

Testing Documentation for Send Me Home

Revision History

Version	Date	Authors
1.0	11/12/2020	Yucheng Long, Adam Maxwell, Bryce Mecham
2.0	12/11/2020	Adam Maxwell, Bryce Mecham

Table of Contents

Revision History	2
Table of Contents	3
Project description	4
Overall testing plan	4
Unit Tests	4
UI Test	4
Player Test	5
Enemy Test	5
Settings Test	5
Collision Test	5
Integration Tests	5
Object Test	5
Map Test	6
UI Test	6
Validation Tests	6
Player Control Test	6
Graphics Rendering Test	6
Game Object Update Test	7
End Objective Test	7
Menu Test	7
System Tests	7
Recovery Test	7
Performance Test	7
Deployment Test	8

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:

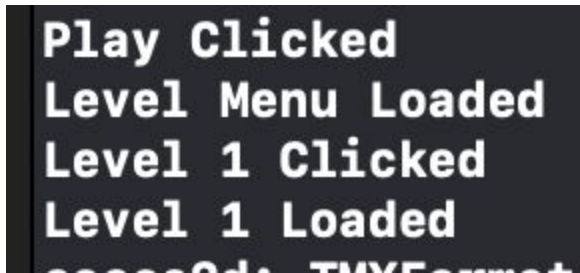
- Unit Testing: Adam Maxwell
- Integration Testing: Bryce Mecham
- Validation Testing: Everyone
- System Testing: Yucheng Long

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.

UI Test Results



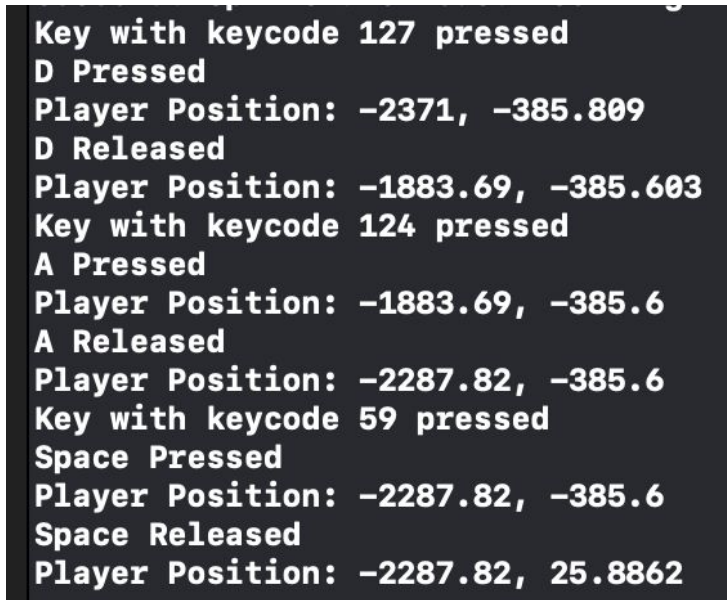
```
Play Clicked
Level Menu Loaded
Level 1 Clicked
Level 1 Loaded
```

This shows the menu buttons clicked and that the level was successfully loaded.

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.

Player Test Results



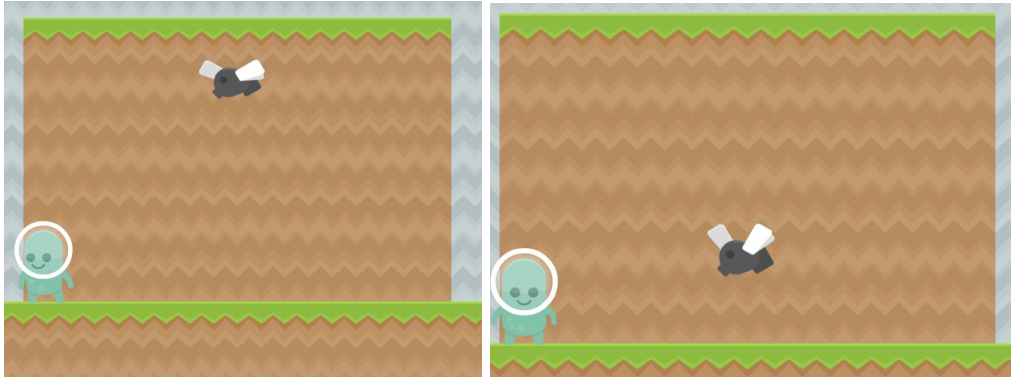
```
Key with keycode 127 pressed
D Pressed
Player Position: -2371, -385.809
D Released
Player Position: -1883.69, -385.603
Key with keycode 124 pressed
A Pressed
Player Position: -1883.69, -385.6
A Released
Player Position: -2287.82, -385.6
Key with keycode 59 pressed
Space Pressed
Player Position: -2287.82, -385.6
Space Released
Player Position: -2287.82, 25.8862
```

This shows the key pressed and the position updates correctly.

Enemy Test

To run this test we will be watching visually to see if the enemy moves.

