**Date: 12/12/2014**

Final Report

**Nine Men's Morris**

**CS 471/571 Software Engineering**

**Advisor: Prof. Dianxiang Xu**

**Member: Milson Munakami**

**Jimmy Wang**

**Sung-Ju Fan-Chiang**

Team Project Final Report

Section I. Team Organization and Buddy Rating ........................................................................ 3

(1) Member's Work and Tasks

(2) Buddy Ratings

(3) Team Meetings and Meeting Minutes

Section II. Requirements Specification ....................................................................................... 5

(1) The Main Use Case Diagrams

(2) Description for Use Cases

(3) Use Case Scenario and Sequence Diagram

Section III. Design........................................................................................................................ 13

(1) User-interface Design

(2) Class Diagram

(3) The Algorithm for Determining When a Mill Is Formed

(4) The Algorithm for Determining When the Game Is Over

(5) The Algorithm for the Automated Computer Player

Section IV. Test ......................................................................................................................... 17

(1) The Test to Against the Use Case Scenarios

(2) Test Steps for the Human Player to Win

(3) Test Steps for the Computer Player to Win

Section V. Lessons Learned ....................................................................................................... 26

Member: Milson Munakami

Member: Jimmy Wang

Member: Sung-Ju Fan-Chiang

Section I. Team Organization and Buddy Rating (1%)

(1) Member's Work and Tasks

|  |  |
| --- | --- |
| Member | Work and Tasks |
| Milson Munakami | Team leader  Game GUI design and programming  Testing  Scrum master  Main programmer for pair programming |
| Jimmy Wang | Coordinator for the project progress  Testing  Backend logic and algorithm  Main programmer for pair programming |
| Sung-Ju Fan-Chiang | Documentation  Testing  Minor programmer for pair programming |

(2) Buddy Ratings

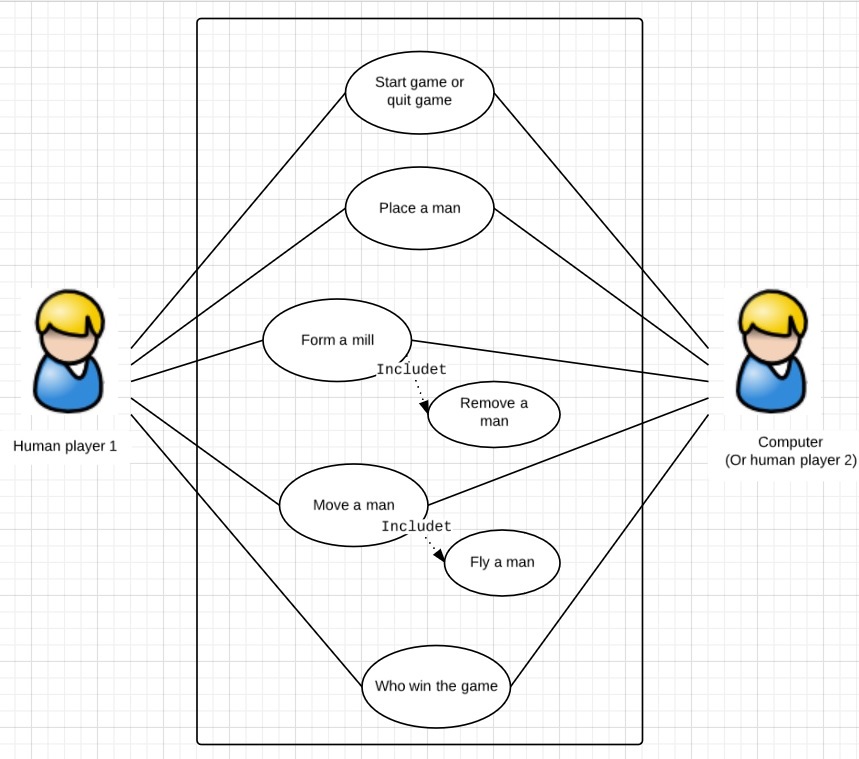
|  |  |  |  |
| --- | --- | --- | --- |
| Member | Buddy | Rating | Note |
| Milson Munakami | Jimmy Wang | 1 |  |
| Sung-Ju Fan-Chiang | 1 |  |
| Jimmy Wang | Milson Munakami | 1 |  |
| Sung-Ju Fan-Chiang | 1 |  |
| Sung-Ju Fan-Chiang | Milson Munakami | 1 |  |
| Jimmy Wang | 1 |  |

(3) Team Meetings and Meeting Minutes

|  |  |  |  |
| --- | --- | --- | --- |
| Meeting # | Time/place | Participants | Topics and Decisions |
| 1 | 10/29  13:00 - 13:30  MEC410 | Milson  Jimmy  Sung-Ju | * Merge place and move algorithm to game board |
| 2 | 10/31  13:00 - 13:40  MEC410 | Milson  Jimmy  Sung-Ju | * Discuss further schedule and job function for team member |
| 3 | 11/5  13:00 - 13:30  MEC410 | Milson  Jimmy  Sung-Ju | * Test the overall placement, movement, remove, and the flow |
| 4 | 11/7  MEC410 | Milson  Jimmy  Sung-Ju | No meeting |
| 5 | 11/12  13:00 - 13:30  MEC410 | Milson  Jimmy  Sung-Ju | * Review all game rules and confirm the function can work |
| 6 | 11/14  13:00 - 13:30  MEC410 | Milson  Jimmy  Sung-Ju | * Review current code by use case workflow * Add automated computer player * Modify code to meet all game rules |
| 7 | 11/19  13:20 - 13:50  MEC410 | Milson  Jimmy  Sung-Ju | * Test the overall function and debug the program |
| 8 | 11/21  13:00 - 13:40  MEC410 | Milson  Jimmy  Sung-Ju | * Test the automated computer function and improve the function |
| 9 | 12/3  13:00 - 13:30  MEC410 | Milson  Jimmy  Sung-Ju | * Edit use case and the 1st draft for final report |
| 10 | 12/5  13:00 - 13:30  MEC410 | Milson  Jimmy  Sung-Ju | * Wrap up the code and add comment * Install ObjectAid license to produce diagram for report |
| 11 | 12/10  13:00 - 14:00  MEC410 | Milson  Jimmy  Sung-Ju | * Last edition for the document * Final demo rehearsal * Last improvement for AI |

Section II. Requirements Specification (14%)

