SE 3XA3: Test Report
Space Invader

L03,G07
Qianlin Chen and chenq84
Jiacheng Wu and wuj187
Tingyu Shi and shit19

April 12, 2022

Contents

List of Tables

Table 1: **Revision History**

Date	**Version**	**Notes**
April 11	All team members	Finish the document

This document specifies the complete testing process for Space Invader. It contains an evaluation of the project's functional and non-functional requirements that are defined in our SRS, the changes made due to testing result, and analysis of the traceability between requirements and modules.

1 Functional Requirements Evaluation

1.1 Model

Test 1	**Test-FR15-M1**
Description	Test if scores are added after bullets from aircraft hit monsters
Type	Unit test(functional, dynamic, automated)
Initial State	Bullet from aircraft hits one of the monsters.
Input	getScore() is called on Score object.
Expected	Player's score increase.
Output	Player's score increase.
Result	**PASS**

Test 2	**Test-FR16-M2**
Description	Test if life point decrements after aircraft get hit
Type	Unit test(functional, dynamic, automated)
Initial State	Monster's bullets hit aircraft
Input	getLives() is called on SpaceShip object
Expected	Aircraft's live decreases
Output	Aircraft's live decreases
Result	**PASS**

Test 3	**Test-FR16-M3**
Description	Test if life point increments after aircraft hits heart item
Type	Unit test(functional, dynamic, automated)
Initial State	Bullets from aircraft hit the health game item in the monster matrix
Input	getLives() is called on SpaceShip object
Expected	Aircraft's live increases by 1
Output	Aircraft's live increases by 1
Result	**PASS**

Test 4	**Test-FR25-M4**
Description	Test if isDead() return true after green monster get hit
Type	Unit test(functional, dynamic, automated)
Initial State	A green monster is hit by one bullet from the aircraft
Input	isDead() is called on Monster object
Expected	isDead() return true
Output	isDead() return true
Result	**PASS**

Test 5	**Test-FR26-M5**
Description	Test if isDead() return true after blue monster get hit twice
Type	Unit test(functional, dynamic, automated)
Initial State	A blue monster is hit by two bullets from the aircraft
Input	isDead() is called on Monster object
Expected	isDead() return true
Output	isDead() return true
Result	**PASS**

Test 6	**Test-FR27-M6**
Description	Test if isDead() return true after pink monster get hit three times
Type	Unit test(functional, dynamic, automated)
Initial State	A pink monster is hit by three bullets from the aircraft
Input	isDead() is called on Monster object
Expected	isDead() return true
Output	isDead() return true
Result	**PASS**

Test 7	**Test-FR29-M7**
Description	Test if area of obstacle decrease after it gets hit by bullet
Type	Unit test(functional, dynamic)
Initial State	One bullet hits an obstacle
Input	No input will be given in this test case
Expected	The area of obstacles should decrease by observation
Output	area of obstacles decrease in our observation
Result	**PASS**

Test 8	**Test-FR30-M8**
Description	Test if the game item GUI disappears only after aircraft hits it
Type	Unit test(functional, dynamic, manual)
Initial State	Game GUI presents one game item, one monster and one aircraft
Input	Monster shoots one bullet to the game item firstly and aircraft shoots one bullet to the game item secondly
Expected	Game item disappears only after aircraft hits it
Output	Game item disappears only after aircraft hits it
Result	**PASS**

Test 9	**Test-FR31-M9**
Description	Test if number of game items is less than 5 for each round
Type	Unit test(functional, dynamic, manual)
Initial State	Start a new round
Input	No input is given but testers need to record the number of game items
Expected	The number of game items of each round is less than 5
Output	The number of game items of each round is less than 5
Result	**PASS**

Test 10	**Test-FR32-M10**
Description	Test if 4 random monsters eliminated after the aircraft hits the bomb item
Type	Unit test(functional, dynamic, manual)
Initial State	GUI displays a monster matrix with a bomb game item in it
Input	Testers operate the aircraft and shoot a bullet from the aircraft to the bomb item
Expected	4 monsters eliminated from the monster matrix
Output	4 monsters eliminated from the monster matrix
Result	**PASS**

