Chessinator Outline

December 27th, 2024

Zach Pittman

Abstract:

“Chessinator” (placeholder) is a web-based application that combines the strategic elements of chess and Stratego. Here, I outline the core mechanics, programming language and resources.

CORE MECHANICS

In the most basic mode, the game is played on a standard chess board. Each player starts with the following pieces:

- 4 Pawns
- 1 King
- 1 Queen
- 1 Rook
- 1 Bishop or knight

Each game is played in two phases: 1) Placement phase, 2) Playing phase.

Placement Phase

At the start of the game, each player places a single piece on the board, alternating turns. Players may play pieces in any order. Pieces may be placed anywhere in the first two ranks (pawns can go in your back row, anything can go in the front row). Your opponent cannot view which pieces you have placed, only their location (see figure 1).

Playing Phase

After all material has been placed, the game resumes as a normal chess game. Pieces can only move as they would in standard chess.

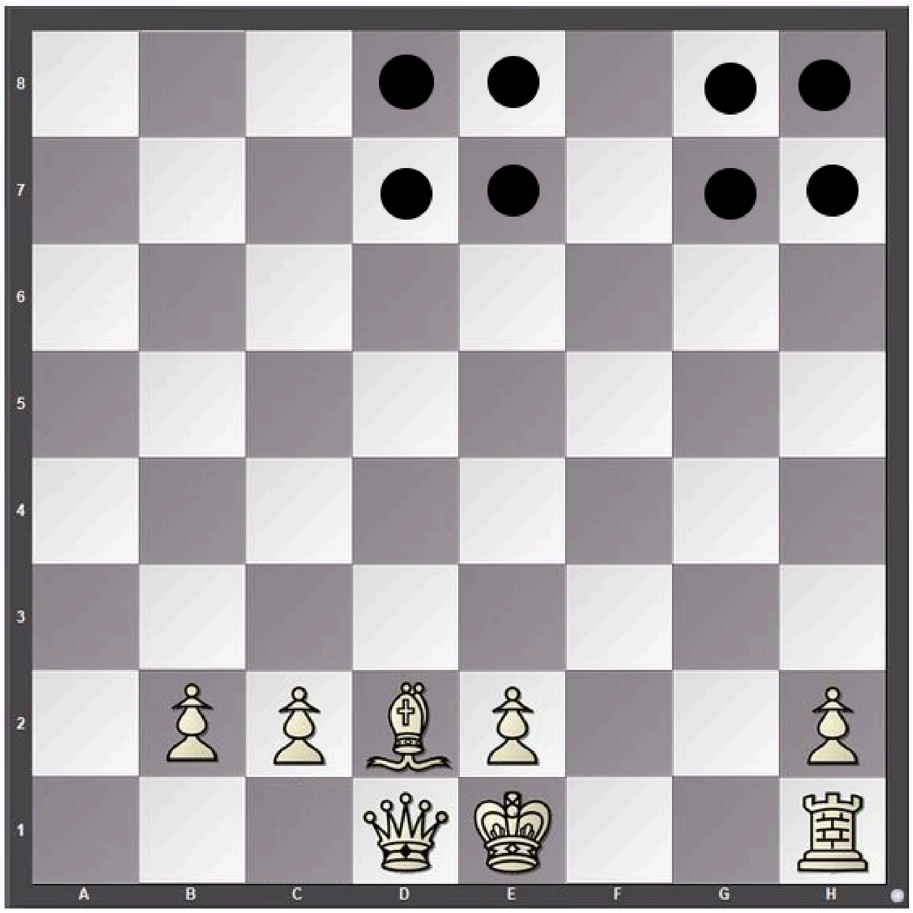


Figure 1. Player’s view of the board and view of opponent’s pieces at start of game.

Because the position of the king is unknown, you as a player must use deductive reasoning to determine which of your opponent’s pieces is the likely king piece. Additionally, it becomes possible to move your king into check, either by bold strategy or blunder. The game ends when either king is taken.

The graphical user interface will assist in keeping track of your guesses as to which pieces are which, much like an online Sudoku.