(1) The Main Use Case Diagrams



(2) Description for Use Cases

|  |
| --- |
| **Brief Description**  The Start game or quit game use case enable the game board to start the game or quit the game. |
| **Step-by-step Description**  1. The first pop up GUI is shown when the game have been run.  GUI_first.jpg  2. Player need to select either to play with another player or to play with computer. That  is to click either dot for player 2 option.    3. Once the decision have been made, player can click OK button to go and the game  board GUI will pop up for further movement. The game can be terminated if player click  Cancel button and there is no game.  4. When the game board GUI ready, player can start to play the game and there are 9 men  available to be placed. Player can start an new game if the New button have been click    any time and the player will start from step 1. Besides, player can terminate the game if    the Quit button have been clicked. |

|  |
| --- |
| **Brief Description**  The Place a man use case enable each player or computer to place a man by turns. All placement need to follow all of the rules. The Place a man use case enable Form a mill use case if the mill rule have been satisfied. The Place a man use case enable Move a man use case if all 9 men have been placed. |
| **Step-by-step Description**  1. The game start with player 1 to place a first man. After that, player 2 place a 2nd man  on the game board. Every man need to place on bold dot and cannot be placed on the  dot which is occupied.  2. If the rule to form a mill is matched, the Place a man use case enable Form a mill use  case. Then the Form a mill use case enable Remove a man use case to remove a man.  3. Repeat step 1 and step 2 until all of 9 men have been placed.  4. The Place a man use case enable Move a man for phase two game to start move a man. |

|  |
| --- |
| **Brief Description**  The Move a man use case enable each player or computer move a man by turns. All movement need to follow all of the rules. The Move a man use case enable Form a mill use case if the mill rule have been satisfied. The Move a man use case enable Who win the game use case if the end of the game status is fulfilled. |
| **Step-by-step Description**  1. The game start with player 1 to move a man. After that, player 2 move another man  by turns. Every movement need to move on bold dot and cannot move on the dot  which is occupied.  2. If the rule to form a mill is matched, the Move a man use case enable Form a mill use  case. Then the Form a mill use case enable Remove a man use case to remove a man.  3. Repeat step 1 and step 2 until one side have only three man on the board. Then the Fly  a man use case have be enabled, and any one of the three men can fly.  4. Either one of the following status happened, go to step 5: Any side have two men only,  or any side have more than 4 men but cannot move any man.  5. The Move a man use case enable Who win the game use case if step 4 is true. And the  Final GUI pop up to show winner. |

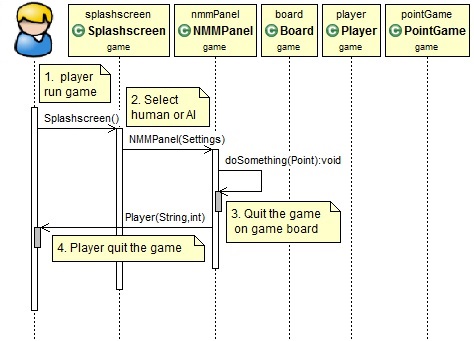
|  |
| --- |
| **Brief Description**  The Form a mill use case enable Remove a man use case to remove a man from another side by the rule. |
| **Step-by-step Description**  1. In the phase one to place a man, once the mill rule is be fulfilled by the Place a man use  case the Remove a man use case have been enabled. There is one man on another side  should be removed, but the man to form a mill cannot be removed.  2. In the phase two to remove a man, once the mill rule is be fulfilled by the Move a man  use case the Remove a man use case have been enabled. There is one man on another  side should be removed, but the man to form a mill cannot be removed.  3. In the phase three to fly a man, , once the mill rule is be fulfilled by the Fly a man  use case the Remove a man use case have been enabled. There is one man on another  side should be removed, but the man to form a mill cannot be removed. |

|  |
| --- |
| **Brief Description**  The Who win the game use case enable final GUI to show winner. |
| **Step-by-step Description**  1. If any of the final status rule is true, the Who win the game use case enable final GUI to  show winner.  2. Player know who is winner and click the OK button to quit the game.    GUI_4.jpg |

(3) Use Case Scenario and Sequence Diagram

Start game or quit game use case

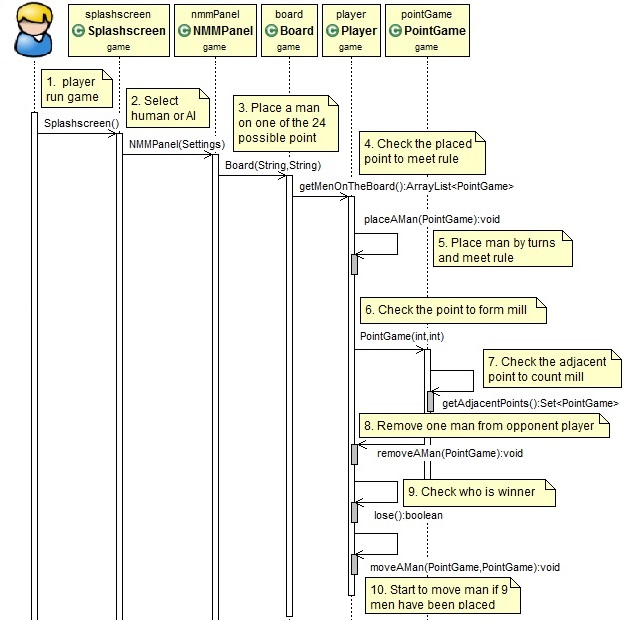
|  |
| --- |
| **Player start game but decide to quit** |
| 1. Player run the game and the pop up GUI is shown.  2. Player select either to play with another player or to play with computer. Then the  The game board replace the first GUI.    3. Never mind player play or not but the player decide to quit the game. Player click Quit  button.  4. The game board disappear and the player quit the game. |



**Note** **A.** The use case is different from the class from JAVA program because the code was done before to construct the use case diagram. But, there are very close. To compare the use case, the Splashscreen is a ***boundary class*** and most likely to be Start game or quit game use case. And the NMMPanel is anotehr ***boundary class***. The Board is an ***entity class*** and participate with partial Place a man use case. The Player is a ***control class*** and have multiple function for Place a man, Move a man, and Who win the game use case. The PointGame is a ***control class*** and most likely to be a Form a mill use case. **Note B.** All of the other use case scenario can be terminated if Quit button have been click, however, all use case scenario have assume this status never happen. **Note C.** The Form a mill use case is enable by Place a man and Move a man use case, therefore, the sequence diagram is on those two use case.