Enemy Test Results

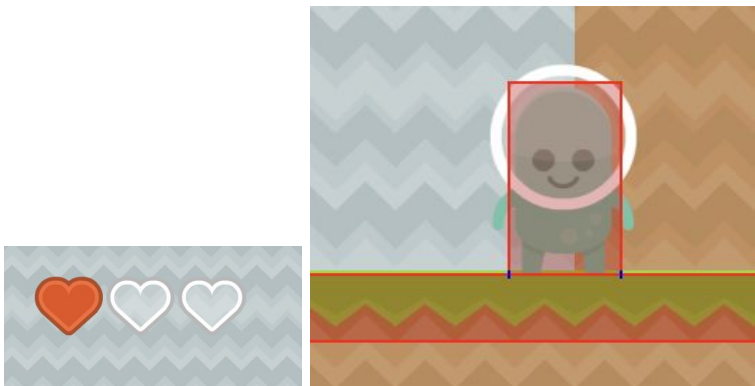


The fly's position moves up, down, and updates to the screen correctly.

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 displaying visual boxes that turn blue on parts that intersect other physics bodies.

Collision Test Results



Hearts go down when colliding with enemies, and the collision box turns blue when in contact with another physics body (look to the right and left of the feet).

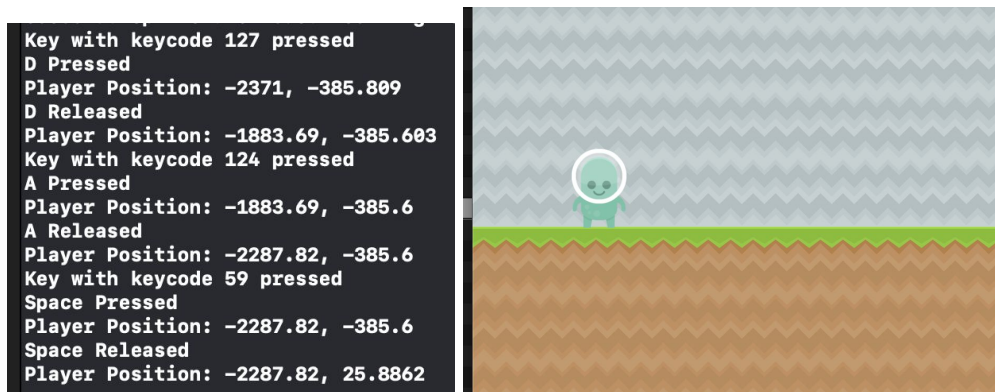
Integration Tests

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.

Object Test Results

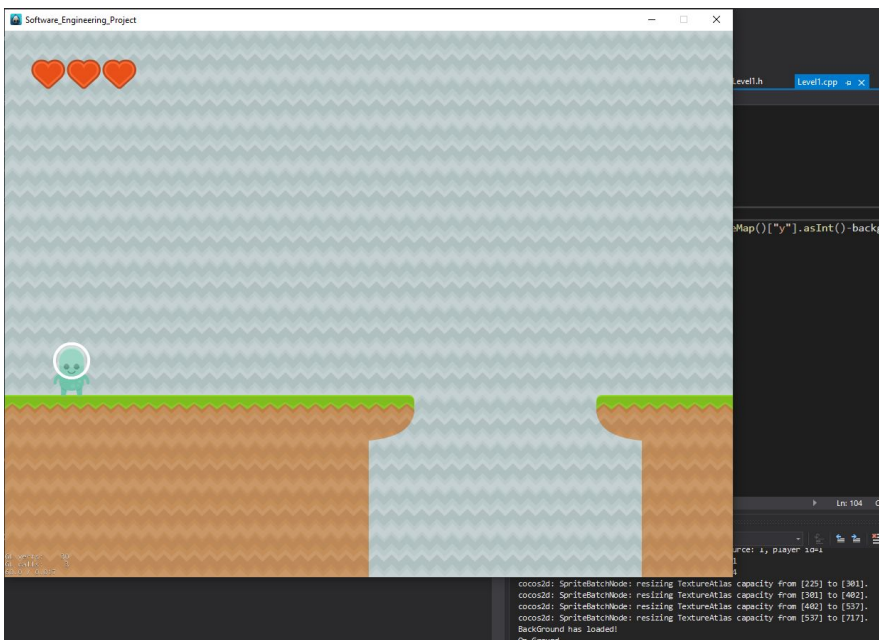
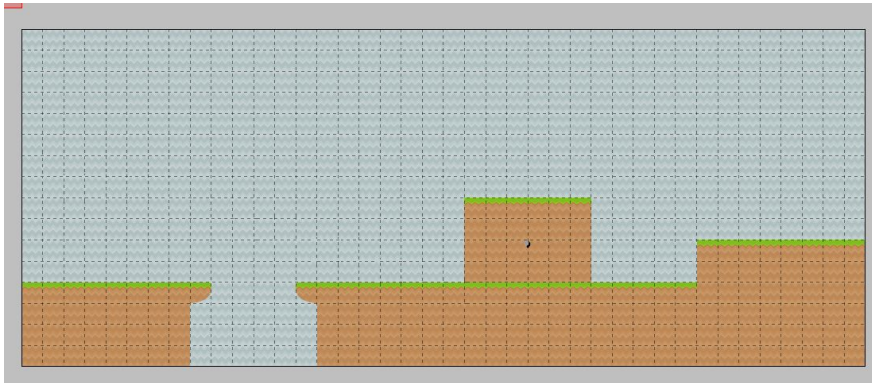


After those series of key movements, our player responded according and the picture on the right shows where he ended up.

