e.Do Robot Checkers AI: Test Plan

Arhum Ahmed, Adrian Ionascu, Eric LaBerge, Trevor Krestchmann

April 2019

Contents

1 Introduction

This document details our specific methodology and steps followed to perform comprehensive testing of our product. In particular, we address the approach, pass/fail criteria, entry/exit criteria, and suspension/resumption criteria for each form of testing.

2 Functional Testing

2.1 Approach

Each of the functions created in the 3 core components of this project, Image Detection, Manipulation, and AI, will be individually tested. Due to the nature of our program, we have decided that unit testing is the best approach for verifying the validity of our functions.

2.2 Pass/Fail Criteria

In order for a test to be considered "passing", it must output the expected result as defined in the test case. If the output does not match, the results are inconsistent, or the program crashes, the test is considered "failing" and the function being tested must be fixed.

2.3 Entry/Exit Criteria

The entry criteria for testing is after implementing all the features required for prototype 3. The exit criteria is that all the functional requirements of the project work on an individual level.

2.4 Suspension/Resumption Criteria

Testing will be stopped when a test case fails. Testing will be continued once the failing test case has been resolved.

3 Non-Functional Testing

3.1 Approach

Performance sensitive functionality, such as the movement of the arm or the response time of the system as a whole, will be tested here to ensure the system does not hang or seem slow during normal gameplay.

3.2 Pass/Fail Criteria

For a test case to receive a passing status in this section, the system must consistently respond faster or fail less than the maximum number of acceptable failures as specified in the non-functional requirements.

3.3 Entry/Exit Criteria

The Entry Criteria of non-functional testing is the successful completion of functional testing. The Exit Criteria of non-functional testing is the successful completion of all required test cases.

3.4 Suspension/Resumption Criteria

Tests will be suspended if the performance of the system does not meet up to the expectations aligned in the non-functional requirements. Testing will resume when problem parts of the system perform at an acceptable rate as specified in the non-functional requirements.

3.5 Risks/Issues

The operating system running on the Robot has been known to fail on occasion, which would lead to the entire system not working as intended. In addition, lighting has been known to affect the veracity of object detection, which in turn will cause errors in execution, namely the Checkers AI.

4 Integration Testing

4.1 Approach

Each of the functions that communicate directly with another function are tested to ensure the expected results are still being outputted.

4.2 Pass/Fail Criteria

In order for a test to be considered "passing", it must output the expected result as defined in the test case. If the output does not match, the results are inconsistent, or the program crashes the test is considered "failing" and the functions involved or the code integrating the two must be fixed.

4.3 Entry/Exit Criteria

The entry criteria for integration testing is after confirming that each individual function the team has written has passed their test cases in the function testing section. The exit criteria is that all functions that directly communicate with each other are properly functioning and yielding the expected results.

4.4 Suspension/Resumption Criteria

Testing will be suspended when a test case fails, and resumed upon its correction.

5 System Testing

5.1 Approach

System Testing will test the System as a whole, playing the game normally with the Robot, ensuring it does not make any mistakes or become unresponsive during gameplay.

5.2 Pass/Fail Criteria

For the System Test Case, any improper operation (crashing, unresponsiveness, excessive mistakes) will constitute a failure. A lack of these issues will constitute a passing test.

5.3 Entry/Exit Criteria

System Testing will begin after the successful completion of Integration Testing and will be concluded upon completion of all pertinent test cases.

5.4 Suspension/Resumption Criteria

System testing will be halted by improper performance. During such an event, we will test backwards, going from integration back to functional testing until the culprit is determined. Resumption will occur once the error is found and corrected.

5.5 Risks/Issues

There are many integrated parts to the system, and improper operation of one individual part will result in strange behavior towards the user. It is imperative that all individual parts and integration are thoroughly tested before proceeding to a system-wide test.