Test 11	**Test-FR33-M11**
Description	Test if number of bullets increment after the aircraft hits the ammo item
Type	Unit test(functional, dynamic, manual)
Initial State	GUI displays a monster matrix with a ammo game item in it
Input	Testers operate the aircraft and shoot a bullet from the aircraft to the ammo item
Expected	number of bullet from the aircraft increases by 1
Output	number of bullet from the aircraft increases by 1
Result	**PASS**

Test 12	**Test-FR34-M12**
Description	Test if number of life points increment after the aircraft hits the heart item
Type	Unit test(functional, dynamic, manual)
Initial State	GUI displays a monster matrix with a heart game item in it
Input	Testers operate the aircraft and shoot a bullet from the aircraft to the heart item
Expected	number of life point from the aircraft increases by 1
Output	number of life point from the aircraft increases by 1
Result	**PASS**

Test 13	**Test-FR35-M13**
Description	Test if game items occur randomly within monster matrix
Type	Unit test(functional, dynamic, manual)
Initial State	Several monster matricess are loaded
Input	No input will be given in this test case, but testers need to make a screenshot for each monster matrix
Expected	Game items occur randomly in different matrices
Output	Game items occur randomly in different matrices according to the matrices
Result	**PASS**

Test 14	**Test-FR40-M14**
Description	Test if the aircraft shoots bullets after pressing SPACE
Type	Unit test(functional, dynamic, manual)
Initial State	Set a new game
Input	Testers shoot bullets by press SPACE
Expected	Aircraft shoots a bullet
Output	Aircraft shoots a bullet
Result	**PASS**

1.2 View

Test 15	**Test-FR1-V1**
Description	Test if the GUI switch to choosing game mode after pressing any key
Type	Functional, Dynamic, Manual
Initial State	Welcome message has been displayed
Input	Testers enter any key
Expected	The GUI display a message and let testers to choose the play mode
Output	The GUI display a message and let testers to choose the play mode
Result	**PASS**

Test 16	**Test-FR2-V2**
Description	Test if GUI display instructions about monster type normally after tester chose the mode
Type	Functional, Dynamic, Manual
Initial State	Game mode selection UI hass been displayed
Input	Testers enter S or D to choose the play mode
Expected	GUI display the game instructions about monster types
Output	GUI display the game instructions about monster types
Result	**PASS**

Test 17	**Test-FR3-V3**
Description	Test if GUI display playing instructions normally after tester chose the mode
Type	Functional, Dynamic, Manual
Initial State	Game mode selection UI hass been displayed
Input	Testers enter S or D to choose the play mode
Expected	GUI display the game playing instructions
Output	GUI display the game playing instructions
Result	**PASS**

Test 18	**Test-FR4-V4**
Description	Test if GUI display scores in game correctly after testers enter the game
Type	Functional, Dynamic, Manual
Initial State	Monsters and game instructions have been displayed
Input	Testers press ENTER key
Expected	The game should display total score at the left up corner and the intial score should be 0
Output	The game should display total score at the left up corner and the intial score should be 0
Result	**PASS**

Test 19	**Test-FR5-V5**
Description	Test if GUI display health points in game correctly after testers enter the game
Type	Functional, Dynamic, Manual
Initial State	Monsters and game instructions have been displayed
Input	Testers press ENTER key
Expected	The game should display health points at the beginning of the game and the number of items should be 5
Output	The game should display 5 health points
Result	**PASS**

Test 20	**Test-FR6-V6**
Description	Test if the background of the game is displayed
Type	Functional, Dynamic, Manual
Initial State	Monsters and game instructions have been displayed
Input	Testers press ENTER key
Expected	The game should display the background
Output	The game display the background we set
Result	**PASS**

Test 21	**Test-FR7-V7**
Description	Test if the monster matrix of the game is displayed
Type	Functional, Dynamic, Manual
Initial State	Monsters and game instructions have been displayed
Input	Testers press ENTER key
Expected	The game should display the monster matrix
Output	The game display the monster matrix
Result	**PASS**