Map Test

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

Map Test Results

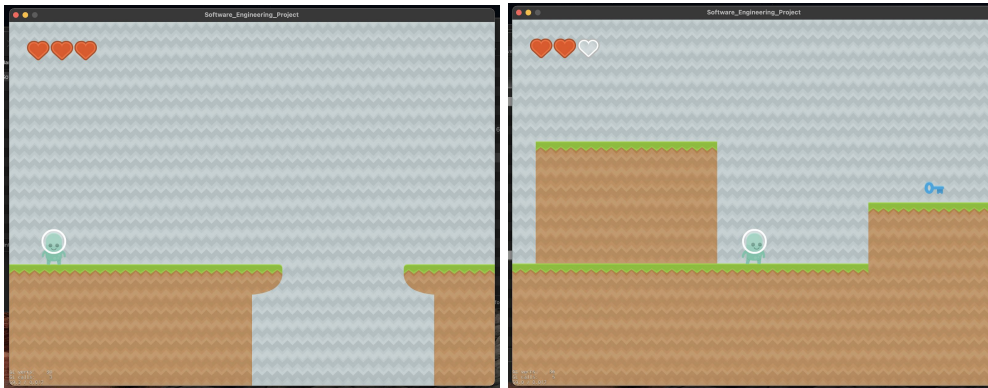


We have a background image that is illustrated in the first photo. The second verifies in the console that the background has actually loaded without any issues into the game.

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.

UI Test Results



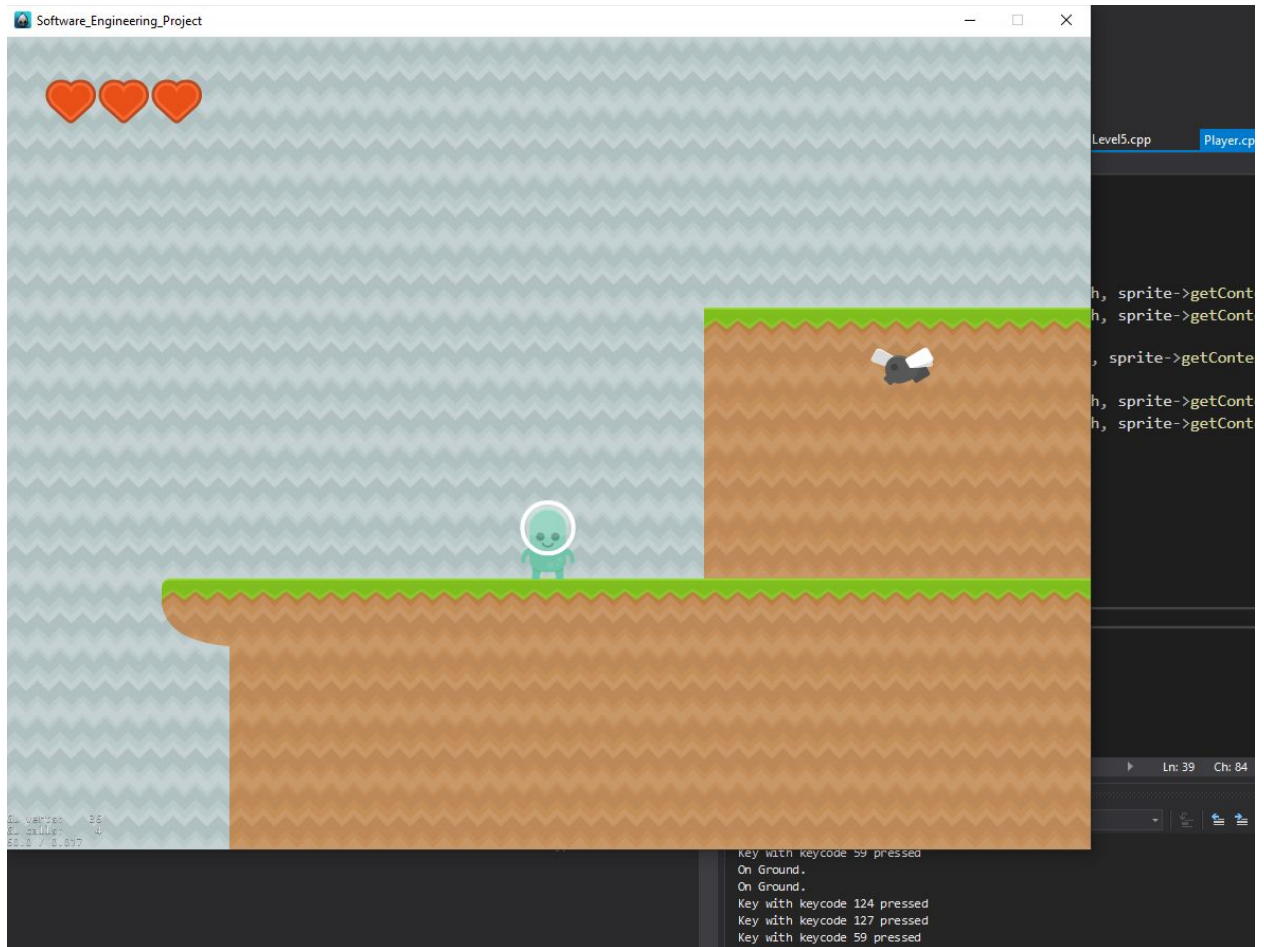
This shows that the player and the heart hud is displaying over the background. This shows when the player is hurt the hearts go down.

