CPSC 224
Alex Keyser
Caleb Lefcort
Colin McClelland
-
4/5/2023

Onitama

Test and Coordination Plans

Team Onitama

Alex Keyser

Caleb Lefcort

Colin McClelland

Course: CPSC 224 - Software Development

Instructor: Aaron S. Crandall

CPSC 224
Alex Keyser
Caleb Lefcort
Colin McClelland
-
4/5/2023

I. Introduction

I.1. Project Overview

Testing of Onitama will occur at all stages of production. The most frequent tests to be performed are small unit tests that evaluate an individual behavior within a single class. Testing is most relevant to the classes that handle the direct game logic. The class containing the most detailed game logic is the board class, which therefore lends itself to being the most important class to thoroughly test. Examples of testing can be observed in the checkValidMove function within the board class. This function receives a coordinate pair as an argument and ensures that it represents a valid move for the player to make. Four obvious tests can be produced to ensure that it works correctly. The first test involves the player attempting to move to an unoccupied space, for which the function should return true. The second test involves the player attempting to move to a space which is occupied by one of their own pieces, for which the function should return false. The third test involves the player attempting to move to a space occupied by an enemy piece, for which the function should return true. The fourth test involves the player attempting to move to a coordinate pair that lies outside of the game board's bounds, for which the function should return false. It will also be important to thoroughly test the game initialization to ensure that the necessary conditions to play a game are met. Another major component of the game that must be tested is the GUI. The GUI will be tested through code and user interaction to ensure that it performs as intended under various circumstances. An example of GUI testing extends the above example. When a player selects a piece and card to move, the spaces corresponding to a valid move should be indicated via a border around the space. Through user testing we can ensure that this mechanism works as intended. We may also test this behavior and other elements of the GUI through code via self clickable buttons. Outside of the main gameplay, testing must also occur for the settings splash screen to ensure that when a user indicates a change with the controller, the model is updated and the view is refreshed accordingly.

CPSC 224
Alex Keyser
Caleb Lefcort
Colin McClelland
-
4/5/2023

I.2. Scope

This document exists to provide an overview of the efforts that will be put into testing our version of Onitama. It serves as a reference point for those contributing to the project as well as those who act as users. In the following sections the strategy that the group has used and will continue to use is stated. The most relevant kinds of testing include unit testing, which will be performed routinely during the development stage, Integration testing, which will be performed routinely as individual pieces of the project reach completion, functional testing, which serves to test a single component of the finished project, system testing, which will occur once the project has been completed and evaluates the overall behavior of the game, and finally user testing, which will be performed by friends and peers in order to gather feedback and identify and potential issues.

II. Testing Strategy

Unit testing

1. Identify which aspects of the program that need to be tested.
2. Create tests to model a comprehensive list of possible behaviors.
3. Ensure the tests are designed accurately with no errors.
4. Record the expected behaviors of the tests.
5. Execute the tests.
6. Record the results.
7. If the actual values are not the same as the expected values, create a list of possible reasons to explain the result.
8. Code will be revised using the list of possible explanations and the tests will be run again.
9. Once the tests are running as expected the code/ behavior is ready to be integrated.

Integration testing

1. Once modules have been thoroughly tested, they are ready for integration testing.
2. Modules will be connected, and a list of desired interactions will be generated.
3. Each interaction will be modeled by a series of tests to demonstrate proper integration
4. Ensure the tests are designed accurately with no errors.
5. Record the expected behaviors of the tests.
6. Execute the tests.
7. Record the results.
8. If the actual values are not the same as the expected values, create a list of possible reasons to explain the result.
9. Code will be revised using the list of possible explanations and the tests will be run again.

III. Test Plans

III.1. Unit Testing

For our unit testing, we have been making tests as we make our classes and methods so we can keep track of the functionality. We have also been making an effort to try making tests on parts that we did not code to encourage us to understand each other's code so we have a better overall understanding of our project as a whole. This has proven very useful so far as we all know which methods serve what purposes, and how all of our classes work together. As I have implied, we are all working on unit testing at least a little bit, and making new tests every week. We think that this is very important for all of us to work on tests as we continue to add more features. During our meetups, we have been looking at each other's unit tests and making sure that they pass, and we plan to continue this check-in for our future meetings as well.

