1 Automated Testing

A limited set of unit tests will be used to test the game mechanics and physics. This test suite will be used to ensure that changes to the game do not break the basic functioning of the game. The test cases that will be used are outlined in the remainder of this section.

| **Test #2:** | **Move left** |
|---|---|
| **Description:** | Tests if the hero moves left when the corresponding input is received when the hero is initially stationary |
| **Type:** | Unit Test (dynamic, automated) |
| **Initial State:** | Custom in-game state with a hero object having x-velocity of zero |
| **Input:** | Keyboard function called with simulated left key down stroke |
| **Output:** | Hero object x-velocity |
| **Pass:** | Hero object x-velocity is less than zero |

| **Test #3:** | **Move right** |
|---|---|
| **Description:** | Tests if the hero moves right when the corresponding input is received when the hero is initially stationary |
| **Type:** | Unit Test (dynamic, automated) |
| **Initial State:** | Custom in-game state with hero object having x-velocity of zero |
| **Input:** | Keyboard function called with simulated right key down stroke |
| **Output:** | Hero object x-velocity |
| **Pass:** | Hero object x-velocity is greater than zero |

| **Test #4:** | **Stop moving left** |
|---|---|
| **Description:** | Tests if hero stops moving left when corresponding input is stopped |
| **Type:** | Unit Test (dynamic, automated) |
| **Initial State:** | Custom in-game state with hero object having x-velocity less than zero |
| **Input:** | Keyboard function called with simulated left key up stroke |
| **Output:** | Hero object x-velocity |
| **Pass:** | Hero object x-velocity is zero |

| **Test #5:** | **Stop moving right** |
|---|---|
| **Description:** | Tests if hero stops moving right when corresponding input is stopped |
| **Type:** | Unit Test (dynamic, automated) |
| **Initial State:** | Custom in-game state with hero object having x-velocity greater than zero |
| **Input:** | Keyboard function called with simulated right key up stroke |
| **Output:** | Hero object x-velocity |
| **Pass:** | Hero object x-velocity is zero |

| **Test #6:** | **Jump from static object** |
|---|---|
| **Description:** | Tests if hero jumps off a static object when corresponding input is received |
| **Type:** | Unit Test (dynamic, automated) |
| **Initial State:** | Custom in-game state with hero object having y-velocity of zero and a bottom edge in contact with a static object |
| **Input:** | Keyboard function called with simulated space bar key down stroke |
| **Output:** | Hero object y-velocity |
| **Pass:** | Hero object y-velocity is greater than zero |

| **Test #7:** | **Wall obstructs hero moving left** |
|---|---|
| **Description:** | Tests whether the hero is stopped by a wall object while moving left |
| **Type:** | Unit Test (dynamic, automated) |
| **Initial State:** | Custom in-game state with hero object having x-velocity less than zero situated directly to the right of a wall object |
| **Input:** | The chipmunk cpSpaceStep function is called |
| **Output:** | Hero object x-velocity |
| **Pass:** | Hero object x-velocity is 0 |

| **Test #8:** | **Wall obstructs hero moving right** |
|---|---|
| **Description:** | Tests whether the hero is stopped by a wall object while moving right |
| **Type:** | Unit Test (dynamic, automated) |
| **Initial State:** | Custom in-game state with hero object having x-velocity greater than zero situated directly to the left of a wall object |
| **Input:** | The chipmunk cpSpaceStep function is called |
| **Output:** | Hero object x-velocity |
| **Pass:** | Hero object x-velocity is 0 |

| **Test #9:** | **Floor supports stationary hero** |
|---|---|
| **Description:** | Tests whether the hero is supported by a floor object |
| **Type:** | Unit Test (dynamic, automated) |
| **Initial State:** | Custom in-game state with stationary hero object situated directly on top of a floor object |
| **Input:** | The chipmunk cpSpaceStep function is called |
| **Output:** | Hero object y-velocity |
| **Pass:** | Hero object y-velocity is 0 |

| | |
|---|---|
| **Test #10:** | **Floor stops hero in free fall** |
| **Description:** | Tests whether the hero in free fall is stopped by a floor object |
| **Type:** | Unit Test (dynamic, automated) |
| **Initial State:** | Custom in-game state with hero object with y-velocity less than zero situated directly on top of a floor object |
| **Input:** | The chipmunk cpSpaceStep function is called |
| **Output:** | Hero object y-velocity |
| **Pass:** | Hero object y-velocity is 0 |

### 3.1.2 Manual Testing

The following manual tests will be carried out to provide testing for game mechanics related requirements that are not covered by unit tests.

