SE 3XA3: Test Report
CraftMaster

Group 307, 3 Craftsmen
Hongqing Cao 400053625
Sida Wang 400072157
Weidong Yang 400065354

April 7, 2020

Contents

List of Tables

List of Figures

Date	**Version**	**Notes**
April 2	1.0	General content added
April 6	2.0	Completed for Rev1

Table 1: **Revision History**

1 Overview

This document specifies the result of the testing specifications that are proposed in the **Test Plan** of CraftMaster. The format of this document follows the template provided by Dr.Bokhari and Thien Trandinh.

1.1 Acronyms, Abbreviations, and Symbols

Abbreviation	**Definition**
SRS	Software Requirement Specification
OS	Operating System
GUI	Graphical User Interface

Table 2: **Table of Abbreviations**

Term	**Definition**
Python	The programming language that is used for the developement of this project
Pyglet	A Python library for the design of graphical user interface
Pytest	A Python library and framework for unit testing
Sandbox Games	A type of game that allows player to create, modify, and destroy the environment
3D Game	A game in three dimensions

Table 3: **Table of Definitions**

2 Functional Requirements Evaluation

Test Case No.	**Test Name**	**Initial State**	**Input**	**Expected Output**	**Pass or Fail**
TFR1	Test Game Start	The software game is installed and ready to execute.	A cursor placement on the game icon and a double-click on the left mouse key.	The program opens the GUI frame with main menu rendered.	**Pass**
TFR2	Test Main Menu and Sub-Menus directions	The GUI is opened and the main menu is rendered.	Mouse clicks on the buttons on the main menu that direct to sub-menus and buttons on the sub-menus that direct to the main menu.	The program will go to sub-menus from the main menu and go to the main menu from sub-menus.	**Pass**
TFR3	Test New Game	The GUI is opened and the game menu is rendered.	Mouse clicks on the “start new game” button.	The program will load a new game scene, initialize the character, and place the crosshair at the center of the window.	**Pass**
TFR4	Test Saved Game	The GUI is opened and the game menu is rendered. There is a saved game scene.	A mouse clicks on the ”load game” button and a mouse clicks on a game saving spot (Game One or Game Two).	The program will load a saved game scene, initialize the character, and place the crosshair at the center of the window.	**Pass**

TFR5	Test Crosshair Position Stability	The software game GUI is opened and the crosshair is placed at the center of the GUI window.	A sequence of movements of the character (controlled by keyboard inputs) followed by a sequence of block operations (controlled by keyboard and mouse inputs).	The crosshair keeps in position (center of GUI window).	**Pass**
TFR6	Test Character Direction	The software game GUI is opened and the game scene and character are loaded.	A movement of the mouse.	The character changes its view direction.	**Pass**
TFR7	Test Block Outline	The software game GUI is opened and the game scene and character are loaded. A block is near the character.	A movement of the mouse to aim the crosshair at the block.	The GUI shows the outline of the block(indicating the block is being aimed at).	**Pass**
TFR8	Test Block Removal	The software game GUI is opened and the game scene and character are loaded. A block is near the character and the player has already aimed the crosshair at the block.	A left-click on the mouse.	The block is being removed from the window and the program plays a sound effect to notify.	**Pass**

TFR9	Test Block Build	The software game GUI is opened and the game scene and character are loaded. A block is near the character and the player has already aimed the crosshair at the block(the crosshair could be pointed to top, bottom, or any side).	A right-click on the mouse.	The new block is being built at the position that is next to the surface of the existing block that the crosshair was pointed to and the program plays a sound effect to notify.	**Pass**
TFR10	Test Background Music	The software game is installed and ready to execute.	A cursor placement on the game icon and a double-click on the left mouse key.	The program plays the background music of the game.	**Pass**
TFR11	Test Marble Block Operations	A crosshair is placed on a marble block.	A left-click on the mouse.	The marble is not removed.	**Pass**
TFR12	Test Day-Night Mode	The GUI is at the setting menu or at the ESC menu.	A left-click on the Day-Night Mode switch.	The Day-Night mode changes.	**Pass**
TFR13	Test Game Quit	The GUI is at the main menu.	A left-click on “quit” button on the main menu.	The program terminates and the GUI closes.	**Pass**
TFR14	Test Game Save	The GUI is at ESC menu.	A left-click on game saving and saving spot selection button.	The json file appears in the game folder.	**Pass**
TFR15	Test ESC menu to Game menu	The GUI is at ESC menu.	A left-click on game saving and saving spot selection button.	The GUI returns to the game menu.	**Pass**