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.

Player Control Test Results

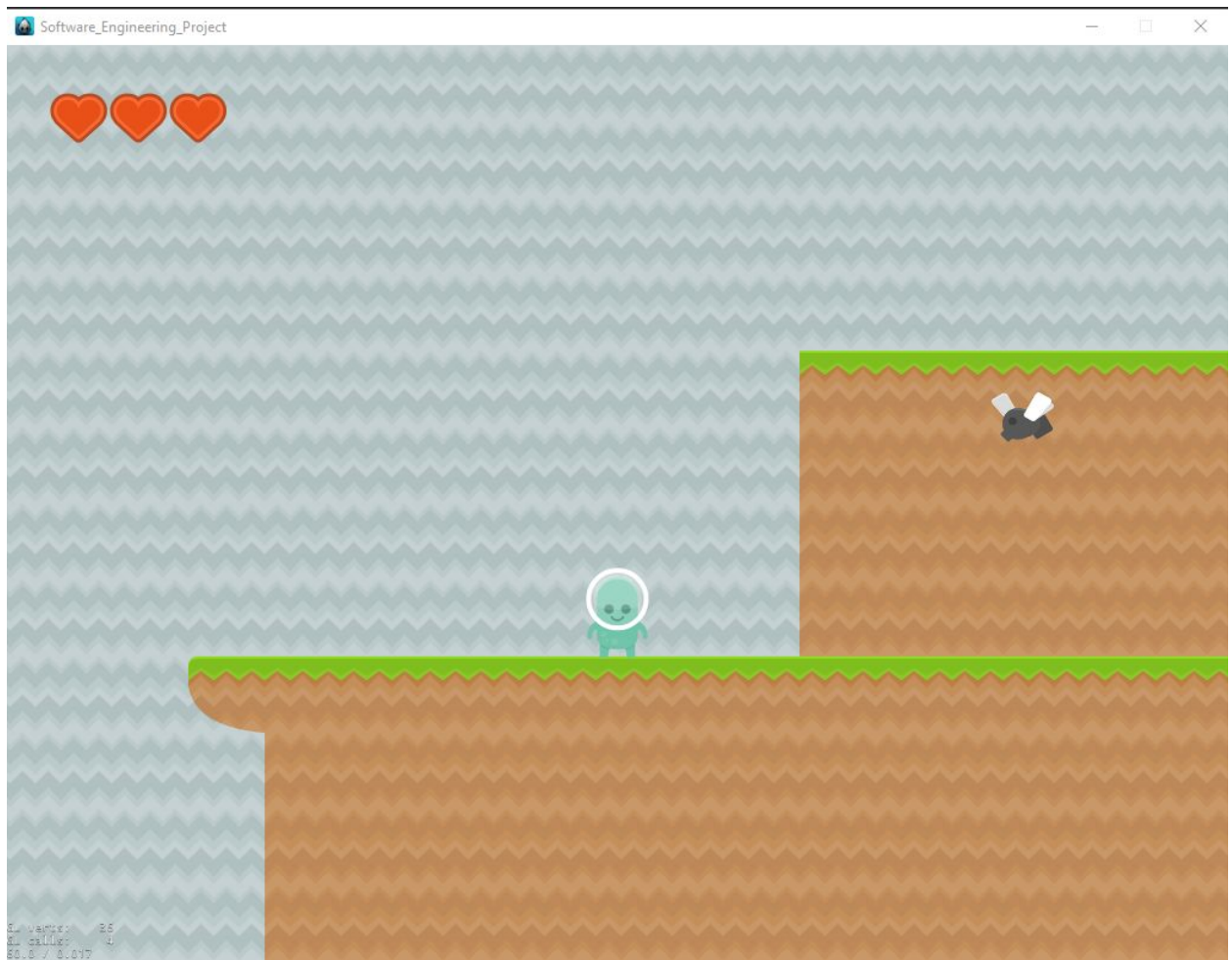


In the console you can see that the keys 124, 127, and 59 have been pressed. Those are our left, right, and jump controls respectively. The player responded to all actions.

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.