Test 22	**Test-FR8-V8**
Description	Test if the game items are displayed randomly
Type	Functional, Dynamic, Manual
Initial State	Monsters and game instructions have been displayed
Input	Testers press ENTER key and make a screenshot
Expected	The game items are displayed randomly in the screenshots
Output	The game items are displayed randomly in the screenshots
Result	**PASS**

Test 23	**Test-FR9-V9**
Description	Test if the game display number of aircraft correctly
Type	Functional, Dynamic, Manual
Initial State	Monsters and game instructions have been displayed
Input	Testers press S key and start the game
Expected	Game GUI display a single aircraft
Output	Game GUI display a single aircraft
Result	**PASS**

Test 24	**Test-FR9-V10**
Description	Test if the game display number of aircraft correctly
Type	Functional, Dynamic, Manual
Initial State	Monsters and game instructions have been displayed
Input	Testers press D key and start the game
Expected	Game GUI display two aircrafts
Output	Game GUI display aircrafts
Result	**PASS**

Test 25	**Test-FR10-V11**
Description	Test if the game display obstacles correctly
Type	Functional, Dynamic, Manual
Initial State	Monsters and game instructions have been displayed
Input	Testers press ENTER key and start the game
Expected	Game GUI display four obstacles between aircraft and monsters
Output	Game GUI display four obstacles between aircraft and monsters
Result	**PASS**

Test 26	**Test-FR11-V12**
Description	Test if the game display scores after the game
Type	Functional, Dynamic, Manual
Initial State	Testers finish the game
Input	No input is given
Expected	Scores are displayed
Output	Scores are displayed
Result	**PASS**

Test 27	**Test-FR12-V13**
Description	Test if the game display WIN after testers pass round 5
Type	Functional, Dynamic, Manual
Initial State	Testers aree in game round 5
Input	Testers pass game round 5
Expected	Game GUI display winning massage
Output	Game GUI display ”WIN!”
Result	**PASS**

Test 28	**Test-FR13-V14**
Description	Test if the game display fail message after testers lose
Type	Functional, Dynamic, Manual
Initial State	Testers are in any game round
Input	Testers' aircraft's hp goes to zero
Expected	Game GUI display fail massage
Output	Game GUI display ”FAIL!”
Result	**PASS**

Test 29	**Test-FR28-V16**
Description	Test if display of green monster is off after one hit
Type	Functional, Dynamic, Manual
Initial State	Testers start a game
Input	Testers shoot the green monster once
Expected	Display of green monster disappear
Output	Display of green monster disappear
Result	**PASS**

Test 30	**Test-FR28-V17**
Description	Test if display of blue monster is off after two hits
Type	Functional, Dynamic, Manual
Initial State	Testers start a game
Input	Testers shoot the blue monster twice
Expected	Display of blue monster disappear
Output	Display of blue monster disappear
Result	**PASS**

Test 31	**Test-FR28-V18**
Description	Test if display of pink monster is off after three hits
Type	Functional, Dynamic, Manual
Initial State	Testers start a game
Input	Testers shoot the pink monster three times
Expected	Display of pink monster disappear
Output	Display of pink monster disappear
Result	**PASS**

1.3 Controller

Test 32	**Test-FR18-C2**
Description	Test if the game contain 5 rounds as designed
Type	Functional, Dynamic, Manual
Initial State	Game instructions and introduction to monsters have been displayed
Input	Testers try to pass the whole game and count the number of game rounds
Expected	number of game rounds is 5
Output	number of game rounds is 5
Result	**PASS**

Test 33	**Test-FR23-C7**
Description	Test if the movement of monster matrix is correct
Type	Functional, Dynamic, Manual
Initial State	Testers are playing in different game rounds
Input	No input
Expected	The movement of the monster matrix is east-south-west
Output	The movement of the monster matrix is east-south-west
Result	**PASS**