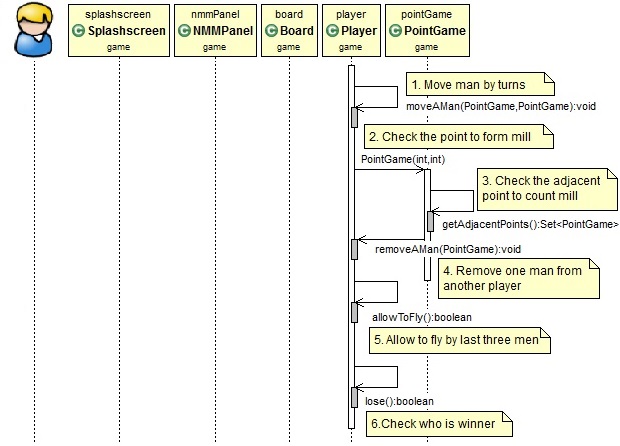
Place a man use case

|  |
| --- |
| **Player places a man on game board, and the placement need to follow all of the rules. The player form a mill to remove one man from another player.** |
| 1. Player run the game and the pop up GUI is shown.  2. Player select either to play with another player or to play with computer.  3. Player start to place a man on the game board on certain spots and by turns.  4. Any man need to be placed by rules and have checked the spots before to be placed.  5. There are nine men each player and all of the men will be placed and check rules.  6. Besides other rules, check each placed man to form a mill or not.  7. By Form a mill use case, the adjacent points need to be check in order to confirm the  mill have been form.  8. If a mill have been form, player remove one man from opponent player by the point  which is not from a formed mill.  9. By Who is winner use case, if the total man on the board and the man available to be  placed are less than three, the player lose the game.  10. Repeat step 5 to 8 until all nine men have been placed. Start to move man if nine  available men have been placed. |



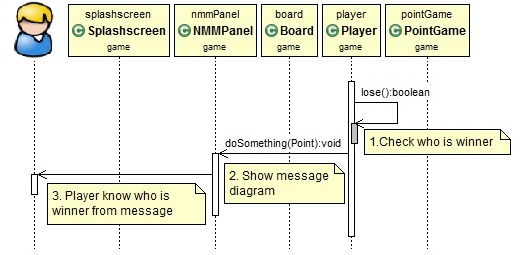
Move a man use case

|  |
| --- |
| **Player moves a man on game board, and the movement need to follow all of the rules. The player form a mill to remove one man from another player.** |
| The movement start when the Place a man use case ended  1. Player move man by turns and try to form a mill to remove the man by another side.  2. Check each movement to confirm the mill is formed.    3. By Form a mill use case, the adjacent points need to be check in order to confirm the  mill have been form.  4. If a mill have been form, player remove one man from another side which is not from a  formed mill. Repeat step 1 to step 4 until one side have only three men.  5. According to the rule, any man allow to fly if the player has one three men.  6. Repeat step 1 to step 5 until another side have only three men and the opponent can fly  a man too. By Who is winner use case for the following rules: all men have been blocked  by another side or only have two men on the board. |



Who is winner use case

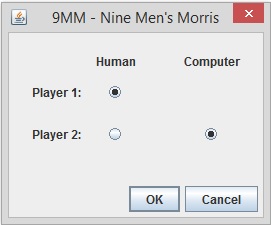
|  |
| --- |
| **The message diagram show player who is winner by the rule.** |
| 1. Check who is winner by the following rules: all men have been blocked by opponent  player or only have two men on the board.  2. The message diagram pop up and display who is winner.    3. The player know who is winner from the message diagram. |



Section III. Design (14%)

(1) User-interface Design

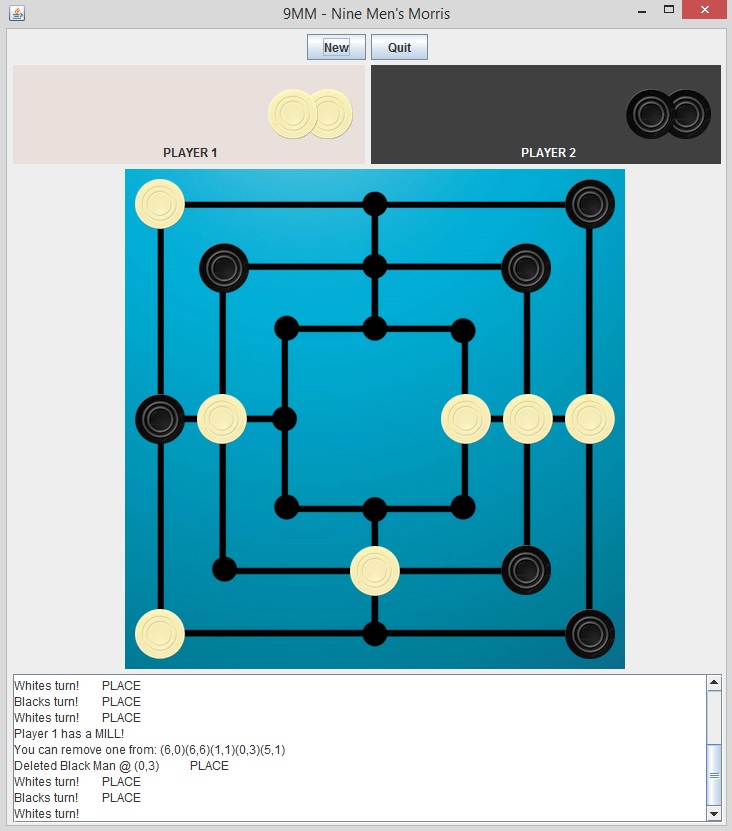
The game start with the GUI as below. The "Player 1" is always there but user and choice "Player 2" to be either Human or Computer. If user choice Human, the game start with twp player and click OK button to go. If user choice Computer, there is software AI to play with user and can click Ok button to go. Besides, user can click Cancel button to cancel the game.



There are two players on the board (the game board shown on next page) by PLAYER 1 for white chips and by PLAYER 2 for black chips. The game start with PLAYER 1 to place white chip on the board and for bold point only. Then the PLAYER 2 can place block chips and both of the player try to make a mill to remove chip for another side. All of the information, any warning, or reminder show on the white board area. Moreover, to start a new game, user can click New button any time to show a new game board and to form a new game. If user need to exit the game without play another one, user can click Quit button to leave the game.

By any normal condition to terminate the game, the GUI below would pop up and define who is the winner.