TFR16	Test Forward Movement	The software game GUI is opened and the game scene and character are loaded. The character is free to move forward(no blocks are close to and at the front of the character).	A click on the “W” key on the keyboard.	The character moves forward.	**Pass**
TFR17	Test Left Movement	The software game GUI is opened and the game scene and character are loaded. The character is free to move to the left(no blocks are close to and on the left of the character).	A click on the “A” key on the keyboard.	The character moves to the left.	**Pass**
TFR18	Test Backward Movement	The software game GUI is opened and the game scene and character are loaded. The character is free to move backward(no blocks are close to and at the back of the character).	A click on the “S” key on the keyboard.	The character moves backward.	**Pass**

TFR19	Test Right Movement	The software game GUI is opened and the game scene and character are loaded. The character is free to move to the right (no blocks are close to and on the right of the character).	A click on the "D" key on the keyboard.	The character moves to the right.	**Pass**
TFR20	Test Jump Action	The software game GUI is opened and the game scene and character are loaded.	A click on the space key on the keyboard.	The character jumps once.	**Pass**
TFR21	Test Flying Mode	The software game GUI is opened and the game scene and character are loaded.	A click on the tab key, followed a sequence of clicks on the "W" and "S" keys, and followed by a click on the tab key on the keyboard.	The flying mode is activated, the character flies and then the flying mode is deactivated.	**Pass**
TFR22	Test Block Type Change - Brick	A block is near the character and the player has already aimed the crosshair at the block(the crosshair could be pointed to top, bottom, or any side).	A click on the "1" key on the keyboard, followed by a right-click on the mouse.	A brick block is built.	**Pass**

TFR23	Test Block Type Change - Grass	A block is near the character and the player has already aimed the crosshair at the block(the crosshair could be pointed to top, bottom, or any side).	A click on the “2” key on the keyboard, followed by a right-click on the mouse.	A grass block is built.	**Pass**
TFR24	Test Block Type Change - Stone	A block is near the character and the player has already aimed the crosshair at the block(the crosshair could be pointed to top, bottom, or any side).	A click on the “3” key on the keyboard, followed by a right-click on the mouse.	A stone block is built.	**Pass**
TFR25	Test ESC menu Open	The software game GUI is opened.	A click on the “ESC” key on the keyboard.	The cursor is released from the GUI and the ESC menu opens.	**Pass**
TFR26	Test Block-Through Movement	The character is on a position where is surrounded by blocks.	A sequence of clicks on “W”, “A”, “S”, “D”.	The character does not move through the surrounding blocks.	**Pass**
TFR27	Test Trivial Keys	The GUI is loaded with a game scene.	A sequence of mouse clicks on keys that are not assigned with any task, in this case use ‘Z’, ‘P’, ‘0’, ‘;’, ‘CTRL’.	The system does nothing to respond and also does not crash.	**Pass**

TFR 4.1	Test Saved Game Exception	The GUI is opened and the game menu is rendered. There is no saved game scene.	A mouse clicks on the "load game" button and a mouse clicks on a game saving spot (Game One or Game Two).	The buttons that control the saving spot selection are not clickable and the system will not load a game scene.	**Pass**
TFR 20.1	Test Jump Action Edge	The software game GUI is opened and the game scene and character are loaded. The character is in a narrow space.	A click on the space key on the keyboard.	The character does not jump.	**Pass**
TFR 7.1	Test Block Outline Edge	The software game GUI is opened and the game scene and character are loaded. Two connected blocks are near the character.	A movement of the mouse to aim the crosshair at the middle of two blocks.	The GUI shows the outline of the block that is closer to the crosshair.	**Pass**
TFR28	Test Menu Click Exception	The software game GUI is rendered with any menu.	A sequence of clicks on the empty spaces on the menu.	The software game does nothing to respond and does not crash.	**Pass**

Table 4: **Tests for Functional Requirements Result**

3 Non-functional Requirements Evaluation

3.1 Appearance

1. **TNFR1 Test Look and Feel**
   Initial State: The game is installed and is given to a group of teenage players.