Test 34	**Test-FR24-C8**
Description	Test if monsters in matrix shoot randomly in different rounds
Initial State	Testers are in different game rounds
Input	No input
Expected	Monsters in matrix shoot randomly in different rounds
Output	Monsters in matrix shoot randomly in different rounds
Result	**PASS**

Test 35	**Test-FR24-C9**
Description	Test if users can enter the game successfully
Initial State	Game instructions and introduction to monsters have been displayed
Input	Testers press ENTER key
Expected	GUI shows the game
Output	GUI shows the game
Result	**PASS**

Test 36	**Test-FR37-C10**
Description	Test if users are able to exit the game
Initial State	Testers are in any game round
Input	Testers press cross icon
Expected	Game exited
Output	Game exited
Result	**PASS**

Test 37	**Test-FR38-C11**
Description	Test if aircraft can be moved properly
Initial State	Aircraft is in a stationary state with a certain position
Input	Testers press keyboard to move the aircraft
Expected	Aircraft move to a correct position
Output	Aircraft moves the a position which is calculated correctly
Result	**PASS**

2 Nonfunctional Requirements Evaluation

2.1 Look and Feel Testing

Test 38:
Test-NFR1-LF1
Type: Dynamic, Automated, Functional
Initial State: Game started.
Input/Condition: Testers use external tool to measure the FPS.
Output/Result: FPS value should always be greater than 30.
How test will be performed: Testers can record the FPS of the game every 10 seconds and measure FPS for 2 minutes. After that, testers can check if all the values recorded are greater than 30.
Result: PASS

Test 39:
Test-NFR2-LF2
Type: Dynamic, Manual, Functional
Initial State: Game started.
Input/Condition: Testers record players' thoughts about the game display.
Output/Result: Over 80% of people can recognize game elements clearly.
How test will be performed: Testing team can invite 10 people to play the game and 8 out of 10 random persons should recognize all the elements clearly.
Result: PASS

Test: 40
Test-NFR3-LF3
Type: functional, dynamic, manual
Initial State: Game started.
Input/Condition: No inputs will be given for this test case. Players will be asked about their thoughts about the game minimalism.
Output/Result: Over 80% of players think the game follows the style of minimalism.
How test will be performed: Invite 10 random persons to play the game and then record their thoughts about the game minimalism.
Result: PASS

Test 41:
Test-NFR4-LF4
Type: functional, dynamic, manual
Initial State: Game started.
Input/Condition: No inputs will be given for this test case. Players will be asked about their thoughts about the game mood.
Output/Result: Over 80% of players think the game mood is intense.
How test will be performed: Invite 10 random persons to play the game and then record their thoughts about the game mood.
Result: PASS

2.2 Usability and Humanity Testing

Test 42:
Test-NFR5-UH1
Type: Dynamic, Manual, Functional
Initial State: Game started.
Input/Condition: No inputs will be given for this test case. Child players will be asked about their thoughts of the ease of the game.
Output/Result: Over 80% of child players think the game is easy.
How test will be performed: Testing team can invite 10 child players to play the game and record their thoughts about the ease of the game. 8 of 10 children should think the game is easy.
Result: PASS

Test 43:
Test-NFR6-UH2
Type: Dynamic, Manual, Functional
Initial State: Game started.
Input/Condition: No inputs will be given for this test case. Testing team should record the learning time of players.
Output/Result: Over 80% of players should be able to play the game with 5 minutes of less learning time.
How test will be performed: Testing team can invite 10 players to play the game and record their learning time. 8 of 10 players should be able to play the game with 5 minutes or less learning time.
Result: PASS

Test 44:
Test-NFR7-UH3
Type: Static, Functional, Manual
Initial State: Game instructions have been displayed.
Input/Condition: No inputs will be given for this test case. Testing team should record the time needed for players to understand game rules.
Output/Result: All players should understand game rules within 10 minutes.
How test will be performed: Testing team can invite 10 players to read game instructions and record the time needed for players to fully understand the game instructions. All game players should be able to understand game instructions within 10 minutes.
Result: PASS

2.3 Performance Testing

Test 45:
Test-NFR8/9-PE1
Type: Dynamic, Functional, Automated
Initial state: Game not started.
Input/Condition: Using external tools to record the response time for each user input.
Output/Result: All the response time should be less than 1 second.
How test will be performed: Testing team starts to play the game and tries to pass all game rounds.