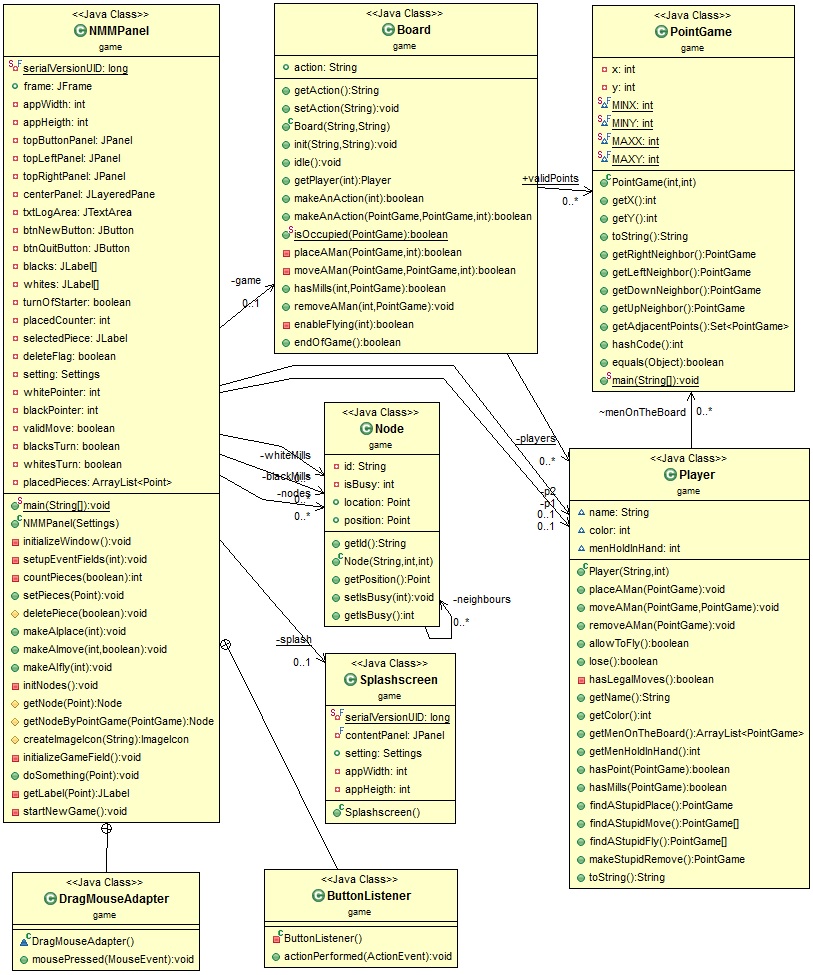


The figure below shows the board for phase two to move pieces. There are no man available on the PLAYER1 and PLAYER2 area to be placed on board. The man which have been chosen to move would be highlighted, e.g., the black pieces in the middle of first line.



(2) Class Diagram

There are six main class which is from the main use cases : NMMPanel, Splashscreen, Node, Board, Player, and PointGame. The ButtonListener is the associated class from NMMPanel class.



(3) The Algorithm for Determining When a Mill Is Formed

The hasMills (PointGame pt) is in the Player.java class.

The algorithm is outlined as following:

Given a point (a man either placed or moved to) of the player’s move, we get the left/right/up/down neighbors of this point, and there are 6 ways that a mill is formed related to this newly added point.

Condition 1: The mill is formed by the Left Neighbor, the point, and the Right Neighbor. (x+-);

Condition 2: The mill is formed by the Up neighbor, the point, and the Down Neighbor. (y+-);

Condition 3: The mill is formed by the Left of the Left, the Left neighbor, and the point. (x--);

Condition 4: The mill is formed by the point, the Right Neighbor, and the Right of the right.(x++);

Condition 5: The mill is formed by the Up of the up, the up neighbor, and the point. (y--);

Condition 6: The mill is formed by the Down of the down, the down neighbor, and the point.

(y++);

If none of the above condition satisfied, the newly added point does not form a mill.

(4) The Algorithm for Determining When the Game Is Over

endOfGame() is in the Board.java class, which is calling player1.lose() || player2.lose() method with a Boolean return. The lose() method is in the Player.java class.

There are 2 conditions for each player that end the game:

1. menOnTheBoard == 2 and menHoldInHand == 0;

2. menHoldInHand == 0 and menOnTheBoard != 3 and !hasLegalMoves().

For hasLegalMoves() method, we basically iterate all the Points for the menOnTheBoard for that player, checking the adjacent points of each, if there is an unoccupied neighbor, return true; otherwise, no legal moves so return false.

(5) The Algorithm for the Automated Computer Player

The computer player is designed to be easy to control (for showing the scenarios).

makeAIplace: call findStupidPlace() in Player.java, which first check if there is any open spots that can form a mill for the computer, if there is, place a man at that point and forms a mill. Otherwise, place at the first available points, as listed in the Board.validPoints arrayList.

MakeAImove: call findAStupidMove() or FindAStupidFly() (if flying), which is iterating through all the points in menHoldInHand, and get the adjacent points for each point, find the first available point to move to, return the point as pointFrom and this adjacent point as pointTo, back to the makeAImove method, and performs the movement.

Section IV. Testing (14%)

(1) The Test to Against the Use Case Scenarios

Start game or quit game use case scenario test

|  |
| --- |
| **Player start game but decide to quit** |
| 1. Player run the game and the pop up GUI is shown.  --> test to run the program -- pass  2. Player select either to play with another player or to play with computer. Then the  The game board replace the first GUI.  --> on first GUI to click Player 2 and click OK --> show game board -- pass  --> on first GUI to click Computer and click OK --> show game board -- pass  GUI_first.jpg  3. Never mind player play or not but the player decide to quit the game. Player click Quit  button.  --> player click New on placement (phase one) --> start the first GUI --> pass  --> player click New on movement (phase two) --> start the first GUI --> pass  --> player click New on flying (phase three) --> start the first GUI --> pass  --> player click Quit on placement (phase one) --> quit the game --> pass  --> player click Quit on movement (phase two) --> quit the game --> pass  --> player click Quit on flying (phase three) --> quit the game --> pass  GUI _5.jpg  4. The game board disappear and the player quit the game.  (All of the test have done on step 3) |

Place a man use case scenario test

|  |
| --- |
| **Player places a man on game board, and the placement need to follow all of the rules. The player form a mill to remove one man from another player.** |
| 1. Player run the game and the pop up GUI is shown.  --> start test to run the program -- pass  2. Player select either to play with another player or to play with computer.  --> on first GUI to click Player 2 and click OK --> show game board -- pass  --> on first GUI to click Computer and click OK --> show game board -- pass    3. Player start to place a man on the game board on certain spots and by turns.  --> player can place a man -- pass  --> player place a man on available spot -- pass  --> player can place a man by turns -- pass  4. Any man need to be placed by rules and have checked the spots before to be placed.  (All of the test have done one step 3)  5. There are nine men each player and all of the men will be placed and check rules.  --> all nine men can be placed by player 1 -- pass  --> all nine men can be placed by player 2 -- pass  6. Besides other rules, check each placed man to form a mill or not.  --> form a mill or not to form a mill by player 1 -- pass  --> form a mill or not to form a mill by player 2 -- pass  7. By Form a mill use case, the adjacent points need to be check in order to confirm the  mill have been form.  --> when player 1 form a mill the adjacent points have a man -- pass  --> when player 2 form a mill the adjacent points have a man -- pass  8. If a mill have been form, player remove one man from opponent player by the point  which is not from a formed mill.  --> when player 1 form a mill, the man from opponent player can be moved -- pass  --> when player 2 form a mill, the man from opponent player can be moved -- pass  --> the man that can be moved from player 1 do not form a mill -- pass  --> the man that can be moved from player 2 do not form a mill -- pass    9. By Who is winner use case, if the total man on the board and the man available to be  placed are less than three, the player lose the game.  --> player 1 win the game because player 2 have no legal move -- pass  --> player 2 win the game because player 1 have no legal move -- pass  --> player 1 win the game because player 2 have less than three men -- pass  --> player 2 win the game because player 1 have less than three men -- pass  10. Repeat step 5 to 8 until all nine men have been placed. Start to move man if nine  available men have been placed.  --> all test have pass to repeat step 5 to 8 for player 1 and ready to move a man -- pass  --> all test have pass to repeat step 5 to 8 for player 2 and ready to move a man -- pass |