Graphics Rendering Test Results

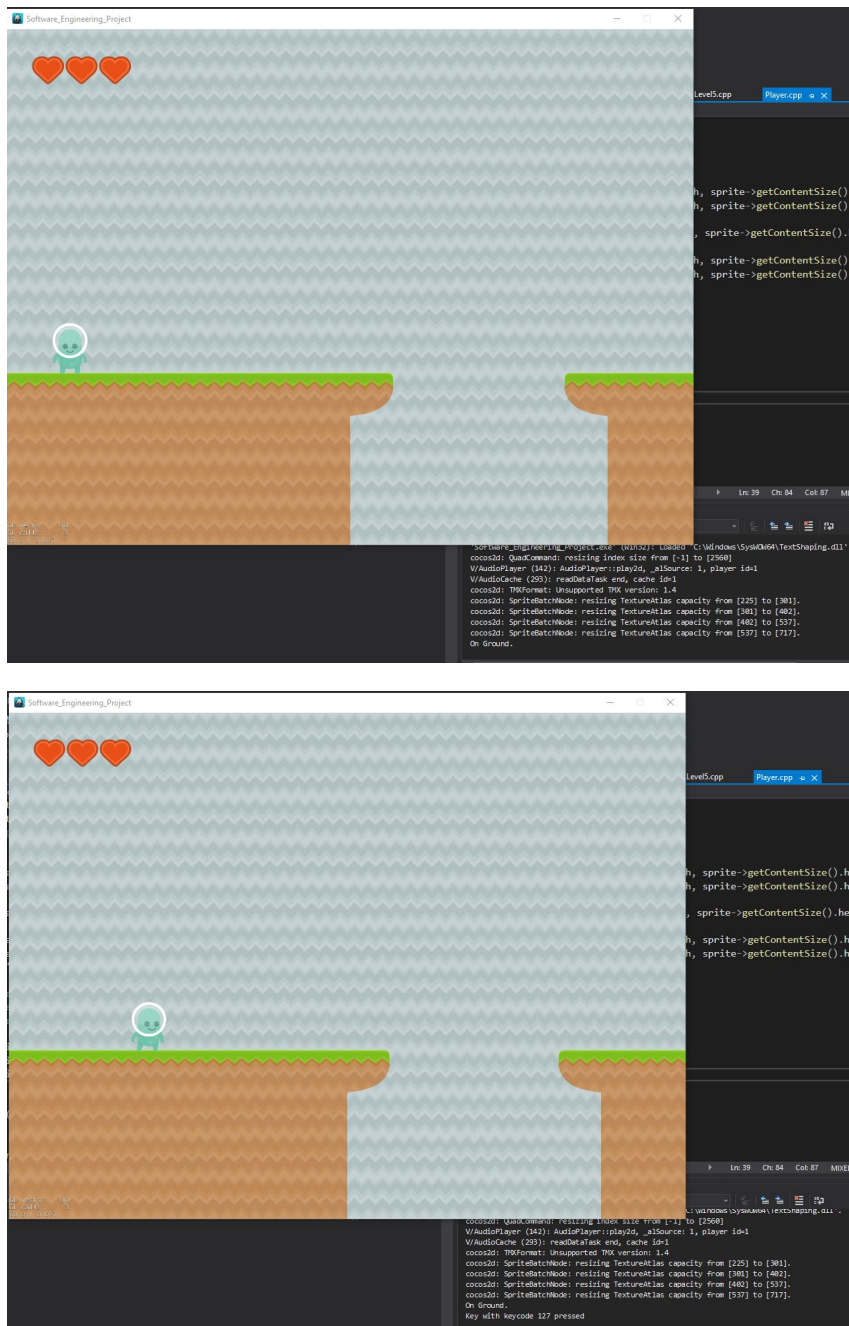


Graphics render to the screen according to our tile map positions that have have previously set. Everything renders in the correct position.

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.