During this process, testing team should record the response time for each user input, all the response time should be less than 1 second.
Result: PASS

Test 46:
Test-NFR10-PE2
Type: Functional, Dynamic, Automated
Initial State: Game not started.
Input/Condition: Use an automated program to start the game at random times.
Output/Result; The game should start successfully each time.
How test will be performed: Use the automated program to start the game 100 times a day and test like this for 5 days. The game should start properly 500 times in total.
Result: PASS

Test 47:
Test-NFR11-PE3
Type: Functional, Dynamic, Manual
Initial State: Game mode is chosen to be double-player mode.
Input/Condition: Players start to play the game.
Output/Result: Games should run properly for a least 2 hours.
How test will be performed: Testing team can invite 10 players to play in double-player mode. All 5 games should be able to run properly for at least 2 hours.
Result: PASS

2.4 Operational and Environmental Testing

Test 48:
Test-NFR12/13-OE1
Type: Dynamic, Functional, Automated
Initial State: Game installed on Windows, Linus and MacOS.
Input/Condition: Using the automated program to run games on three different platforms.
Output/Result: Game can run properly over 90% of the time on three different platforms.
How the test will be performed: Testing team can use the automated program to run the game on three different platforms 100 times. The game should be able to run properly over 90 times.
Result: PASS

Test 49
Test-NFR15-OE3
Type: Functional, Dynamic, Manual
Initial State: Game not installed.
Input/Condition: No inputs will be given for this test case. Testing team will let players to install the game and record the installation process.
Output/Result: Players should be able to install without any problems.
How the test will be performed: Testing team can invite 10 people and let them to install the game. All 10 people should not have any problems installing the game.

Result: PASS

2.5 Maintainability and Support Testing

Test 50:
Test-NRF16-MS1
Type: Manual
Initial state: A new feature is decided to add to the game. (The feature here means the feature may be added after the game is released).
Input/Condition: Development team starts to prepare MIS and coding. After that, development team tests the newly added feature.
Output/Result:The process mentioned above (Writing MIS, coding and testing) should be completed within two weeks.
How test will be performed: After the game is released, project manager should come up with a new feature and let development team to implement this new feature. As a result, we can time how long this new feature can be implemented.
Result: PASS

Test 51:
Test-NRF17-MS2
Type: Manual, Dynamic, Automated
Initial State: Testers open source code and terminal.
Input/Condition: Testers type command in terminal in order to generate doxygen files.
Output/Result: Doxygen files should be generated successfully and all contents are correct.
How test will be performed: Testers can try to generate doxygen files in terminal. After generating all the doxygen files, testers should check if doxygen files contents can match the source code. **Result: PASS**

Test 52:
Test-NFR18-MS3
Type: Manual
Initial state: Development team has already read all the previous messages from players.
Input/Condition: Testing team leaves a message to the development team.
Output/Result: Testing team should receive a response from the development team.
How test will be performed: Testing team leaves an advice to the development team in its project repo. After that, testing team will check if they can receive the response from the development team. **Result: PASS**

Test 53:
Test-NFR19-MS4
Type: Dynamic, Manual
Initial state: Three computers with three different operating systems(Windows, Linux, MacOS) do not have our game installed.
Input/Condition: Testing team tries to install our game in three computers.
Output/Result: Three installations are successful and our game can run properly on three operating

systems.
How test will be performed: Testing team tries to install and run game in three different operating systems and use a checklist to show that game can run properly in three operating systems.
Result: PASS

2.6 Security Testing

Test 54:
Test-NFR20-SE1
Type: Manual, Dynamic
Initial State: In double-player game mode.
Input/Condition:Two testers control two aircraft. ”Controlling two aircraft” means moving them left and right and shooting bullets from two aircraft.
Output/Result: A specific aircraft can only be controlled by the corresponding player. For example, aircraft1 should not be controlled by tester2 and aircraft2 should not be controlled by tester1.
How test will be performed: Two testers start a new game and chooses double-player mode. Tester1 controls aircraft1 and tester2 controls aircraft2. The following three scenarios should be tested:

- Aircraft1 moves and aircraft2 stays still.
- Aircraft2 moves and aircraft1 stays still.
- Two aircraft move concurrently.