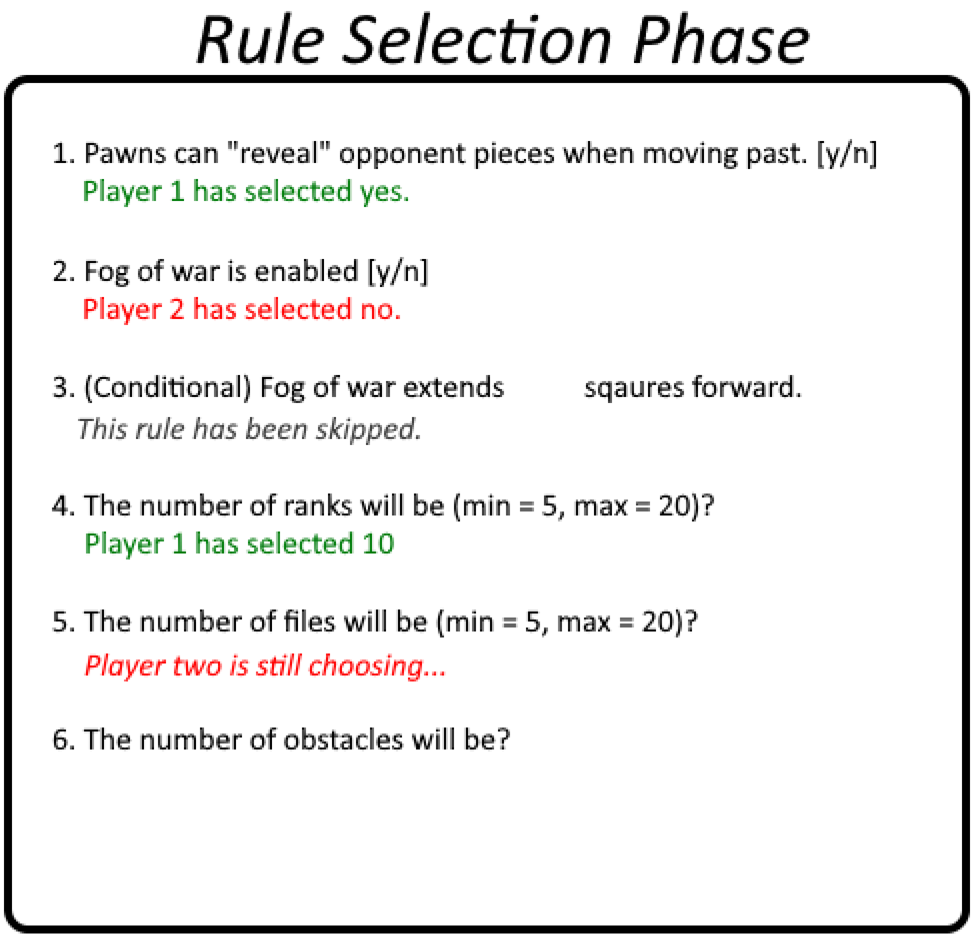
This game will have multiple play modes. Right now there are “basic” and “advanced” modes.

BASIC MODE

The basic mode operates exactly as described in the CORE MECHANICS section. No deviation.

ADVANCED MODE

The advanced mode adds several features. The most notable feature will be an additional "game setup" phase, where each player determines the rules that dictate board dimensions, obstacles, movement restrictions, fog of war, and more. Similar to the placement of pieces, players decide the outcome of each rule choice in alternating turns. For example:



Rule Selection Phase

1. Pawns can "reveal" opponent pieces when moving past. [y/n]
   Player 1 has selected yes.

2. Fog of war is enabled [y/n]
   Player 2 has selected no.

3. (Conditional) Fog of war extends sqaures forward.
   This rule has been skipped.

4. The number of ranks will be (min = 5, max = 20)?
   Player 1 has selected 10

5. The number of files will be (min = 5, max = 20)?
   Player two is still choosing...

6. The number of obstacles will be?

Figure 2. An example of the content of the "Rule Selection Phase" dialog box.

Take note, that some rules depend on the outcome of upstream rules, like rule 3 in Figure 2. It could become a strategy to choose the outcomes of upstream rules for the sole purpose of being selected as the player to choose the outcome of downstream rules that are more important to executing your playing strategy.

Each rule phase will have 10 rules. Those 10 rules will be randomly selected from a larger list of up to 30 possible adjustable rules (not devised yet).

PROGRAMMING LANGUAGE AND RESOURCES

1. Web socket for lobby and matchmaking and backend management (socket.io)
2. React for building web components (good for low graphic web games)
3. Can use Amazon web service to host when we want to go live.
4. Github page.

There is a near-infinite supply of tutorials for socket.io, React, and socket.io/React applications. The backend development of this game will be, by orders of magnitude, the most difficult part of this project. Mostly, learning how to properly build React components to interact with server-based information. The programming of the logic of the game itself is fairly straightforward.

The outline for structuring the code is a big lift, so this is mostly just to get us going right now!