Expected Result: The group should be satisfied with the attractiveness and style of the game.

Result: The survey was given to three teenage family members of the testing team. The average of the survey result was above the expected average value.

Pass/Fail: Pass

Significance: The test result reflects that the software game product is attractive enough to teenage players.

3.2 Usability

1. **TNFR2: Test Game Difficulty**
   Initial State: The game is installed and is given to a group of random aged people to play.

   Condition: The group should be able to play the game with no difficulty.

   Result: The survey was given to ten family members of the testing team that are from different age groups. The average of the survey result was slightly above the expected average value.

   Pass/Fail: Fail

   Significance: The test result reflects that the software game product is slightly difficult for everyone to play.

2. **TNFR3: Test Learning Curve**
   Initial State: The game is installed and is given to a group of random aged people to play.

   Condition: The group should be able to learn the game within a time between **LEARN_MIN** to **LEARN_MAX** minutes.

   Result: The survey was given to ten family members of the testing team that are from different age groups. The average of the survey result was within the expected time period. However, some of the group members reported that it was difficult for them to learn the game within **LEARN_MAX** minutes.

   Pass/Fail: Pass

Significance: The test result reflects that the software game product is slightly difficult for everyone to learn. The feedback reflects that the it takes time for some individuals to get familiar with the menus and button.

3.3 Performance

1. **TNFR4: Test Speed**
   Initial State: The game is installed.

   Condition: The software game should respond to game events in less than **RESPONSE** second for 99% of the interrogations and no response should take longer than **FALSE_RESPONSE** second.

   Result: The response time of the software game passed the condition of the test case.

   Pass/Fail: Pass

   Significance: The test result reflects that the Python language and Pyglet provides sufficient execution speed for this software game.

2. **TNFR5: Test Availability**
   Initial State: The game is installed.

   Condition: The software game should allow access to the game at different times.

   Result: All accesses to the game were successful.

   Pass/Fail: Pass

   Significance: The test result reflects that the availability of this software game is stable.

3. **TNFR6: Test Reliability**
   Initial State: The game is installed.

   Condition: The software game should run for five hours.

   Result: The software game successfully ran for eight hours. By the end of the eight hours, the tester shut down the game.

   Pass/Fail: Pass

   Significance: The test result reflects that this software game is reliable.

3.4 Operational and Environmental Aspects

1. **TNFR7: Test Adjacent System Effect**
   Initial State: The game is installed.

   Condition: The software game should not produce any negative effects on adjacent systems.

   Result: The tester monitored other programs' performance while running CraftMaster. Some of other programs such as Microsoft Teams experienced latency effect.

   Pass/Fail: Fail

   Significance: The test result reflects that this software game might produce negative effects on adjacent systems.

3.5 Maintainability and Support

1. **TNFR8: Test Adaptability**
   Initial State: The game is available to be downloaded from internet.

   Condition: The software game should be easily downloaded, installed, and opened onto both Windows and Linux OS.

   Result: The tester downloaded, installed and opened the game on both Windows and Linux OS.

   Pass/Fail: Pass

   Significance: The test result reflects that this software game is adaptable to both Windows and Linux OS.

3.6 Security

1. **TNFR9: Test Integrity**
   Initial State: The game is installed.

   Condition: The software game should prevent low level threats.

   Result: The tester created three threat test cases to intentionally abuse the software game. The software did not fail.

   Pass/Fail: Pass

   Significance: The test result reflects the software game is integrity is convinced to some extent.

3.7 Cultural Aspects

1. **TNFR10: Test Cultural Politeness**
   Initial State: The game is installed and is given to a group of people from different cultural groups to play.

   Condition: The group should have satisfaction with the cultural politeness of the game.

   Result: The survey was given to ten friends and family members of the testing team that are from China and Canada. The average of the survey result was above the expected value.

   Pass/Fail: Pass

   Significance: The test result reflects the software game is cultural polite.

3.8 Legal Aspects

1. **TNFR11: Test Compliance**
   Initial State: The game is installed and the documentation is complete.

   Condition: The software product should not violate the Digital Millennuim Copy-right Act[1].

   Result: This meeting with the legal expert was done online instead of in-person due to the current situation. The feedback received shows that the software product does not violate the Digital Millennuim Copy-right Act[1].