For all three scenarios mentioned above, two aircraft should only be controlled by the correspond tester.
Result: PASS

2.7 Cultural and Political Testing

Test 55:
Test-NFR22-CP1
Type: Manual, Dynamic
Initial state: Game not started.
Input/Condition: Testing team starts to play game and record the process of playing.
Output/Result: There should not be any offensive messages or images in the recording.
How test will be performed: Testing team starts to play the game and record the screen. After that, testing team will analysis the recording to ensure that there are no offensive messages or pictures.
Result: PASS

2.8 Legal Testing

Test 56:
Test-NF23-LE1 Type: Manual, Dynamic

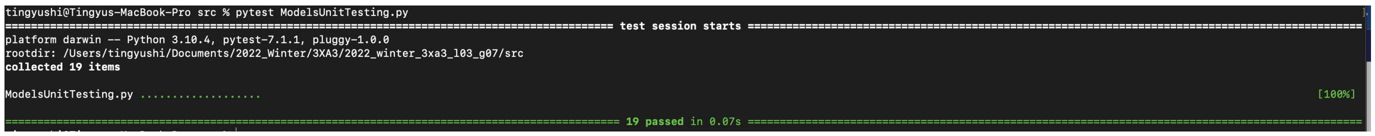
Initial state: Game not started.
Input/Condition: Testing team starts to play game and record the process of playing.
Output/Result: There should not be any illegal things in the recording.
How test will be performed: Testing team starts to play the game and record the screen. After that, testing team will analysis the recording to ensure that everything is legal. Maybe testing team can invite domain experts.
Result: PASS

3 Comparison to Existing Implementation

The original project doesn't have any test plan and test report. Therefore, we cannot compare our test report to the original one.

4 Unit Testing

We used PyTest to conduct the unit test on our project. Since a lot of contents of our project are GUI based. So we only apply the unit testing to our models.
There are 19 unit test cases in total to assess the correctness of game models. The following is the result screenshot:



```
tingyushi@Tingyus-MacBook-Pro src % pytest ModelsUnitTesting.py
============================== test session starts ==============================
platform darwin -- Python 3.10.4, pytest-7.1.1, pluggy-1.0.0
rootdir: /Users/tingyushi/Documents/2022_Winter/3XA3/2022_winter_3xa3_l03_g07/src
collected 19 items

ModelsUnitTesting.py ...................                                   [100%]

============================== 19 passed in 0.07s ===============================
```

5 Changes Due to Testing

Through the processes of project, we used functional and not functional tests towards our project. We not only used it to spot errors and fix them, but we also used the tests to improve our project as well. There are a lot of changes we made after doing different aspects of tests.

5.1 Play Testing

In the processes of our testing, we mostly used Integration testing to play the game as a whole since it's the most efficient way to test a video game. After playing this game, we found the game is a little boring and that's why we added three more game items in the end to put more elements in this game

5.2 Difficulty Testing

During the testing, we focused on testing the difficulty of the game. At the very start, the game is too easy since the speed of enermies' was too slow and speed of bullets from aircraft is too fast. After that, we adjust the speed of bullets of aircraft and monsters to make the game more challenging to play.

5.3 Music Testing

The original project has music with it. However, during our play tests, we found that the music is too distracting and annoyinng, so we decided to remove the music from it.

6 Automated Testing

Since our project is a game, it requires input from testers all the time. Most of our testing is manual testing. However, we used automated testing on our game models.