Game Object Update Test Results

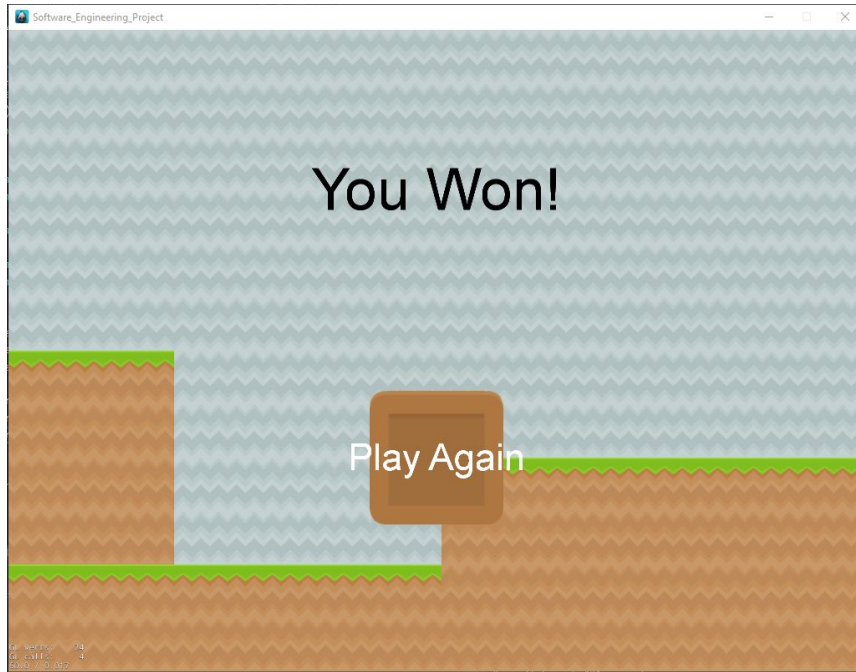


The first image is the first level just started, and the second image is I press the key to go right. It updated the position of the character, moved the camera, and adjusted the hearts in the top left. This verifies that objects update when input is given. In the console you can see a key was pressed.

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.

End Objective Test Results

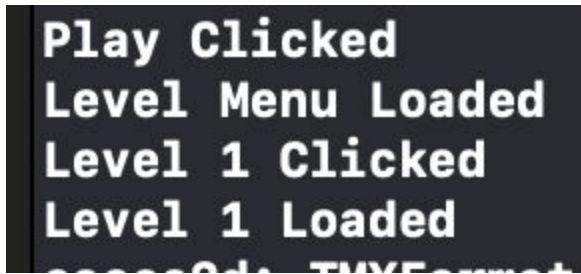


In Level 5, if you get to the star a Win screen show up for the player telling them they have beat the game.

Menu Test

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

Menu Test Results



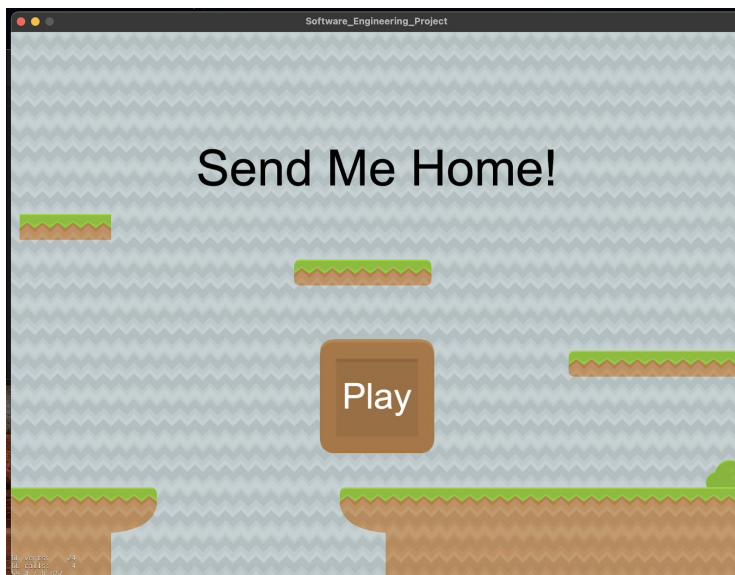
A sample of the buttons clicked, all of the buttons on our Main menu and our Level Menu work.

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.