6 Functional Test Cases

6.1 Checkers AI

Test Case ID	FT1
Test Case Description	AI recognizes valid move
Pre-conditions	1. User has made a valid move.
Test Steps	1. Previous board state array and current board state array are passed into isValidMove function.
Expected Results	1. isValidMove function does not return error message 2. AI knows it's its turn and makes its move.
Priority	Medium
Pass/Fail Criteria	The test is successful if the AI takes its turn after the user is done, fails otherwise

Test Case ID	FT2
Test Case Description	AI recognizes invalid move.
Pre-conditions	1. User has made an invalid move
Test Steps	1. Previous board state array and current board state array are passed into isValidMove function
Expected Results	1. isValidMove function returns error message stating user made an invalid move 2. AI still assumes it's the human's turn and continues waiting for a valid move.
Priority	Medium
Pass/Fail Criteria	Test is successful if error message is printed and AI does not take it's turn, fails otherwise.

Test Case ID	FT3
Test Case Description	Recognizes human has not made a move yet
Pre-conditions	1. It is the human's turn. 2. The human has not made their move yet.
Test Steps	1. Previous board state array and current board state array are passed into isValidMove function
Expected Results	1. AI does not make move
Priority	Low
Pass/Fail Criteria	Test passes if the AI does not take its turn, fails otherwise.

Test Case ID	FT4
Test Case Description	AI makes valid move
Pre-conditions	1. Is is the AIs turn.
Test Steps	1. Current board state array is passed into runAI function
Expected Results	1. The AI outputs a valid move
Priority	High
Pass/Fail Criteria	The test passes if the AI outputs source, destination, and pieces jumped in a legal move it makes, fails otherwise.

Test Case ID	FT5
Test Case Description	RunAI makes a valid jump
Pre-conditions	1. It is the AIs turn. 2. The only move the AI can make is a jump
Test Steps	1. Current board state array is passed into runAI function
Expected Results	1. The AI jumps the piece. 2. The AI correctly outputs the piece it jumped.
Priority	High
Pass/Fail Criteria	The test passes if the AI jumps the piece and outputs source, destination, and piece jumped, fails otherwise.

Test Case ID	FT6
Test Case Description	AI properly detects when game is over
Pre-conditions	1. One color does not appear in the board array
Test Steps	1. Current board state array is passed into runAI function
Expected Results	1. The AI outputs a signal that the game is over.
Priority	Low
Pass/Fail Criteria	The test passes if the AI outputs a signal stating the game is over and says who the winner is, fails otherwise.

Test Case ID	FT7
Test Case Description	AI can make valid moves for a king
Pre-conditions	1. It is the AI's turn 2. The AI only has 1 or more kings (no pawns).
Test Steps	1. The current board state is passed to runAI
Expected Results	1. The AI moves the king to a valid position.
Priority	High
Pass/Fail Criteria	The test passes if the AI makes a legal move for the king (moves top left, top right, bottom left, bottom right) and outputs the source, destination, and pieces jumped in the move.

Test Case ID	FT8
Test Case Description	RunAI can make valid multi-jumps
Pre-conditions	1. It is the AI's turn 2. The AI only has a multijump move available.
Test Steps	1. The current board state is passed to runAI
Expected Results	1. The AI moves the piece from the source to the final jump location 2. The AI stores the pieces jumped in an array.
Priority	High
Pass/Fail Criteria	The test passes if the AI jumps all the possible pieces in the move and outputs the source from its initial location, the destination of its final jump, and the pieces it jumped for that move. The test fails otherwise.

Test Case ID	FT9
Test Case Description	Machine learning adjusts the heuristic after it is defeated
Pre-conditions	1. The AI is defeated in checkers.
Test Steps	1. Run the AIVSAI function
Expected Results	1. The AI randomly adjusts the heuristic it was using. 2. The AI will use the new heuristic for future games.
Priority	Medium
Pass/Fail Criteria	The test will pass if the heuristic is randomly changed after the game.

Test Case ID	FT10
Test Case Description	Machine learning does not adjust the heuristic after it is victorious
Pre-conditions	1. The AI is victorious in checkers.
Test Steps	1. Run the AIVSAI function
Expected Results	1. The AI does nothing with the heuristic it was currently using. 2. The AI will use the same heuristic for future games.
Priority	Medium
Pass/Fail Criteria	The test will pass if the heuristic is unchanged after the game.