   Pass/Fail: Pass

   Significance: The test result reflects the software game does not violate the Digital Millennuim Copy-right Act[1].

3.9 Health and Safety

1. **TNFR12: Test Safety**
   Initial State: The game is installed and the documentation is complete.

   Condition: The software product should not generate any mental or physical threat to the players.

Result: This meeting with the safety expert was done online instead of in-person due to the current situation. The feedback received shows that the software product does not generate any mental or physical threat to the players.

Pass/Fail: Pass

Significance: The test result reflects the software game does not generate any mental or physical threat to the players.

3.10 Robustness

1. **TNFR13: Test Robustness**
   Initial State: The game is installed and ready to execute.

   Condition: The software program does not crash or generate unstable behaviours when receiving unexpected input.

   Result: The unexpected inputs did not cause the software program to behave in an unstable way.

   Pass/Fail: Pass

   Significance: The test result reflects the software game is robust to some extent.

4 Comparison to Existing Implementation

CraftMaster is a reimplementation of Folgeman's Simple Minecraft-inspired Demo(referred to as original project)[2]. The primary improvement of CraftMaster is module decomposition. The original project was built in one Python file with 902 lines, which is low in modularity. In the development process, the development team applied software design principles and software architectural design patterns to increase the modularity and implement the module decomposition. The detail of the module decomposition can be found in the **Module Guide(MG)**. The module decomposition supports the maintainability of this project, which is beneficial for testing. In addition, CraftMaster has refined software qualities and functionalities compared to the original project. The new features and functionalities were fully tested by Functional Requirements test cases and the qualities were evaluated by Non-functional Requirements Evaluations.

5 Unit Testing

The unit testing for all modules was planned out in the **Test Plan**. However, when it came into practice, the testing team realized that some of the Behaviour-Hidding modules are not fully testable by unit testing since the GUI implementation can only be tested by manual testing instead of automated testing. **Section 7** will explain this issue in detail and **Section 10** will compare the actual coverage to the expected coverage. For the functions and classes that are testable, there are multiple unit test cases, including boundary test cases, equivalence test cases, and exception test cases, made for each of the functions. All unit test cases made by the testing team passed.

6 Changes Due to Testing

All the test cases for the functional requirements passed. Therefore, there were no adjustments made to the functionalities and features of CraftMaster. The significance of a few failed Non-functional Requirements Evaluations motivated the development team to make some minor changes and considerations to the software game. The changes and considerations are specified below.

- **Usability:** The result and feedback of the usability test cases reflect that the game is slightly difficult for some individuals to learn and play. To increase the usability of CraftMaster, the development team refined the **User Guide** by adding game menu guidelines and game download guidelines. In addition, the development noticed that some of the game menu buttons are not named reasonably. The names of a few buttons were changed to reduce the confusion caused by the text.

- **Operational and Environmental Aspects:**
  The test case **TNFR7: Test Adjacent System Effect** failed and the testing team noticed that the software game does produce some negative impact on adjacent software programs. It is believed that this issue might be caused by some hardware limitations or low performance of the adjacent programs. The development team has tried to solve this issue. However, since there is no computer hardware expert on the team, it is difficult to figure out the problem correctly. Therefore, this problem is left for future investigation.

7 Automated Testing

As mentioned in the **Test Plan**, the testing tool for automated testing is Pytest. The testing team made efforts to implement automated test cases for every module of CraftMaster. During this process, the team noticed that the graphical content like menu implementation in some modules, including screen, game, world, gameScene, mainScene, settingScene, is not testable by automated testing. Therefore, the testing methods for those implementations were changed to manual testing. As a consequence, the code coverage for those modules cannot meet the expected coverage. Besides, the other modules are fully testable by automated testing and the code coverage meets the expected coverage.

