**A. Source Code and URLS**

Source Code: see sourceCode.pdf

URLS: (list AI URLS)

**B. Requirements**

Nonfunctional requirements:

-The game engine shall be written in Java.

-The Java GridWorld UI and API shall be used by the game engine to display the game board

-Messages shall be displayed in the GridWorld message field.

-The game engine shall support AI players written in PHP or python.

-The game engine shall use HTTP POST to send JSON data to the AIs.

-The AIs' responses shall also be JSON.

Functional requirements:

-The game engine shall send each player's Web AI data on both teams' pieces' locations, destinations, and damaged pieces, as well as the board's size.

-Formatted as:

{

"boardSize":18,

"pieces":[{"x":0,"y":0,"damage":0,"team":0},{"x":1,"y":1,"damage":1,"team":0}],

"destinations":[{"x":0,"y":0},{"x":1,"y":1}],

"enemy":[{"x":0,"y":0,"damage":0,"team":1},{"x":1,"y":1,"damage":1,"team":1}],

"enemydestinations":[{"x":0,"y":0},{"x":1,"y":1}]

}

-The game engine shall receive data from each AI on its next move, including where it is moving from and to.

-Formatted as:

{

"from":{"x":0,"y":0},

"to":[{"x":1,"y":1},{"x":2,"y":2}]

}

-"to" field shall consist of sequence of jump moves

-The game engine shall repeat a cycle of sending the teams their data, receiving information on each team's next move, verifying each move's validity, and performing the moves.

-The game engine shall make 1 move at a time for each team.

-If either team has not submitted a move, the game engine shall not enact either move.

-The game engine shall ensure all submitted moves are valid by the rules of Halma.

-If either team submitted an invalid move, the game engine shall not enact either move, and the UI shall display an error.

-The rules are available at: http://lyle.smu.edu/~coyle/halmagame/halma1.0/canvas.html#halma

-The AIs shall only send information for a single move of a single piece at a time.

-Otherwise, the move shall be considered invalid, so the game engine shall not enact either player's move, and the UI shall display an error.

-Upon a player's victory, the UI shall declare "halmate" and stop requesting moves.

-Messages displayed by the UI shall include the most recent moves made, the number of turns elapsed, any errors that have occurred, and if "halmate" has occurred.

-Each team shall have different colored pieces.

Collision Rules:

-Intentional collisions onto already occupied squares will result in:

Damage = 5 for enemy piece

Damage = 0 for home piece

-An accidental head-on collision is when both teams move to the same square on the same turn:

Damage = 5 for both pieces

-Repeat collisions will result in the "damage" count being reset to 5.

-The damage count will decrement upon each successful move with a lower limit of 0.

-Upon reaching damage of 0, the piece is able to jump again.

-Collisions shall be determined by matching destination squares only, not by intermediate jumps.

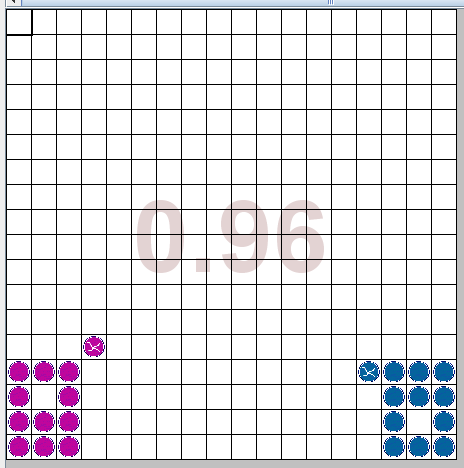
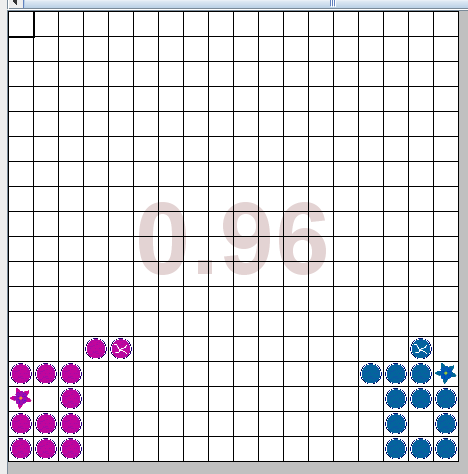
-Colliding pieces shall occupy the same square.

-Collided pieces shall be colored black.

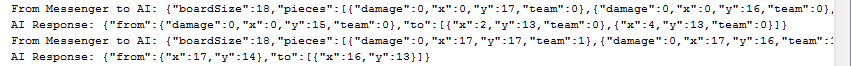
**C. Design Document**

**D. Test Results**

The following is a test to see if the board updates properly based on data sent by the AIs. You can see the data sent from the game engine to each of the AIs, as well as each of their responses.



Board Before Board After

Game Engine and UI communication:

**E. Project Repository**

<https://github.com/vipulkohli/JHalma>