| | |
|---|---|
| **Test #11:** | **No mid-air jumps** |
| **Description:** | Tests that the hero cannot jump when not in contact with a surface |
| **Type:** | Functional (dynamic, manual) |
| **Tester(s):** | Development team |
| **Pass:** | Hero cannot jump when not in contact with a surface |

| | |
|---|---|
| **Test #12:** | **Zoom in test** |
| **Description:** | Tests that the stage zooming works properly (in) |
| **Type:** | Functional (dynamic, manual) |
| **Tester(s):** | Development team |
| **Pass:** | Stage view can be zoomed in |

| **Test #13:** | **Zoom out test** |
|---|---|
| **Description:** | Tests that the stage zooming works properly (out) |
| **Type:** | Functional (dynamic, manual) |
| **Tester(s):** | Development team |
| **Pass:** | Stage view can be zoomed out |

| **Test #14:** | **Hero death** |
|---|---|
| **Description:** | Tests that the hero is killed when coming into contact with a fatal hazard |
| **Type:** | Functional (dynamic, manual) |
| **Tester(s):** | Development team |
| **Pass:** | Hero is killed when contacting fatal hazards |

| **Test #15:** | **Stage win** |
|---|---|
| **Description:** | Tests that the stage is won when hero comes in contact with the checkered goal |
| **Type:** | Functional (dynamic, manual) |
| **Tester(s):** | Development team |
| **Pass:** | Stage is won and user is returned to the main menu |

| **Test #16:** | **General physics behaviour** |
|---|---|
| **Description:** | Tests that the physics behaves as expected from a qualitative perspective |
| **Type:** | Functional (dynamic, manual) |
| **Tester(s):** | Development team |
| **Pass:** | Physics appears to function correctly |

| **Test #17:** | **General collision behaviour** |
|---|---|
| **Description:** | Tests that collisions behave as expected from a qualitative perspective |
| **Type:** | Functional (dynamic, manual) |
| **Tester(s):** | Development team |
| **Pass:** | Collisions appear to behave correctly |

## 3.2 Game Design Testing

Once the game mechanics systems have been implemented and shown to be working correctly through the testing described in §3.1, the game itself will be built on top. The design of the game can be broken down into the design of the game world, graphics, and sound. Automated testing for this phase would be time-consuming and difficult to implement. Therefore, all of the game design testing will consist of manual tests.

### 3.2.1 Game World Testing

The following tests will be carried out to ensure that the game world is designed correctly.

| **Test #18:** | **All areas reachable** |
|---|---|
| **Description:** | Tests that all areas of the game world that are intended to be reachable by the hero are in fact reachable by the hero |
| **Type:** | Functional (dynamic, manual) |
| **Tester(s):** | Development team |
| **Pass:** | No areas are unreachable based on a thorough playthrough testing of the game |

| **Test #19:** | **No "points of no return"** |
|---|---|
| **Description:** | Tests that there are no areas of the game world that will cause the hero to become stuck (e.g. inescapable pits) |
| **Type:** | Functional (dynamic, manual) |
| **Tester(s):** | Development team |
| **Pass:** | There are no inescapable areas detected on a thorough playthrough testing of the game |

### 3.2.2 Graphics Testing

The following tests will be carried out to ensure that the game graphics are properly implemented.

| **Test #20:** | **Textures** |
|---|---|
| **Description:** | Tests if textures are properly implemented |
| **Type:** | Functional (dynamic, manual) |
| **Tester(s):** | Development team |
| **Pass:** | In-game textures appear correct by inspection |

| **Test #21:** | **Lighting** |
|---|---|
| **Description:** | Tests if lighting effects are properly implemented |
| **Type:** | Functional (dynamic, manual) |
| **Tester(s):** | Development team |
| **Pass:** | Lighting effects appear correct by inspection |

### 3.2.3 Audio Testing

The following tests will be carried out to ensure that the game audio is properly implemented.

| **Test #22:** | **Background music** |
|---|---|
| **Description:** | Tests if background music is properly implemented |
| **Type:** | Functional (dynamic, manual) |
| **Tester(s):** | Development team |
| **Pass:** | Background music plays while in game |

| **Test #23:** | **Sound effects** |
|---|---|
| **Description:** | Tests if sound effects are properly implemented |
| **Type:** | Functional (dynamic, manual) |
| **Tester(s):** | Development team |
| **Pass:** | Appropriate sounds play when events take place (e.g. hero death, complete stage) |

### 3.2.4 Miscellaneous Testing

The following tests will be carried out to ensure all requirements not tested under the previous categories are met.