6.2 Manipulation

Test Case ID	FT11
Test Case Description	Moving to a location
Pre-conditions	1. The Robot is powered 2. The Robot is accepting messages
Test Steps	1. Invoke the moveToPick() function of the Manipulation class
Expected Results	1. The Robot will move to the specified location 2. "Invalid Move" will be outputted, and the Robot will move to its resting position
Priority	Medium
Pass/Fail Criteria	The test will pass if it yields one of the two expected results, it will fail if anything else occurs

Test Case ID	FT12
Test Case Description	Pick up a piece
Pre-conditions	1. The Robot is powered 2. The Robot is accepting messages 3. The pick spot contains a checkers piece
Test Steps	1. Invoke the pickUpPiece() function of the Manipulation class
Expected Results	1. The Robot will pick up the piece at the specified location 2. The Robot will miss the piece at the specified location, and return to pick position empty handed
Priority	Medium
Pass/Fail Criteria	The test will pass if it successfully picks up the piece, it will fail otherwise

Test Case ID	FT13
Test Case Description	Drop a piece
Pre-conditions	1. The Robot is powered 2. The Robot is accepting messages 3. The drop spot is empty 4. The Robot is currently holding a piece
Test Steps	1. Invoke the dropPiece() function of the Manipulation class
Expected Results	1. The Robot will drop the piece at the specified location 2. The Robot will drop the piece away from the specified location
Priority	Medium
Pass/Fail Criteria	The test will pass if it successfully drops the piece in the right spot, it will fail otherwise

Test Case ID	FT14
Test Case Description	Make a checkers move
Pre-conditions	1. The Robot is powered 2. The Robot is accepting messages 3. There is a checkers board populated with pieces 4. The Robot is currently holding a piece
Test Steps	1. Invoke the checkersMove() function of the Manipulation class
Expected Results	1. The Robot will pick up the piece at the first location, and drop it at the second location 2. The Robot will miss the piece at the first location, and drop nothing at the second location 3. The Robot will pick up the piece at the first location, but drop it away from the second location
Priority	Medium
Pass/Fail Criteria	The test will pass if it successfully picks up the piece at the first location and drops it at the second location, and will fail in any other case

Test Case ID	FT15
Test Case Description	Remove a piece from play
Pre-conditions	1. The Robot is powered 2. The Robot is accepting messages 3. There is a checkers board populated with pieces
Test Steps	1. Invoke the removePiece() function of the Manipulation class
Expected Results	1. The Robot will pick up the piece at the location, and drop it in the captured pieces bucket 2. The Robot will miss the piece at the location, and drop nothing in the captured pieces bucket 3. The Robot will pick up the piece at the location, but fail to drop it in the captured pieces bucket
Priority	Medium
Pass/Fail Criteria	The test will pass if it successfully picks up the piece at the location and drops it in the captured pieces bucket and will fail in any other case

6.3 Object Detection

Test Case ID	FT16
Test Case Description	Object detection recognizes all pieces present within image
Pre-Conditions	1. Camera has been setup above the board with pieces in view
Test Steps	1. Capture an image from the camera 2. Pass the image into the function for detecting pieces
Expected Results	Function returns array of pieces detected within image
Priority	High
Pass/Fail Criteria	The test will pass if all pieces within the image are detected, otherwise the test will fail

Test Case ID	FT17
Test Case Description	Object detection recognizes all squares on the checker board
Pre-Conditions	1. Camera has been setup above the board
Test Steps	1. Capture an image from the camera 2. Pass the image into the function for detecting the board
Expected Results	Function returns array of squares that it derives from the picture of the board
Priority	High
Pass/Fail Criteria	The test will pass if the board can be detected and all squares are outputted, otherwise the test will fail

Test Case ID	FT18
Test Case Description	Object detection recognizes the pieces as different colors
Pre-Conditions	1. An image of the board with pieces visible has been obtained
Test Steps	1. Pass in image and array containing piece locations to function
Expected Results	Function returns two arrays with different colored pieces in their respective array
Priority	High
Pass/Fail Criteria	The test will pass if both arrays accurately contain the pieces detected, otherwise the test will fail

Test Case ID	FT19
Test Case Description	Object detection recognizes all king pieces
Pre-Conditions	1. Camera has been setup above the board
Test Steps	1. Capture an image from the camera 2. Pass the image into the function for detecting king pieces
Expected Results	Function returns array with circles it has detected to be kings
Priority	High
Pass/Fail Criteria	The test will pass if all king pieces are detected within the image, otherwise the test will fail