7 Trace to Requirements

Table 2: Model Traceability Matrix 1

Tests/Requirement	FR15	FR16	FR25	FR26	FR27	FR29
Test-FR15-M1	X					
Test-FR16-M2		X				
Test-FR16-M3		X				
Test-FR25-M4			X			
Test-FR26-M5				X		
Test-FR27-M6					X	
Test-FR29-M7						X

Table 3: Model Traceability Matrix 2

Tests/Requirement	FR30	FR31	FR32	FR33	FR34	FR35	FR40
Test-FR30-M8	X						
Test-FR31-M9		X					
Test-FR32-M10			X				
Test-FR33-M11				X			
Test-FR34-M12					X		
Test-FR35-M13						X	
Test-FR40-M14							X

7.1 View Traceability Matrices

Table 4: View Traceability Matrix 1

Tests/Requirement	FR1	FR2	FR3	FR4	FR5	FR6	FR7	FR8	FR9
Test-FR1-V1	X								
Test-FR2-V2		X							
Test-FR3-V3			X						
Test-FR4-V4				X					
Test-FR5-V5					X				
Test-FR6-V6						X			
Test-FR7-V7							X		
Test-FR8-V8								X	

Table 5: View Traceability Matrix 2

TestCase\Requirements	FR9	FR10	FR11	FR12	FR13	~~FR14~~	FR28
Test-FR9-V9	X						
Test-FR9-V10	X						
Test-FR10-V11		X					
Test-FR11-V12			X				
Test-FR12-V13				X			
Test-FR13-V14					X		
~~Test-FR14-V15~~						~~X~~	
Test-FR28-V16							X
Test-FR28-V17							X
Test-FR28-V18							X

7.2 Control Traceability Matrices

Table 6: Control Traceability Matrix 1

Tests/Requirement	FR17	FR18	FR19	FR20	FR21	FR22
Test-FR17-C1	X					
Test-FR18-C2		X				
Test-FR19-C3			X			
Test-FR20-C4				X		
Test-FR21-C5					X	
Test-FR22-C6						X

For the above table, please only focus on Test-FR18-C2. Other test cases will not be considered after the revision.

Table 7: Control Traceability Matrix 2

Tests/Requirement	FR23	FR24	FR36	FR37	FR38	~~FR38~~
Test-FR23-C7	X					
Test-FR24-C8		X				
Test-FR36-C9			X			
Test-FR37-C10				X		
Test-FR38-C11					X	
~~Test-FR38-C12~~						~~X~~

7.3 Nonfunctional Req Test Matrices

Table 8: Nonfunctional Req Test Matrix 1

Tests/Requirement	NFR1	NFR2	NFR3	NFR4	NFR5	NFR6	NFR7
Test-NFR1-LF1	X						
Test-NFR2-LF2		X					
Test-NFR3-LF3			X				
Test-NFR4-LF4				X			
Test-NFR5-UH1					X		
Test-NFR6-UH2						X	
Test-NFR7-UH3							X

Table 9: Nonfunctional Req Test Matrix 2

Tests/Requirement	NFR8	NFR9	NFR10	NFR11	NFR12	NFR13	~~NFR14~~	NFR15	NFR16
Test-NFR8/9-PE1	X	X							
Test-NFR10-PE2			X						
Test-NFR11-PE3				X					
Test-NFR12/13-OE1					X	X			
~~Test-NFR14-OE2~~							~~X~~		
Test-NFR15-OE3								X	
Test-NFR16-MS1									X

Table 10: Nonfunctional Req Test Matrix 3

Tests/Requirement	NFR17	NFR18	NFR19	NFR20	~~NFR21~~	NFR22	NFR23	~~NFR24~~
Test-NFR17-MS2	X							
Test-NFR18-MS3		X						
Test-NFR19-MS4			X					
Test-NFR20-SE1				X				
~~Test-NFR21-SE2~~					~~X~~			
Test-NFR22-CP1						X		
Test-NFR23-LE1							X	
~~Test-NFR24-HS1~~								~~X~~