| **Test #24:** | **Menu system** |
|---|---|
| **Description:** | The menu system works as intended |
| **Type:** | Functional (dynamic, manual) |
| **Tester(s):** | Development team |
| **Pass:** | All menu options work correctly |

| **Test #25:** | **General look and feel** |
|---|---|
| **Description:** | The game has the intended look and feel |
| **Type:** | Functional (dynamic, manual) |
| **Tester(s):** | Development team |
| **Pass:** | The game is a 2.5-D platformer with an Indiana Jones adventure theme |

| **Test #26:** | **Operating system support** |
|---|---|
| **Description:** | The game runs on Windows 7, Mac OS X, and Ubuntu |
| **Type:** | Functional (dynamic, manual) |
| **Tester(s):** | Development team |
| **Pass:** | Game can be compiled and system tests all pass on each platform |

| **Test #27:** | **Spelling and grammar check** |
|---|---|
| **Description:** | The game uses proper English and is free of any spelling or grammatical errors |
| **Type:** | Functional (dynamic, manual) |
| **Tester(s):** | Development team |
| **Pass:** | No spelling or grammatical errors are detected (or all detected errors are corrected) |

| **Test #28:** | **Hardware requirements** |
|---|---|
| **Description:** | Tests for the minimum hardware requirements required to maintain an average frame rate of at least $\sigma$ frames per second |
| **Type:** | Functional (dynamic, manual) |
| **Tester(s):** | Development team |
| **Pass:** | Game maintains frame rate of at least $\sigma$ frames per second in a stage that contains $\Phi$ dynamic objects when tested on a low-performance system |

# 4 Future Testing

## 4.1 User Experience Testing

User experience testing will be completed by a testing group consisting of $\delta$ individuals who were not involved in the development of the game. The testing group will be given a copy of the game and asked to complete a survey (see Appendix A) after having played the game to provide feedback. Due to time constraints, these tests will be carried out subsequent to the revision 1 project submission.

| **Test #29:** | **Entertainment** |
|---|---|
| **Description:** | Tests that the game is entertaining |
| **Type:** | Functional (dynamic, manual) |
| **Tester(s):** | Testing group |
| **Pass:** | Average survey score of at least $\Theta$ |

| **Test #30:** | **Challenge** |
|---|---|
| **Description:** | Tests that the game is adequately challenging (not too easy or too hard) |
| **Type:** | Functional (dynamic, manual) |
| **Tester(s):** | Testing group |
| **Pass:** | Average survey score within $\Psi$ of 5 |

| **Test #31:** | **Controls** |
|---|---|
| **Description:** | Tests that the game controls are intuitive |
| **Type:** | Functional (dynamic, manual) |
| **Tester(s):** | Testing group |
| **Pass:** | Average survey score of at least $\Omega$ |

# 5 Trace to Requirements

## 5.1 Functional Requirements

A trace between tests and functional requirements is provided in Table 3.

**Table 3: Requirements Traceability**

| Requirement | Test(s) |
|---|---|
| 1 | 24 |
| 2 | 24 |
| 3 | 24 |
| 4 | 24 |
| 5 | 2, 4 |
| 6 | 3, 5 |
| 7 | 6 |
| 8 | 11 |
| 9 | 16 |
| 10 | 19 |
| 11 | 18 |
| 12 | 12 |
| 13 | 13 |
| 14 | 9, 10 |
| 15 | 7, 8 |
| 16 | 14, 17 |
| 17 | 14 |
| 18 | 15 |
| 19 | 24 |
| 20 | 25 |
| 21 | 25 |
| 22 | 20, 21, 25 |
| 23 | 22, 23 |
| 24 | 29 |
| 25 | 31 |
| 26 | 30 |
| 27 | 28 |
| 28 | 26 |
| 29 | 27 |

# 6 Timeline

This document is structured to roughly match the anticipated chronology of testing. The testing timeline is given in Table 4.

**Table 4:** Testing timeline

| **Completion Date** | **Responsible Party** | **Task** |
|---|---|---|
| 11/15/2015 | Development team | Completion of the proof of concept demo |
| 04/15/2016 | Chao Ye | Implementation of input unit test cases |
| 04/15/2016 | Chao Ye | Implementation of collision unit test cases |
| 04/18/2016 | Steven Palmer | Completion of game design testing |
| Future | Testing group | Completion of user experience survey |

# 7 Appendix A: Testing Survey

The following survey will be filled out by members of the alpha and beta testing groups.

### User Experience Survey

The following survey should be filled out after playing the game for at least 30 minutes.

**Time played:**

Please provide a ranking between 0 and 10 in each of the following categories. Please include notes on what you did and did not like, and what could be done to improve the game.

**Entertainment:** 0 1 2 3 4 5 6 7 8 9 10
[ 0 = most boring, 10 = most fun ]

**Difficulty:** 0 1 2 3 4 5 6 7 8 9 10
[ 0 = easiest, 10 = most difficult ]

**Controls:** 0 1 2 3 4 5 6 7 8 9 10
[ 0 = non-intuitive, 10 = intuitive ]

**Notes:**