Testing Documentation for

Send Me Home

Revision History

Version	Date	Authors
1.0	11/12/2020	Yucheng Long, Adam Maxwell, Bryce Mecham

Table of Contents

Project description

Our project is a simple platformer game made for mobile devices. It is built with C++ and Cocos2d-x as the game engine. It will include an alien trying to fix his spaceship to get back home. There will be about five different levels. The alien will have to jump on different platforms to collect parts for his spaceship. In the process of finding the parts, he will have to avoid enemies. After collecting the parts to the spaceship, the alien will have to make it back to his spaceship to advance it to the next level. The user will be able to control the alien by a virtual joystick for movement and tap to jump.

Overall testing plan

Our testing approach will test each feature as they are implemented. There will be regression testing included to make sure that the new features don't break the old features. By doing this, it will make our program less buggy and we won't have to change as much later. We will test the system as a whole at the end to ensure that our game runs smoothly and bug free on multiple devices. A lot of our testing will be printing to the console to verify positions, events, or actions have occurred. We will have another team member test our code, and we will have outside people test our game to avoid bias in our testing.

Team assignments:

- Unit Testing: Adam Maxwell
- Integration Testing: Bryce Mecham
- Validation Testing: Everyone
- System Testing: Yucheng Long

Unit Tests

UI Test

This will be tested by clicking buttons and sliders to make sure the UI manipulates variables and switches to the corresponding object. We will print the action to the console to verify that it has been manipulated correctly.

Player Test

To run this test we will be printing out the players instance variables to the console to make sure they are being manipulated in accordance to the actions that are being sent to the player.

Enemy Test

To run this test we will be printing out the enemy instance variables to the console to make sure they are being manipulated in accordance to the actions that are being sent to the enemy.

Settings Test

We will make sure that the audio and control switches change the actual audio and control interface.

Collision Test

The collision test will be to make sure that when two objects collide the object is either picked up, or the player loses some life depending on if it is a spaceship part, or an enemy.. We will test this by printing to the console when a collision is supposed to occur.

Integration Tests

We will use a bottom up approach. Because we start with the lowest hierarchical components and move to the highest hierarchical components.

Object Test

Our integration Tests will start with the game object test. Our object test will make sure that it is integrated with our game. This test will include updates to positions, and states

of our game objects. It will also include the tests of game objects being rendered to the screen in the right positions. This will be tested via printing values to the console.

Map Test

We would test the map next. We will test if it is rendered to the background, and that animations on the background are running using the game's render and update methods.

UI Test

Finally we would test the UI. We will test to make sure the overlay appears on top of the background and over all game objects. We will make sure that it will accept user input to the game and that it manipulates the game's states accordingly.

Validation Tests

Player Control Test

We will test this by providing an action to the player and see if the player responds to that action. This will validate functional requirement FR-1 in our Software Requirements Diagram.

Graphics Rendering Test

We will test this by providing a graphic, and see if it appears on the screen in the right position. This will validate functional requirement FR-2 in our Software Requirements Diagram.

Game Object Update Test

We will provide an event and see that any game objects affected by the event respond accordingly. This will validate functional requirement FR-3 in our Software Requirements Diagram.

End Objective Test

We will test this by when the player reaches the goal, that the game ends and the player is deemed the winner. This will validate functional requirement FR-4 in our Software Requirements Diagram.

Menu Test

We will test this by seeing if the start menu and the settings menu have all the buttons they need, and respond to user input. This will validate functional requirement FR-5 in our Software Requirements Diagram.

System Tests

Recovery Test

We will test that the game boots back up after an artificially made system crash that will emulate a real one. This will also test whether the program has a reliable backup measure.

Performance Test

We want our game to look and play smoothly. We will test that our game runs at 60 frames per second. This will be tested by putting in an overlay over the game screen that shows our current frames per second while playing the game, to verify that our game plays at around 60 frames per second.

Deployment Test

This test will be to see if the game runs on windows, android, and ios devices by deploying them to each different device. Then on each device we will check to make sure that the resolution/graphics rendering is shown correctly, that the user input works as normal, and that the settings controls are working as expected.