Move a man use case scenario test

|  |
| --- |
| **Player moves a man on game board, and the movement need to follow all of the rules. The player form a mill to remove one man from another player.** |
| The movement start when the Place a man use case ended  1. Player move man by turns and try to form a mill to remove the man by another side.  --> player 1 can move a man by turns -- pass  --> player 2 can move a man by turns -- pass  2. Check each movement to confirm the mill is formed.  --> form a mill or not to form a mill by player 1 -- pass  --> form a mill or not to form a mill by player 2 -- pass    3. By Form a mill use case, the adjacent points need to be check in order to confirm the  mill have been form.  --> when player 1 form a mill the adjacent points have a man -- pass  --> when player 2 form a mill the adjacent points have a man -- pass  4. If a mill have been form, player remove one man from another side which is not from a  formed mill. Repeat step 1 to step 4 until one side have only three men.  --> when player 1 form a mill, the man from opponent player can be moved -- pass  --> when player 2 form a mill, the man from opponent player can be moved -- pass  --> the man that can be moved from player 1 do not form a mill -- pass  --> the man that can be moved from player 2 do not form a mill -- pass  --> all test have pass to repeat step 1 to 4 for player 1 and ready to fly a man -- pass  --> all test have pass to repeat step 1 to 4 for player 2 and ready to fly a man -- pass  5. According to the rule, any man allow to fly if the player has one three men.  --> player 1 can fly a man with the last three men -- pass  --> player 2 can fly a man with the last three men -- pass  6. Repeat step 1 to step 5 until another side have only three men and the opponent can fly  a man too. By Who is winner use case for the following rules: all men have been blocked  by another side or only have two men on the board.  --> repeat step 1 to step 5, player 1 win the game because player 2 have no legal move  - - pass  --> repeat step 1 to step 5, player 2 win the game because player 1 have no legal move  -- pass  --> repeat step 1 to step 5, player 1 win the game because player 2 have less than three  men -- pass  --> repeat step 1 to step 5, player 2 win the game because player 1 have less than three  men -- pass |

Who is winner use case scenario test

|  |
| --- |
| **The message diagram show player who is winner by the rule.** |
| 1. Check who is winner by the following rules: all men have been blocked by opponent  player or only have two men on the board.  --> player 1 win the game because player 2 have no legal move -- pass  --> player 2 win the game because player 1 have no legal move -- pass  --> player 1 win the game because player 2 have less than three men -- pass  --> player 2 win the game because player 1 have less than three men -- pass  2. The message diagram pop up and display who is winner.  --> player 1 win the game --> message diagram pop up and display player 1 is winner  -- pass  --> player 2 win the game --> message diagram pop up and display player 2 is winner  -- pass    3. The player know who is winner from the message diagram.  (All of the test have done on step 2) |

(2) Test Steps for the Human Player to Win

Some of the tests have done to against the computer player, and one of the test have shown below. According to all of the use case, the tests follow the flow and play the game by rules. To let human win, the human player need to pay attention to not make too many mistake to let computer always form mill. All of the test not only review the placement and movement by all rules but can check out how computer place a man and move a man. All of the tests have cover the following use case: place a man, move a man, form a mill, remove a man, fly a man, until who win the game. Both human player and computer should try or let it try form a mill, remove a man, and fly a man to make sure the possible AI function have been applied. The detail steps have been shown and form a mill and to remove a man have been highlighted as the test below, and human win the game.

Game Initializing......Done

Game started: White (Player1) first:

Start: true

Player 1: Player\_1\_Human

Player 2: Player\_2\_Computer

Player1's turn.

Player1 place a man on (0,0)

(3,0)

false

(0,0)

Player2's turn.

Player2 place a man on (3,0)

(0,0)

(3,0)

Player1's turn.

Player1 place a man on (0,3)

(6,0)

false

(0,0)(0,3)

(3,0)

Player2's turn.

Player2 place a man on (6,0)

(0,0)(0,3)

(3,0)(6,0)

Player1's turn.

Player1 place a man on (0,6)

Player2 lose a man on (6,0)

Point (6,0) **removed**\*\*\*\*\*\*\*\*\*\*

(6,0)

false

(0,0)(0,3)(0,6)

(3,0)

Player2's turn.

Player2 place a man on (6,0)

(0,0)(0,3)(0,6)

(3,0)(6,0)

Player1's turn.

Player1 place a man on (1,1)

(3,1)

false

(0,0)(0,3)(0,6)(1,1)

(3,0)(6,0)

Player2's turn.

Player2 place a man on (3,1)

(0,0)(0,3)(0,6)(1,1)

(3,0)(6,0)(3,1)

Player1's turn.

Player1 place a man on (3,2)

(5,1)

false

(0,0)(0,3)(0,6)(1,1)(3,2)

(3,0)(6,0)(3,1)

Player2's turn.

Player2 place a man on (5,1)

(0,0)(0,3)(0,6)(1,1)(3,2)

(3,0)(6,0)(3,1)(5,1)

Player1's turn.

Player1 place a man on (1,3)

(2,2)

false

(0,0)(0,3)(0,6)(1,1)(3,2)(1,3)

(3,0)(6,0)(3,1)(5,1)

Player2's turn.

Player2 place a man on (2,2)

(0,0)(0,3)(0,6)(1,1)(3,2)(1,3)

(3,0)(6,0)(3,1)(5,1)(2,2)

Player1's turn.

Player1 place a man on (2,3)

Player2 lose a man on (2,2)

Point (2,2) **removed**\*\*\*\*\*\*\*\*\*\*

(2,2)

false

(0,0)(0,3)(0,6)(1,1)(3,2)(1,3)(2,3)

(3,0)(6,0)(3,1)(5,1)

Player2's turn.

Player2 place a man on (2,2)

(0,0)(0,3)(0,6)(1,1)(3,2)(1,3)(2,3)

(3,0)(6,0)(3,1)(5,1)(2,2)

Player1's turn.

Player1 place a man on (1,5)

Player2 lose a man on (3,0)

Point (3,0) **removed**\*\*\*\*\*\*\*\*\*\*

(3,0)

false

(0,0)(0,3)(0,6)(1,1)(3,2)(1,3)(2,3)(1,5)

(6,0)(3,1)(5,1)(2,2)

Player2's turn.

