Contrasting Chess

Requirements and Test Document

Troy Neubauer

Date 3/19/2021

CS 225, Spring 2021

Embry-Riddle Aeronautical University

Daytona Beach campus

1 Aerospace Boulevard

Daytona Beach, FL 32114

**Introduction:**

Contrasting Chess is a turn based strategy game similar to chess, where users can play with other players on the same computer or against a simple AI.

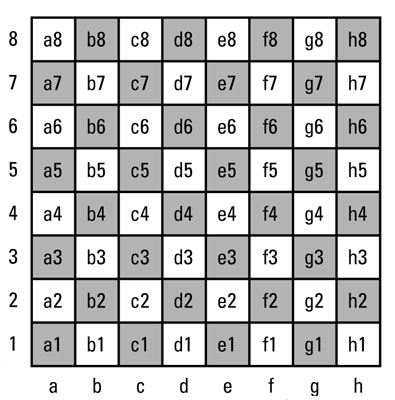
**Background information:**

Contrasting Chess is built off of chess but with different pieces, a diffirent board size, and a different starting position. However it barrows from chess in a lot of ways. Players take turns picking a piece to move and then following its rules to determine how to move it. You cant pass on your turn and you objective is to make an unstoppable threat to capture your opponents the king (checkmate, where the king is attacked and has no legal moves that shield him from attack).

The following bullet points outline the moves for Constasting Chess’s pieces:

* King
  + Can move or capture one square in any direction as long as the target square is not attacked by an opponent’s piece
  + Can only move diagionally if the king did not move on the last move
* Elephant
  + Con move or capture any number of squares either vertically, horizontally or diagionally. Cannot capture rodents
* Bear
  + Can move or capture horizontally or vertically up to 4 squares from the starting square
* Horse
  + Can move or capture 2 squares in one direction plus 1 square in a the perpendicular direction (same as chess).
* Dragon
  + Can move or capture diagionally any amount of squares
* Moose
  + Can move diagionally forward up to 2 squares or capture square forward
* Rodent
  + Can move 1 square forward any move or 2 squares when more than 3 squares away from the opponent’s back rank, ar 3 squares forward on the piece’s first move. Can capture one square forward and diagionally.
* All Rules
  + Any move that results in stalemate (a situation where a player has no legal moves) is not allowed

The picture below shows which squares are which on a chess board. This same notation well be used in Contrasting Chess, but extended up to J10 to allow for a 10x10 board. This notation will became relevant for test cases.



**Requirements:**

This section actually contains the requirements. Remember that requirements must be atomic (requiring only one thing) and testable (able to be objectively measured via test). The section starts with a short description of the purpose of the section (presenting the requirements), and any other additional text that helps the reader understand what the purpose of the requirements are. You must have at least five requirements. You may have more, but your grade is based on five. The requirements shall be numbered or identified in some fashion. Each requirement shall be associated with the user story, or stories, that it relates to. Formats may vary, but a numbered list or table is typically a good way to present this kind of information.

The basic requirements are listed below. As this game relies heavily on the movement rules for each piece, the requirements lean heavity on verifying these rules sometimes in a somewhat arbutary fashon. Each subsection of a rule (1. A rodent can move 3 squares the first time it moves 2. A rodent can move 2 squares when more than 3 squares away from the opponents back rank 3. Or a rodent can move 1 square forward), each of these is a movement rule subsection and will be tested in the end with a positive, negitive and neutral test case, possibly meaning that a single piece has 9+ test cases to check its movement rules. Therefore I have listed each piece’s movement as a high level requirement evene thougth it technically consists of mutiple sub parts because otherwise I would only be able to fit 2 or so pieces into the 5 requirements.

1. The program shall give users the ability to move pieces using the mouse | User story(ies) 1, 2, 3

2. The King follows the above rules for movement | User story(ies) 6

3. The Elephant follows the above rules for movement | User story(ies) 6

4. The Bear follows the above rules for movement | User story(ies) 6

5. The Horse follows the above rules for movement | User story(ies) 6

6. The Dragon follows the above rules for movement | User story(ies) 6

7. The Moose follows the above rules for movement | User story(ies) 6

8. The Rodent follows the above rules for movement | User story(ies) 6

9. The program shall utalize Rust to store moves | User story(ies) 3, 4,

10. The program shall utalize Rust to calculate legal moves | User story(ies) 4, 6, 7

**Test Cases:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Req’t**  **ID** | **Test**  **Case**  **ID** | **Initial**  **Conditions**  **And Input** | **Expected Behavior**  **Or Output** | **Actual**  **Behavior**  **Or Output** | **Pass**  **Fail** |
| 1 | 1A | A rodent is on square A2. The user clicks on A2 and then on A3 | The rodent piece moves to the A3 square on the board |  |  |
| 1 | 1B | A rodent is on square A2. The user clicks on A2 and then again on A2 | No action is taken because this move is invalid |  |  |
| 1 | 1C | A rodent is on square A2. The user clicks on A2 and then a position off the board | No action is taken because this move is invalid |  |  |
| 2 | 2A | A king on A1 | The king can be moved to A2 |  |  |
| 2 | 2B | A king on A1 | The king cannot me moved to A4 |  |  |
| 2 | 2C | An opponent’s elephant on B3. King on A1 | The king has no legal moves since all possible squares are covered by the queen |  |  |
| 3 | 3A | Elephant on A1 | The elephant can be moved to A10 |  |  |
| 3 | 3B | Elephant on A1 | The elephant can be moved to J1 |  |  |
| 3 | 3C | Elephant on A1 | The elephant can be moved to J10 |  |  |
| 4 | 4A | Bear on A1 | The bear can move to A5 |  |  |
| 4 | 4B | Bear on A1 | The bear cannot move to B2 |  |  |
| 4 | 4C | Bear on A1 | The bear cannot move to A6 |  |  |
| 5 | 5A | Horse on A1 | The horse can move to B3 |  |  |
| 5 | 5B | Horse on A1 | The horse can move to C2 |  |  |
| 5 | 5C | Horse on A1 | The horse cannot move to C3 |  |  |

**References:**

<https://youtu.be/U4ogK0MIzqk> - Coding Adventure: Chess AI | Sebastian Lague

**Appendices:**

None as of P3