8 Trace to Modules

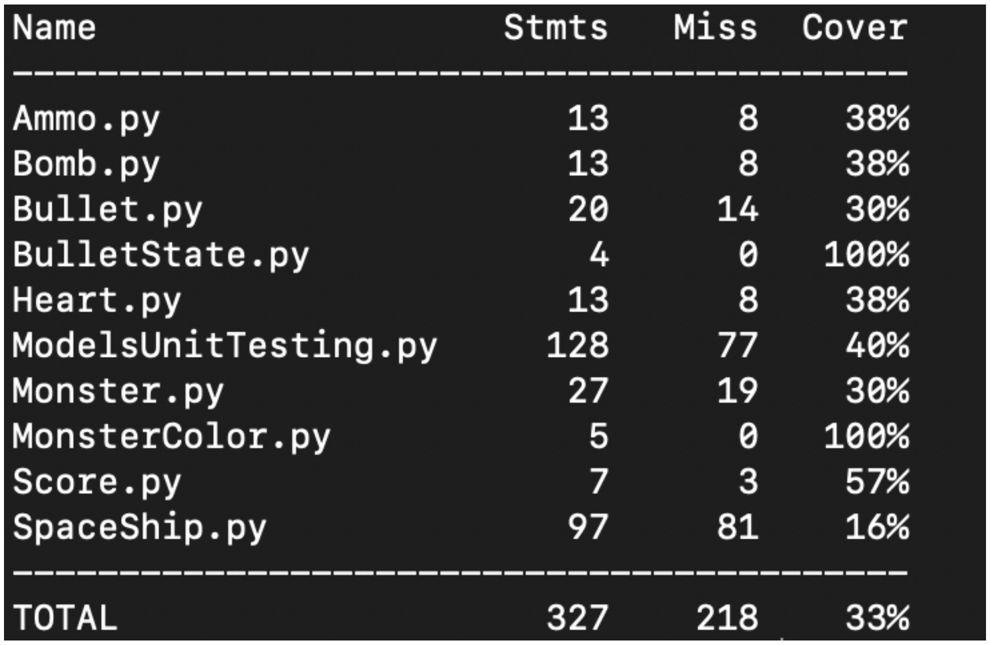
Table 11: Trace to Modules Table

Test number	Module
Test1	M1, M2, M17, M14, M18
Test2	M17, M15, M1, M2
Test3	M1, M2, M23, M15, M17
Test4	M14
Test5	M14
Test6	M14
Test7	M17, M31, M19, M1, M2
Test8	M6
Test9	M16
Test10	M24, M1, M2, M16, M14
Test11	M22, M15, M1, M2, M17
Test12	M23, M15, M1, M2
Test13	M16
Test14	M15, M1, M2
Test15	M12, M1
Test16	M27, M1
Test17	M27, M1
Test18	M8
Test19	M10
Test20	M13
Test21	M6
Test22	M6, M14
Test23	M5, M3
Test24	M5, M3
Test25	M9
Test26	M8
Test27	M3, M30
Test28	M29, M3
Test29	M6, M1, M2
Test30	M6, M1, M2
Test31	M6, M1, M2
Test32	M1, M2, M16
Test33	M6, M16
Test34	M16
Test35	M3
Test36	M3
Test37	M15

The reason why we only include functional requirements here is because for the non-functional requirements, it's not applicable to trace them to specific modules since some of them are tested by interviews and tested subjectively.

9 Code Coverage Metrics

The code coverage of this project is only 33 percent because at most of the time we use integration test to test the whole game. For controller and view modules, most of the lines cannot be reached since they depend on certain input from the user. The following is the screenshot of our code coverage:



```
Name                     Stmts   Miss  Cover
--------------------------------------------
Ammo.py                     13      8    38%
Bomb.py                     13      8    38%
Bullet.py                   20     14    30%
BulletState.py               4      0   100%
Heart.py                    13      8    38%
ModelsUnitTesting.py       128     77    40%
Monster.py                  27     19    30%
MonsterColor.py              5      0   100%
Score.py                     7      3    57%
SpaceShip.py                97     81    16%
--------------------------------------------
TOTAL                      327    218    33%
```