7 Non-Functional Test Cases

Test Case ID	NT1
Test Case Description	System Response
Pre-conditions	1. The system is fully operational
Test Steps	1. Make a move 2. Record the amount of time needed for the Robot to begin making its move
Expected Results	1. The Robot will respond within 20 seconds of the completion of a player move 2. The Robot will respond after more than 20 seconds of the completion of a player move
Priority	Medium
Pass/Fail Criteria	The test will pass if the Robot begins its move within 20 seconds of the completion of the human player's move

Test Case ID	NT2
Test Case Description	Pick up Performance
Pre-conditions	1. The system is fully operational
Test Steps	1. Have the Robot pick up 20 pieces
Expected Results	1. The Robot will fail during one or none of the 20 moves 2. The Robot will fail during two or more of the 20 moves
Priority	Medium
Pass/Fail Criteria	The test will pass if the Robot fails during less than one of the 20 moves

Test Case ID	NT3
Test Case Description	Placement Performance
Pre-conditions	1. The system is fully operational
Test Steps	1. Have the Robot place 20 pieces
Expected Results	1. The Robot will fail during one or none of the 20 moves 2. The Robot will fail during two or more of the 20 moves
Priority	Medium
Pass/Fail Criteria	The test will pass if the Robot fails during less than one of the 20 moves

Test Case ID	NT4
Test Case Description	Reliability
Pre-conditions	1. The system is fully operational
Test Steps	1. Have the Robot play 10 games
Expected Results	1. The program will crash during one or less of the 10 games 2. The program will crash during two or more of the 10 games
Priority	Medium
Pass/Fail Criteria	The test will pass if the Robot fails during less than one of the 10 games

8 Integration Test Cases

Test Case ID	IT1
Test Case Description	Object Detection works with manipulation to pick up a piece
Pre-Conditions	1. Camera has been setup above the board with a piece visible 2. Robot arm is calibrated
Test Steps	1. Capture an image from the camera 2. Pass the image into the function for detecting piece and its location 3. Derive real world coordinates from previous function results 4. Pass real world coordinates to manipulation code
Expected Results	Robot arm is able to grab the visible piece and move it to specified location
Priority	High
Pass/Fail Criteria	The test will pass if the robot is able to grab the piece, otherwise the test will fail

Test Case ID	IT2
Test Case Description	Object Detection works with AI to plan the next move
Pre-Conditions	1. Camera has been setup above the board with a piece visible
Test Steps	1. Capture an image from the camera 2. Pass the image into the function for detecting piece and its location 3. Convert previous function output to form the AI will expect as input
Expected Results	AI will output the next move it plans to make
Priority	High
Pass/Fail Criteria	The test will pass if the AI is able to output a valid board move, otherwise the test will fail

9 System Test Cases

Test Case ID	ST1
Test Case Description	All parts of the project work properly to make a move in checkers
Pre-Conditions	1. Camera has been setup above the board with pieces and board visible 2. Robot has been calibrated
Test Steps	1. Capture an image from the camera 2. Pass the image into object detection functions to detect board and pieces 3. Convert object detection output into array that the AI will input 4. Get next move from AI 5. Convert object detection output to real world coordinates 6. Give manipulation function real world coordinates
Expected Results	The robot will make a valid move in the current game it is playing
Priority	High
Pass/Fail Criteria	The test will pass if the robot is able to make a valid move without error, otherwise the test will fail

10 Schedule

Date	Target	Responsibility
4/12/19	Functional Test Cases Complete	AI: Arhum & Trevor Image Detection: Adrian Manipulation: Eric
4/13/19	Non Functional Test Cases Complete	Arhum, Adrian, Eric, Trevor
4/15/19	Integration Test Cases Complete	Arhum, Adrian, Eric, Trevor
4/16/19	System Test Cases Complete	Arhum, Adrian, Eric, Trevor
4/17/19	User Acceptance Complete	Arhum, Adrian, Eric, Trevor
4/18/19	Final Product Submission	Arhum, Adrian, Eric, Trevor