III.2. Integration Testing

As a group, we have been doing really well at branching off of each other's progress as well as checking in with each other to make sure anything that we push or code will work. We make sure to do plenty of tests before pushing to main and continue to make progress on our own branches. Right now we are currently integrating Colin’s UI design into our main project,

while also making sure that Alex's GUI integration with the action listeners still works as intended. Making sure that everything works as a team has been a huge part in our success so far and we plan to keep up the same strategy as we progress.

III.3. System Testing

For our system testing, we plan to create a checklist of different requirements that we want our system to meet. Along with requirements, we also want to have a list of possible interactions that we want to make sure aren't possible for the user to accomplish such as pressing too many buttons at once. Once we have our UI integrated, and our game working within it, we will start playing instances of the game and checking off our requirements as we go, and taking notes of problems that we face during these checks. Once we finish a test, we will now have a list of accomplishments as well as a list of bugs that we can look to fix in the future. This will most likely be a recurring strategy that we will implement in our future meetings.

III.3.1. Functional testing:

Very similar to the system testing as a whole, I think the best way to apply our functional testing is to create a checklist that includes all the major game mechanics of the official Onitama rule book. Once we have made that, we will play a couple of full games of our system and mark off the boxes as we go and create a separate list of issues that we run into. Having a defined rule system for the project makes this sort of testing very easy to see what is right and what is wrong and will be a great way to test the system.

III.3.2. User Acceptance Testing:

For our user acceptance testing, we plan to test our system along with a real game of Onitama. Since our goal for this project is to re-create this game in our own program, this will be a great way to test its requirements. Along with that, we will also have our friends test out the game on all of our settings to see how they react to it as a whole and take in user advice to hopefully improve the experience. As we do not quite have defined stakeholders, I would say that the most important aspect that we are trying to achieve is to provide a random user with a fun, and easy-to-understand experience while playing the game. If our users are running into problems with those two goals, we will take notes and try to improve them the best we can.

CPSC 224
Alex Keyser
Caleb Lefcort
Colin McClelland
-
4/5/2023

IV. Glossary

Board - A five by five or seven by seven grid similar to a chess board.

Cards - A list of possible moves.

Piece - The pieces of the game all have the same movesets that are determined by the cards. The only exception is the king which can lose the game if captured.

GUI - Graphical user interface.

Unit test

V. References

Niebling, William. "Onitama Rulebook." Arcane Wonders, January 1, 2015. arcanewonders.com/wp-content/uploads/2021/05/Onitama-Rulebook.pdf.

CPSC 224
Alex Keyser
Caleb Lefcort
Colin McClelland
-
4/5/2023

Appendix-A

Example Testing Strategy:

1. Identify the requirements to be tested. All test cases shall be derived using the current Software Requirements Specification.

2. Identify which particular test(s) will be used to test each module.

3. Review the test data and test cases to ensure that the unit has been thoroughly verified and that the test data and test cases are adequate to verify proper operation of the unit.

4. Identify the expected results for each test.

5. Document the test case configuration, test data, and expected results.

6. Perform the test(s).

7. Document the test data, test cases, and test configuration used during the testing process. This information shall be submitted via the revised Test Plan document.

8. Successful unit testing is required before the unit is eligible for component integration/system testing.

CPSC 224
Alex Keyser
Caleb Lefcort
Colin McClelland
-
4/5/2023

9. Unsuccessful testing requires a bug form to be generated. This document shall describe the test case, the problem encountered, its possible cause, and the sequence of events that led to the problem. It shall be used as a basis for later technical analysis.

10. Test documents and reports shall be submitted. Any specifications to be reviewed, revised, or updated shall be handled immediately.