Player2 place a man on (3,0)

(0,0)(0,3)(0,6)(1,1)(3,2)(1,3)(2,3)(1,5)

(6,0)(3,1)(5,1)(2,2)(3,0)

Player1's turn.

Player1 place a man on (3,4)

(4,2)

false

(0,0)(0,3)(0,6)(1,1)(3,2)(1,3)(2,3)(1,5)(3,4)

(6,0)(3,1)(5,1)(2,2)(3,0)

Player2's turn.

Player2 place a man on (4,2)

(0,0)(0,3)(0,6)(1,1)(3,2)(1,3)(2,3)(1,5)(3,4)

(6,0)(3,1)(5,1)(2,2)(3,0)(4,2)

Player1's turn.

Player1 to move a man:

Available points:

(0,0)(0,3)(0,6)(1,1)(3,2)(1,3)(2,3)(1,5)(3,4)From:

------Selected FROM Point was: ------

1,5

------Selected TO Point was: ------

3,5

Player1 move a man from (1,5) to (3,5)

Player1 to move a man:

Available points:

(6,0)(3,1)(5,1)(2,2)(3,0)(4,2)------Selected FROM Point was: ------

6,0

To:

(6,3)

Player2 move a man from (6,0) to (6,3)

(0,0)(0,3)(0,6)(1,1)(3,2)(1,3)(2,3)(3,4)(3,5)

(3,1)(5,1)(2,2)(3,0)(4,2)(6,3)

Player1's turn.

Player1 to move a man:

Available points:

(0,0)(0,3)(0,6)(1,1)(3,2)(1,3)(2,3)(3,4)(3,5)From:

------Selected FROM Point was: ------

0,6

------Selected TO Point was: ------

3,6

Player1 move a man from (0,6) to (3,6)

Player 1 has a **MILL**!

Player2 lose a man on (6,3)

Point (6,3) **removed**\*\*\*\*\*\*\*\*\*\*

Player1 to move a man:

Available points:

(3,1)(5,1)(2,2)(3,0)(4,2)------Selected FROM Point was: ------

5,1

To:

(5,3)

Player2 move a man from (5,1) to (5,3)

(0,0)(0,3)(1,1)(3,2)(1,3)(2,3)(3,4)(3,5)(3,6)

(3,1)(2,2)(3,0)(4,2)(5,3)

Player1's turn.

Player1 to move a man:

Available points:

(0,0)(0,3)(1,1)(3,2)(1,3)(2,3)(3,4)(3,5)(3,6)From:

------Selected FROM Point was: ------

1,3

------Selected TO Point was: ------

1,5

Player1 move a man from (1,3) to (1,5)

Player1 to move a man:

Available points:

(3,1)(2,2)(3,0)(4,2)(5,3)------Selected FROM Point was: ------

3,1

To:

(5,1)

Player2 move a man from (3,1) to (5,1)

(0,0)(0,3)(1,1)(3,2)(2,3)(3,4)(3,5)(3,6)(1,5)

(2,2)(3,0)(4,2)(5,3)(5,1)

Player1's turn.

Player1 to move a man:

Available points:

(0,0)(0,3)(1,1)(3,2)(2,3)(3,4)(3,5)(3,6)(1,5)From:

------Selected FROM Point was: ------

0,3

------Selected TO Point was: ------

1,3

Player1 move a man from (0,3) to (1,3)

Player 1 has a **MILL**!

Player2 lose a man on (4,2)

Point (4,2) **removed**\*\*\*\*\*\*\*\*\*\*

Player1 to move a man:

Available points:

(2,2)(3,0)(5,3)(5,1)------Selected FROM Point was: ------

3,0

To:

(3,1)

Player2 move a man from (3,0) to (3,1)

(0,0)(1,1)(3,2)(2,3)(3,4)(3,5)(3,6)(1,5)(1,3)

(2,2)(5,3)(5,1)(3,1)

Player1's turn.

Player1 to move a man:

Available points:

(0,0)(1,1)(3,2)(2,3)(3,4)(3,5)(3,6)(1,5)(1,3)From:

------Selected FROM Point was: ------

1,3

------Selected TO Point was: ------

0,3

Player1 move a man from (1,3) to (0,3)

Player1 to move a man:

Available points:

(2,2)(5,3)(5,1)(3,1)------Selected FROM Point was: ------

5,3

To:

(4,3)

Player2 move a man from (5,3) to (4,3)

(0,0)(1,1)(3,2)(2,3)(3,4)(3,5)(3,6)(1,5)(0,3)

(2,2)(5,1)(3,1)(4,3)

Player1's turn.

Player1 to move a man:

Available points:

(0,0)(1,1)(3,2)(2,3)(3,4)(3,5)(3,6)(1,5)(0,3)From:

------Selected FROM Point was: ------

2,3

------Selected TO Point was: ------

1,3

Player1 move a man from (2,3) to (1,3)

Player 1 has a **MILL**!

Player2 lose a man on (2,2)

Point (2,2) **removed**\*\*\*\*\*\*\*\*\*\*

Player1 to move a man:

Available points:

(5,1)(3,1)(4,3)------Selected FROM Point was: ------

5,1

To:

(3,0)

Player2 **flying** enabled!

Player2 move a man from (5,1) to (3,0)

(0,0)(1,1)(3,2)(3,4)(3,5)(3,6)(1,5)(0,3)(1,3)

(3,1)(4,3)(3,0)

Player1's turn.

Player1 to move a man:

Available points:

(0,0)(1,1)(3,2)(3,4)(3,5)(3,6)(1,5)(0,3)(1,3)From:

------Selected FROM Point was: ------

3,5

------Selected TO Point was: ------

5,5

Player1 move a man from (3,5) to (5,5)

Player1 to move a man:

Available points:

(3,1)(4,3)(3,0)------Selected FROM Point was: ------

3,1

To:

(6,0)

Player2 **flying** enabled!

Player2 move a man from (3,1) to (6,0)

(0,0)(1,1)(3,2)(3,4)(3,6)(1,5)(0,3)(1,3)(5,5)

(4,3)(3,0)(6,0)

Player1's turn.

Player1 to move a man:

Available points:

(0,0)(1,1)(3,2)(3,4)(3,6)(1,5)(0,3)(1,3)(5,5)From:

------Selected FROM Point was: ------

3,6

------Selected TO Point was: ------

3,5

Player1 move a man from (3,6) to (3,5)

Player 1 has a **MILL**!

Player2 lose a man on (4,3)

Point (4,3) **removed**\*\*\*\*\*\*\*\*\*\*

**White wins.**

(3) Test Steps for the Computer Player to Win

All of tests have been done to review the game rules applied. According to the AI, the computer try to form as many mill as possible to remove a man from opponent player; therefore, the human player just try to let computer do its job. As the test below, from phase 1 to phase 3 computer form some mill and remove a man on either placement, movement, or flying to win the game. The detail steps have been shown and form mill and to remove a man have been highlighted as the test below, and computer win the game.