Recovery Test Results

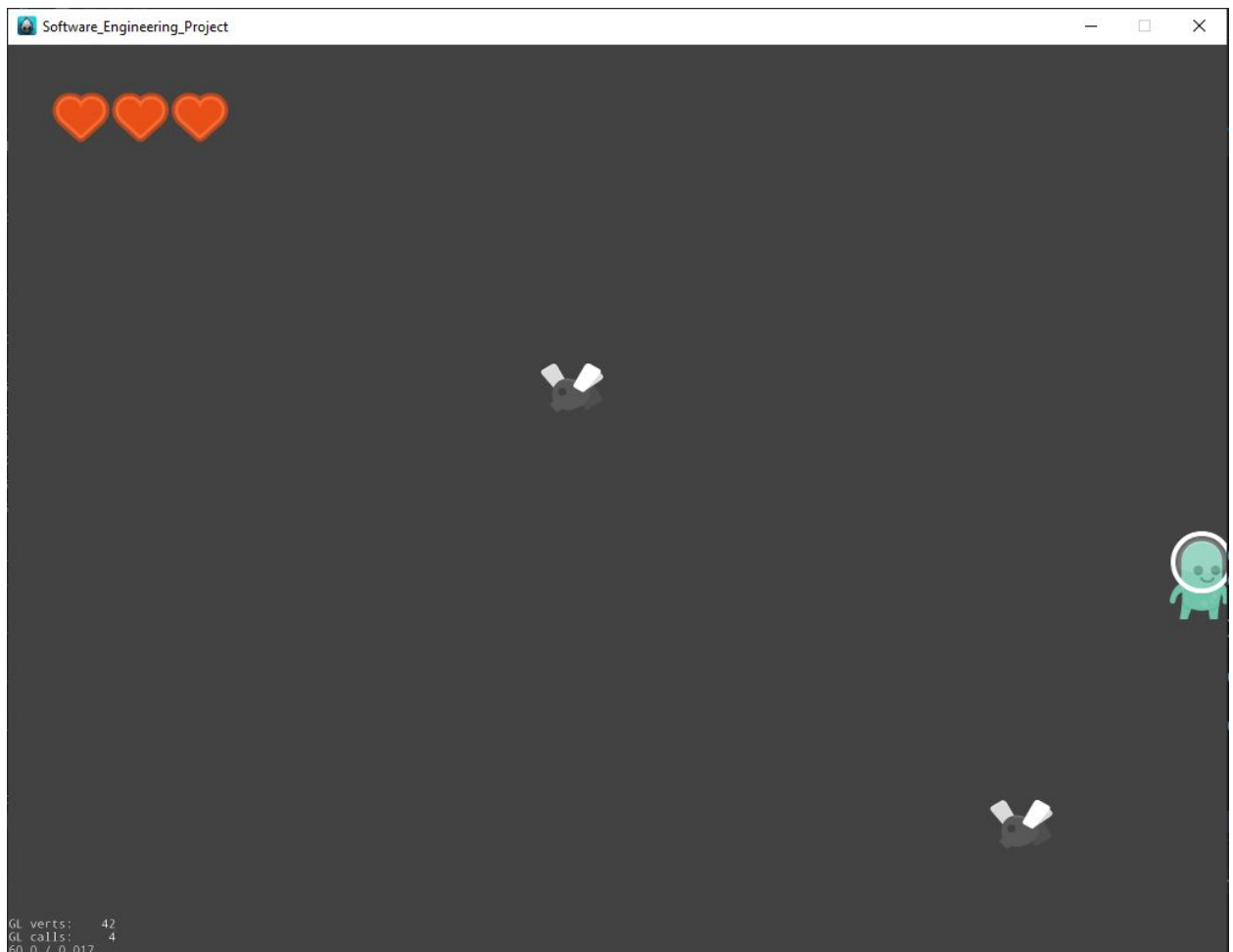


The game successfully launched after an induced crash without corrupting anything.

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.