8 Trace to Requirements

Test No.	Req.
TFR1	FR1, FR2
TFR2	FR2.1
TFR3	FR2.1.1, FR3, FR4
TFR4	FR2.1.1, FR3, FR4
TFR5	FR4.1
TFR6	FR6
TFR7	FR9
TFR8	FR10, FR15
TFR9	FR11, FR15
TFR10	FR14
TFR11	FR17
TFR12	FR2.1.2, FR12.2
TFR13	FR2.2
TFR14	FR12.3, FR13
TFR15	FR12.1
TFR16	FR5.1
TFR17	FR5.2
TFR18	FR5.3
TFR19	FR5.4
TFR20	FR7

TFR21	FR8, FR8.1, FR8.2, FR8.3
TFR22	FR11.1
TFR23	FR11.2
TFR24	FR11.3
TFR25	FR12
TFR26	FR16
TFR27	FR18
TFR4.1	FR2.1.1, FR3, FR4
TFR20.1	FR7
TFR7.1	FR9
TFR28	FR2, FR2.1, FR2.1.1, FR2.1.2, FR2.2, FR 12, FR12.1, FR12.2, FR12.3
TNFR1	NFR1
TNFR2	NFR2
TNFR3	NFR3
TNFR4	NFR4
TNFR5	NFR5
TNFR6	NFR6
TNFR7	NFR7
TNFR8	NFR9
TNFR9	NFR10
TNFR10	NFR11
TNFR11	NFR12
TNFR12	NFR13
TNFR13	NFR14

Table 5: **Trace Between Requirements and Test Cases**

9 Trace to Modules

The modules specified in **Module Guide(MG)** are listed below for the convenience of look up.

M1: block

M2: button

M3: creature

M4: devTools

M5: game

M6: loadSource

M7: main

M8: gameScene

M9: mainScene

M10: settingScene

M11: player

M12: processQueue

M13: screen

M14: shape

M15: world

M16: Hardware-Hiding Module(Pyglet)

Test No.	**Module**
TFR1	M1, M2, M3, M4, M5, M6, M7, M8, M9, M10, M11, M12, M13, M14, M15, M16
TFR2	M5, M9, M2, M16, M14
TFR3	M5, M9, M2, M14, M16, M11, M3, M8, M13
TFR4	M5, M9, M2, M14, M16, M11, M3, M8, M13
TFR5	M16, M8, M13, M5
TFR6	M5, M8, M11, M3
TFR7	M16, M5, M8, M15, M11, M13, M14
TFR8	M5, M8, M15, M11, M12, M6, M16
TFR9	M5, M8, M15, M12, M11, M6, M16
TFR10	M7, M6, M16, M5
TFR11	M5, M8, M6, M1
TFR12	M10, M2, M14, M15, M16, M5

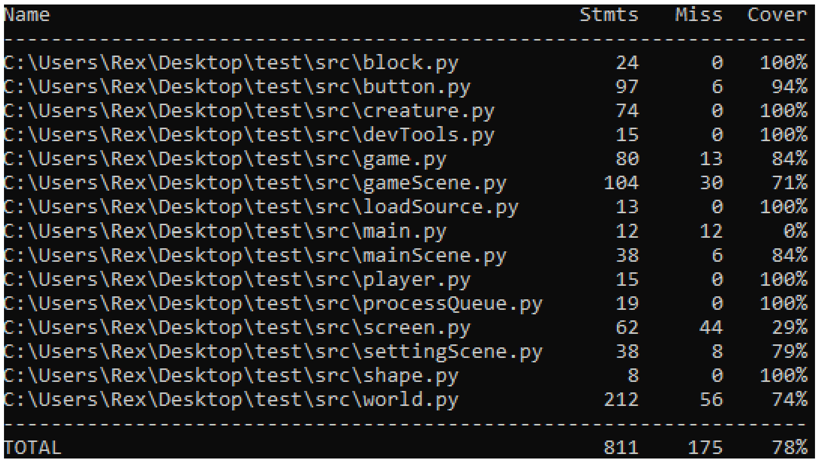
TFR13	M16, M9, M5, M2, M14
TFR14	M5, M10, M2, M14
TFR15	M5, M16, M10
TFR16	M5, M8, M11, M3
TFR17	M5, M8, M11, M3
TFR18	M5, M8, M11, M3
TFR19	M5, M8, M11, M3
TFR20	M5, M8, M11, M3
TFR21	M5, M8, M11, M3
TFR22	M5, M8, M6
TFR23	M5, M8, M6
TFR24	M5, M8, M6
TFR25	M5, M16
TFR26	M5, M8, M15, M11
TFR27	M5, M8, M10, M9
TFR4.1	M5, M9, M2, M14, M16, M3, M11, M8, M13
TFR20.1	M5, M8, M11, M3
TFR7.1	M16, M5, M8, M15, M11, M13, M14
TFR28	M16, M5, M7, M9, M2, M14, M13, M10, M15
TNFR1	M1, M2, M3, M4, M5, M6, M7, M8, M9, M10, M11, M12, M13, M14, M15
TNFR2	M1, M2, M3, M4, M5, M6, M7, M8, M9, M10, M11, M12, M13, M14, M15
TNFR3	M1, M2, M3, M4, M5, M6, M7, M8, M9, M10, M11, M12, M13, M14, M15
TNFR4	M1, M2, M3, M4, M5, M6, M7, M8, M9, M10, M11, M12, M13, M14, M15, M16
TNFR5	M1, M2, M3, M4, M5, M6, M7, M8, M9, M10, M11, M12, M13, M14, M15, M16
TNFR6	M1, M2, M3, M4, M5, M6, M7, M8, M9, M10, M11, M12, M13, M14, M15, M16
TNFR7	M16
TNFR8	M16
TNFR9	M1, M2, M3, M4, M5, M6, M7, M8, M9, M10, M11, M12, M13, M14, M15, M16
TNFR10	M1, M2, M3, M4, M5, M6, M7, M8, M9, M10, M11, M12, M13, M14, M15