Game Initializing......Done

Game started: White (Player1) first:

Start: true

Player 1: Player\_1\_Human

Player 2: Player\_2\_Computer

Player1's turn.

Player1 place a man on (0,0)

(3,0)

false

(0,0)

Player2's turn.

Player2 place a man on (3,0)

(0,0)

(3,0)

Player1's turn.

Player1 place a man on (1,1)

(6,0)

false

(0,0)(1,1)

(3,0)

Player2's turn.

Player2 place a man on (6,0)

(0,0)(1,1)

(3,0)(6,0)

Player1's turn.

Player1 place a man on (2,2)

(3,1)

false

(0,0)(1,1)(2,2)

(3,0)(6,0)

Player2's turn.

Player2 place a man on (3,1)

(0,0)(1,1)(2,2)

(3,0)(6,0)(3,1)

Player1's turn.

Player1 place a man on (2,3)

(0,0)(1,1)(2,2)(2,3)

(3,0)(6,0)(3,1)

Player2's turn.

Player2 place a man on (3,2)

Player 2 has a **MILL**!

First free point: (0,0)

Player1 lose a man on (0,0)

Point (0,0) **removed**\*\*\*\*\*\*\*\*\*\*

(1,1)(2,2)(2,3)

(3,0)(6,0)(3,1)(3,2)

Player1's turn.

Player1 place a man on (6,6)

(1,1)(2,2)(2,3)(6,6)

(3,0)(6,0)(3,1)(3,2)

Player2's turn.

Player2 place a man on (0,0)

Player 2 has a **MILL**!

First free point: (1,1)

Player1 lose a man on (1,1)

Point (1,1) *removed*\*\*\*\*\*\*\*\*\*\*

(2,2)(2,3)(6,6)

(3,0)(6,0)(3,1)(3,2)(0,0)

Player1's turn.

Player1 place a man on (5,5)

(1,1)

false

(2,2)(2,3)(6,6)(5,5)

(3,0)(6,0)(3,1)(3,2)(0,0)

Player2's turn.

Player2 place a man on (1,1)

(2,2)(2,3)(6,6)(5,5)

(3,0)(6,0)(3,1)(3,2)(0,0)(1,1)

Player1's turn.

Player1 place a man on (4,4)

(2,2)(2,3)(6,6)(5,5)(4,4)

(3,0)(6,0)(3,1)(3,2)(0,0)(1,1)

Player2's turn.

Player2 place a man on (5,1)

Player 2 has a **MILL**!

First free point: (2,2)

Player1 lose a man on (2,2)

Point (2,2) **removed**\*\*\*\*\*\*\*\*\*\*

(2,3)(6,6)(5,5)(4,4)

(3,0)(6,0)(3,1)(3,2)(0,0)(1,1)(5,1)

Player1's turn.

Player1 place a man on (3,5)

(2,2)

false

(2,3)(6,6)(5,5)(4,4)(3,5)

(3,0)(6,0)(3,1)(3,2)(0,0)(1,1)(5,1)

Player2's turn.

Player2 place a man on (2,2)

(2,3)(6,6)(5,5)(4,4)(3,5)

(3,0)(6,0)(3,1)(3,2)(0,0)(1,1)(5,1)(2,2)

Player1's turn.

Player1 place a man on (4,3)

(2,3)(6,6)(5,5)(4,4)(3,5)(4,3)

(3,0)(6,0)(3,1)(3,2)(0,0)(1,1)(5,1)(2,2)

Player2's turn.

Player2 place a man on (4,2)

Player 2 has a **MILL**!

First free point: (2,3)

Player1 lose a man on (2,3)

Point (2,3) **removed**\*\*\*\*\*\*\*\*\*\*

(6,6)(5,5)(4,4)(3,5)(4,3)

(3,0)(6,0)(3,1)(3,2)(0,0)(1,1)(5,1)(2,2)(4,2)

Player1's turn.

Player1 to move a man:

Available points:

(6,6)(5,5)(4,4)(3,5)(4,3)From:

------Selected FROM Point was: ------

3,5

------Selected TO Point was: ------

3,6

Player1 move a man from (3,5) to (3,6)

Player1 to move a man:

Available points:

(3,0)(6,0)(3,1)(3,2)(0,0)(1,1)(5,1)(2,2)(4,2)------Selected FROM Point was: ------

6,0

To:

(6,3)

Player2 move a man from (6,0) to (6,3)

(6,6)(5,5)(4,4)(4,3)(3,6)

(3,0)(3,1)(3,2)(0,0)(1,1)(5,1)(2,2)(4,2)(6,3)

Player1's turn.

Player1 to move a man:

Available points:

(6,6)(5,5)(4,4)(4,3)(3,6)From:

------Selected FROM Point was: ------

3,6

------Selected TO Point was: ------

0,6

Player1 move a man from (3,6) to (0,6)

Player1 to move a man:

Available points:

(3,0)(3,1)(3,2)(0,0)(1,1)(5,1)(2,2)(4,2)(6,3)------Selected FROM Point was: ------

3,0

To:

(6,0)

Player2 move a man from (3,0) to (6,0)

(6,6)(5,5)(4,4)(4,3)(0,6)

(3,1)(3,2)(0,0)(1,1)(5,1)(2,2)(4,2)(6,3)(6,0)

Player1's turn.

Player1 to move a man:

Available points:

(6,6)(5,5)(4,4)(4,3)(0,6)From:

------Selected FROM Point was: ------

4,4

------Selected TO Point was: ------

3,4

Player1 move a man from (4,4) to (3,4)

Player1 to move a man:

Available points:

(3,1)(3,2)(0,0)(1,1)(5,1)(2,2)(4,2)(6,3)(6,0)------Selected FROM Point was: ------

3,1

To:

(3,0)

Player2 move a man from (3,1) to (3,0)

Player 2 has a **MILL**!

Ask Player 2: to remove a man of Player 0

Computer can remove one from:

(6,6)(5,5)(4,3)(0,6)(3,4)

Select the 0 Man you want to remove while some NOthing on hand

First free point: (6,6)

Opponent is: Name: Player1 Color: white

MenHoldInHand=0 MenOnTheBoard: [(6,6), (5,5), (4,3), (0,6), (3,4)]

A good Point to Remove is: (6,6)

Player1 lose a man on (6,6)

Point (6,6) **removed**\*\*\*\*\*\*\*\*\*\*

(5,5)(4,3)(0,6)(3,4)

(3,2)(0,0)(1,1)(5,1)(2,2)(4,2)(6,3)(6,0)(3,0)

Player1's turn.