Performance Test Results

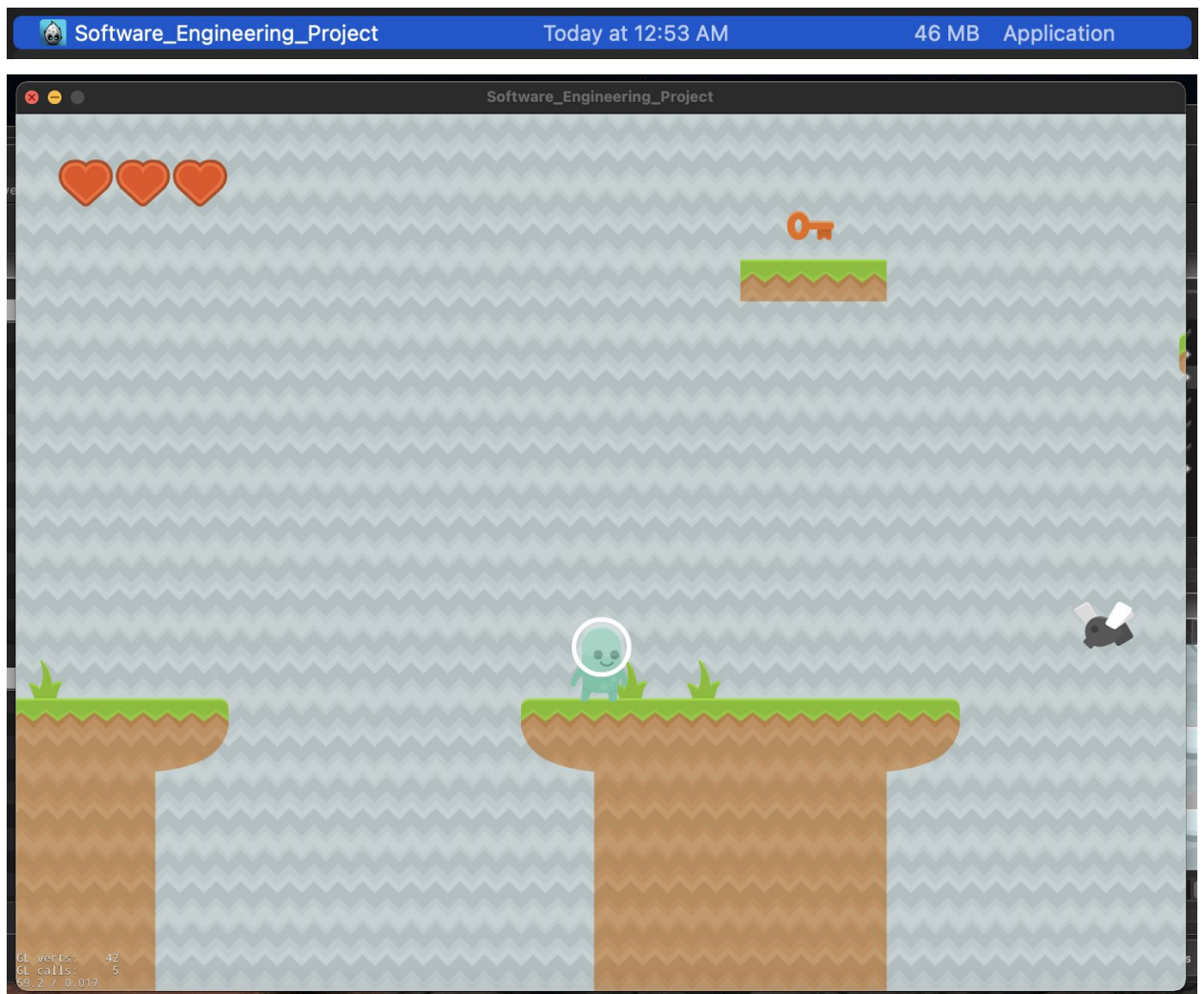


Level 5 is our level with the most objects moving around at a quicker pace and updating at a single time. In the bottom left part of the screenshot it shows our game running smoothly at 60 frames per second.

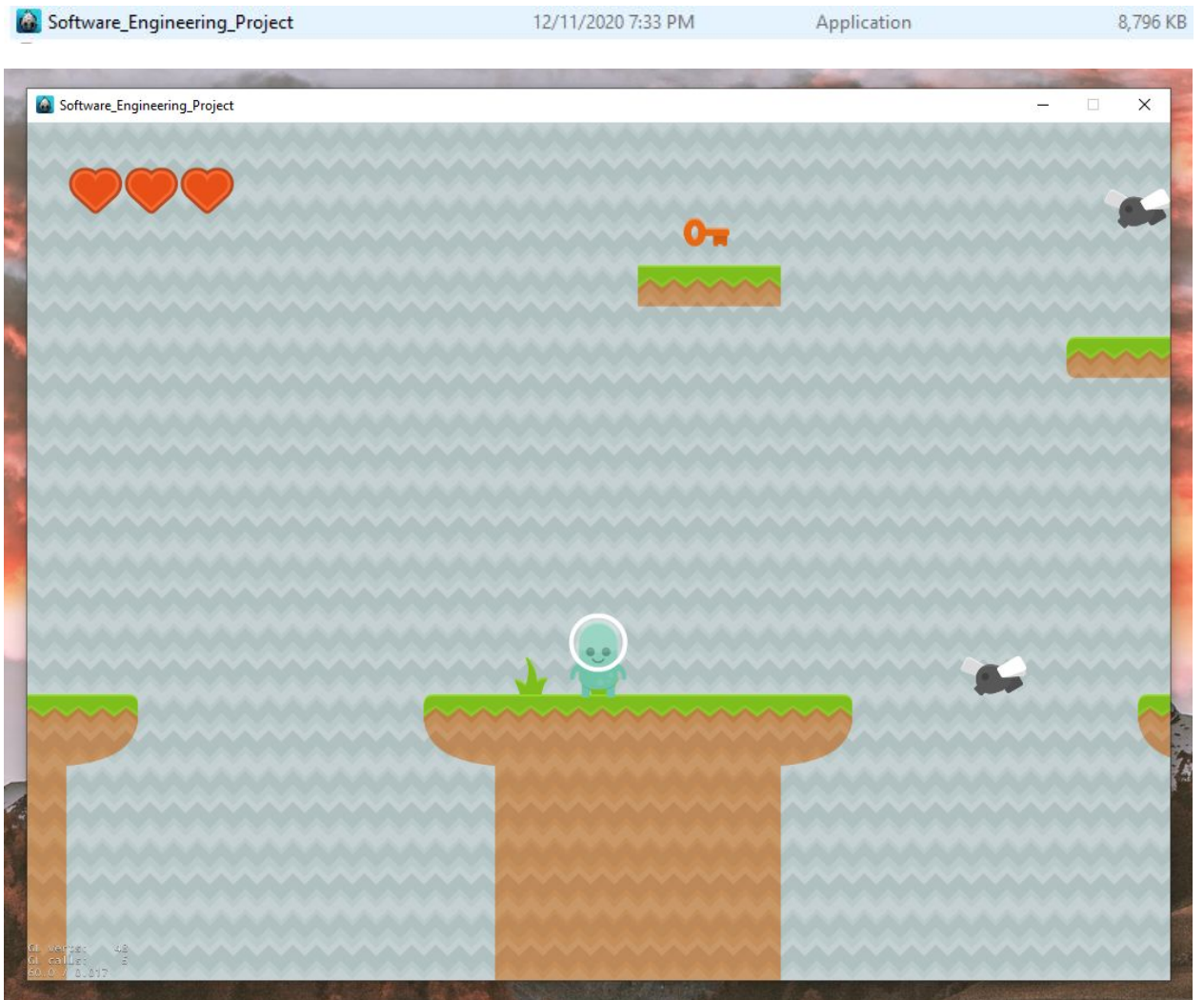
Deployment Test

This test will be to see if the game runs on windows and mac 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.

Deployment Test Results



Mac renders and shows correctly. User input works too because I was able to move to the middle platform. You can also see our fps in the bottom left.



Windows shows and renders correctly as well. User input worked correctly in order to get in the same spot on the level.