TNFR11	M1, M2, M3, M4, M5, M6, M7, M8, M9, M10, M11, M12, M13, M14, M15
TNFR12	M1, M2, M3, M4, M5, M6, M7, M8, M9, M10, M11, M12, M13, M14, M15
TNFR13	M1, M2, M3, M4, M5, M6, M7, M8, M9, M10, M11, M12, M13, M14, M15

Table 6: **Trace Between Modules and Test Cases**

10 Code Coverage Metrics

The expected code coverage proposed in the **Test Plan** was 90%. All the modules that are not associated with the GUI are fully testable by automated testing and the coverage for them is between 90% and 100%. As explained in Section 7, the GUI associated modules are not fully testable by automated testing and the coverage for them cannot reach 90%. The main module is a client to all other modules and it acts as a simple trigger to the software game and therefore it is not tested. The code coverage metric is shown below.



```
Name                                            Stmts   Miss  Cover
-------------------------------------------------------------------
C:\Users\Rex\Desktop\test\src\block.py             24      0   100%
C:\Users\Rex\Desktop\test\src\button.py            97      6    94%
C:\Users\Rex\Desktop\test\src\creature.py          74      0   100%
C:\Users\Rex\Desktop\test\src\devTools.py          15      0   100%
C:\Users\Rex\Desktop\test\src\game.py              80     13    84%
C:\Users\Rex\Desktop\test\src\gameScene.py        104     30    71%
C:\Users\Rex\Desktop\test\src\loadSource.py        13      0   100%
C:\Users\Rex\Desktop\test\src\main.py              12     12     0%
C:\Users\Rex\Desktop\test\src\mainScene.py         38      6    84%
C:\Users\Rex\Desktop\test\src\player.py            15      0   100%
C:\Users\Rex\Desktop\test\src\processQueue.py      19      0   100%
C:\Users\Rex\Desktop\test\src\screen.py            62     44    29%
C:\Users\Rex\Desktop\test\src\settingScene.py      38      8    79%
C:\Users\Rex\Desktop\test\src\shape.py              8      0   100%
C:\Users\Rex\Desktop\test\src\world.py            212     56    74%
-------------------------------------------------------------------
TOTAL                                             811    175    78%
```

Figure 1: **Code Coverage**

11 Appendix

11.1 Symbolic Parameters

- **AVG1: 7**
- **AVG2: 3**
- **LEARN_MIN: 2**
- **LEARN_MAX: 30**
- **RESPONSE: 0.1**
- **FALSE_RESPONSE: 0.5**
- **OPERATION_NUM: 10**
- **TIME: 60**
- **TRIES: 100**

References

[1] ”Video Games and the law: Copyright, Trademark and Intellectual Property”, NewMediaRights, 2020. [Online]. Available: https://www.newmediarights.org/guide/legal/Video_Games_law_ Copyright_Trademark_Intellectual_Property. [Accessed: 06- Feb- 2020].

[2] Fogleman, “fogleman/Minecraft,” GitHub, 16-Feb-2019. [Online]. Available: https://github.com/fogleman/Minecraft. [Accessed: 10-Feb-2020].