Player1 to move a man:

Available points:

(5,5)(4,3)(0,6)(3,4)From:

------Selected FROM Point was: ------

4,3

------Selected TO Point was: ------

4,4

Player1 move a man from (4,3) to (4,4)

Player1 to move a man:

Available points:

(3,2)(0,0)(1,1)(5,1)(2,2)(4,2)(6,3)(6,0)(3,0)------Selected FROM Point was: ------

3,2

To:

(3,1)

Player2 move a man from (3,2) to (3,1)

Player 2 has a **MILL**!

Ask Player 2: to remove a man of Player 1

Computer can remove one from:

(5,5)(0,6)(3,4)(4,4)

Select the 0 Man you want to remove while some NOthing on hand

First free point: (5,5)

Opponent is: Name: Player1 Color: white

MenHoldInHand=0 MenOnTheBoard: [(5,5), (0,6), (3,4), (4,4)]

A good Point to Remove is: (5,5)

Player1 lose a man on (5,5)

Point (5,5) **removed**\*\*\*\*\*\*\*\*\*\*

(0,6)(3,4)(4,4)

(0,0)(1,1)(5,1)(2,2)(4,2)(6,3)(6,0)(3,0)(3,1)

Player1's turn.

Player1 to move a man:

Available points:

(0,6)(3,4)(4,4)From:

------Selected FROM Point was: ------

3,4

------Selected TO Point was: ------

3,5

Player1 **flying** enabled!

Player1 move a man from (3,4) to (3,5)

Player1 to move a man:

Available points:

(0,0)(1,1)(5,1)(2,2)(4,2)(6,3)(6,0)(3,0)(3,1)------Selected FROM Point was: ------

0,0

To:

(0,3)

Player2 move a man from (0,0) to (0,3)

(0,6)(4,4)(3,5)

(1,1)(5,1)(2,2)(4,2)(6,3)(6,0)(3,0)(3,1)(0,3)

Player1's turn.

Player1 to move a man:

Available points:

(0,6)(4,4)(3,5)From:

------Selected FROM Point was: ------

3,5

------Selected TO Point was: ------

3,6

Player1 **flying** enabled!

Player1 move a man from (3,5) to (3,6)

Player1 to move a man:

Available points:

(1,1)(5,1)(2,2)(4,2)(6,3)(6,0)(3,0)(3,1)(0,3)------Selected FROM Point was: ------

1,1

To:

(1,3)

Player2 move a man from (1,1) to (1,3)

(0,6)(4,4)(3,6)

(5,1)(2,2)(4,2)(6,3)(6,0)(3,0)(3,1)(0,3)(1,3)

Player1's turn.

Player1 to move a man:

Available points:

(0,6)(4,4)(3,6)From:

------Selected FROM Point was: ------

4,4

------Selected TO Point was: ------

3,4

Player1 **flying** enabled!

Player1 move a man from (4,4) to (3,4)

Player1 to move a man:

Available points:

(5,1)(2,2)(4,2)(6,3)(6,0)(3,0)(3,1)(0,3)(1,3)------Selected FROM Point was: ------

5,1

To:

(5,3)

Player2 move a man from (5,1) to (5,3)

(0,6)(3,6)(3,4)

(2,2)(4,2)(6,3)(6,0)(3,0)(3,1)(0,3)(1,3)(5,3)

Player1's turn.

Player1 to move a man:

Available points:

(0,6)(3,6)(3,4)From:

------Selected FROM Point was: ------

3,4

------Selected TO Point was: ------

3,5

Player1 flying enabled!

Player1 move a man from (3,4) to (3,5)

Player1 to move a man:

Available points:

(2,2)(4,2)(6,3)(6,0)(3,0)(3,1)(0,3)(1,3)(5,3)------Selected FROM Point was: ------

2,2

To:

(3,2)

Player2 move a man from (2,2) to (3,2)

Player 2 has a **MILL**!

Ask Player 2: to remove a man of Player 1

Computer can remove one from:

(0,6)(3,6)(3,5)

Select the 0 Man you want to remove while some NOthing on hand

First free point: (0,6)

Opponent is: Name: Player1 Color: white

MenHoldInHand=0 MenOnTheBoard: [(0,6), (3,6), (3,5)]

A good Point to Remove is: (0,6)

Player1 lose a man on (0,6)

Point (0,6) **removed**\*\*\*\*\*\*\*\*

**BlackWins**

Section V. Lessons Learned (2%)

Member: Milson Munakami

(1)What did you personally gain from the project?

(2) What does your program do well, and what could your program do better?

(3) How could you improve your development process if you develop a similar game from scratch?

Member: Jimmy Wang

(1) What did you personally gain from the project?

A good time arrangement should always be helpful. For the first demo presentation we cannot demonstrate well since our GUI part is not well connected with our backend system, but we’ve made significant progress after the midterm. constructed

(2) What does your program do well, and what could your program do better?

The project work well to apply all rules, and the game can be run perfectly by two players. If there is more time, the AI placement and movement could be approved to be advance level. The AI function still meet all rules for the game and can take action. This AI works fine to make a mill and remove a man from opponent player. However, the excellent automated function need fully devotion by programmer from organized and completed algorithm. Work persistence is what we need to keep in mind for difficult job.

(3) How could you improve your development process if you develop a similar game from scratch?

For the development process, the workflow should be applied better for further project. By the experience from the mill program, the practical flow will work out well step by step. It is almost the same feeling that I have in mind at the middle of this semester. We'd better use all resource and information to let the job done. The resource include people, time, and the available information from internet. It need to be ready from the very beginning and involve to the process. The best project may not be produced by smartest people but could be made by most ambitious people to contribute all energy to fight.

Member: Sung-Ju Fan-Chiang

(1) What did you personally gain from the project?

After midterm, the project have certain progress to move a man and apply all rules to the game. However, the code did not been developed by use case diagram, but it is great to know the method to produce program step by step. The analysis workflow should be very helpful to construct large program and to maintain the program. However, even the project was not follow the workflow, the use case diagram still can match the class on Java program to figure out the workflow. Plan ahead and do the analysis should improve the reliability and correction, and the precise workflow will reduce further cost to keep the program.

(2) What does your program do well, and what could your program do better?

The project work well to apply all rules, and the game can be run perfectly by two players. If there is more time, the AI placement and movement could be approved to be advance level. The AI function still meet all rules for the game and can take action. This AI works fine to make a mill and remove a man from opponent player. However, the excellent automated function need fully devotion by programmer from organized and completed algorithm. Work persistence is what we need to keep in mind for difficult job.

(3) How could you improve your development process if you develop a similar game from scratch?

For the development process, the workflow should be applied better for further project. By the experience from the mill program, the practical flow will work out well step by step. It is almost the same feeling that I have in mind at the middle of this semester. We'd better use all resource and information to let the job done. The resource include people, time, and the available information from internet. It need to be ready from the very beginning and involve to the process. The best project may not be produced by smartest people but could be made by most ambitious people to contribute all energy to fight.