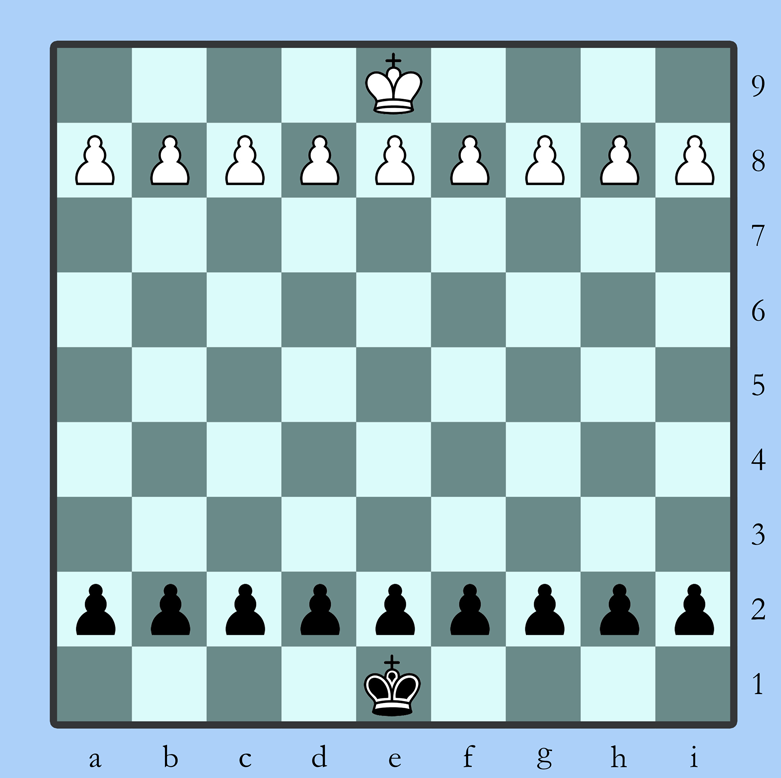
# GREAT WAR

a deck-building chess game

Chess is a deep and magnificent game, and it is nearly solved by computer science. There isn’t a human being alive who can beat the best AI chess engines. Great War introduces new gameplay elements into the framework of chess—chance, hidden information, and great variety, primarily—which can erase much of the edge that computers have over people.

## Overview

The standard game of Great War begins on a 9x9 chessboard, with numbered rows called **ranks** and letter-ordered columns called **files**. The game begins with one king in the central file of each player’s back rank and a row of pawns in each player’s second rank, as below:



To provide the rest of the pieces, players start with a hand of five **cards** from a pre-built **deck**. Cards work primarily to install pieces on the board, although some cards can be used to alter play in other ways. With a few exceptions, pieces can only be played in a player’s back rank.

Each player’s **bank** determines which cards can be played at a given time. Players start with nine **points** in their bank, and each player’s bank increases by .5 after their turn. Points are also added when piece captures take place, and on certain other special conditions.

On a player’s turn, they may choose to either **play** a card or **move** a piece on the board. The pieces and cards can influence each other: some cards may enhance the movement of particular pieces, or a certain number of the same type of piece on the board may affect the point cost of cards of the same type.

The list of **pieces** available to players is extensive. A selection of examples is listed at the end of this document, along with diagrams of the range in which they may move or capture.

Games are played to a **clock**, with terms agreed by players before the game starts. The standard time control is 45 minutes for a game, with a bonus of ten seconds per move, but games can be set to last any length of time.

The game ends when a player chooses to **resign**, one player forces **checkmate** on the other, or both players agree to a **draw**.

# MECHANICS

The basic business of the game relies on some pretty common programming concepts. There should be plenty of existing code to draw from, especially for the chess parts. Here’s a [“pure chess library”](https://github.com/niklasf/python-chess) in Python and a (presumably) [working two-player game](https://gist.github.com/geocachecs/d8d2f402b0843231231b) written in C++, for example.

Examples of card-battle games are common on github as well, although without compiling/running them I can’t say if the code would necessarily be the right fit. Here are a [couple](https://github.com/hamdyaea/Battle-Cards-with-GUI) of [examples](https://github.com/reinaldons/card-battle) written in Python; more are available.

## Chess parts

* **Board**
  + 81 squares in a 9x9 grid. Players will ultimately have the ability to customize the board, but the standard size is good for prototyping.
* **Turns**
  + Players can either move a piece or play a card each turn.
* **Piece movement (square-restricted)**
  + This is the L-shapes for knights, the diagonals for bishops, 2/1 squares for first-rank pawns, castling, etc.
* **Promotion**
  + In chess, pawns that reach the far end of the board may **promote** to other pieces. As well as classical pawn promotion, Great War allows exotic promotion conditions for a variety of pieces, created by specific circumstances, special effects or other pieces on the board.
* **Check and checkmate**
  + When a player’s king is in check, the only legal moves should be those that either move the king, move a friendly piece into the path of the check, or capture the enemy piece. If no legal moves exist, the game ends.
* **Timer** 
  + Each player has a timer that runs during their move. When a move is made, the opponent’s timer starts running. This will eventually be necessary but is probably a low priority in early-stage development.

## Card parts

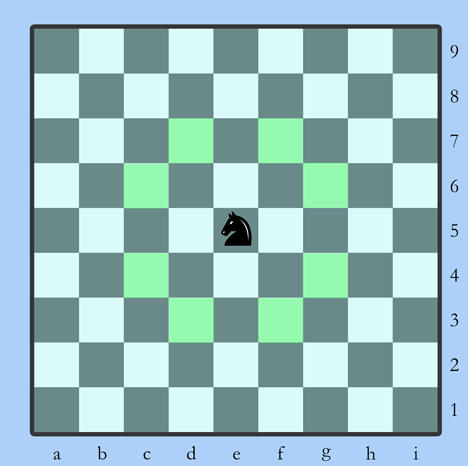
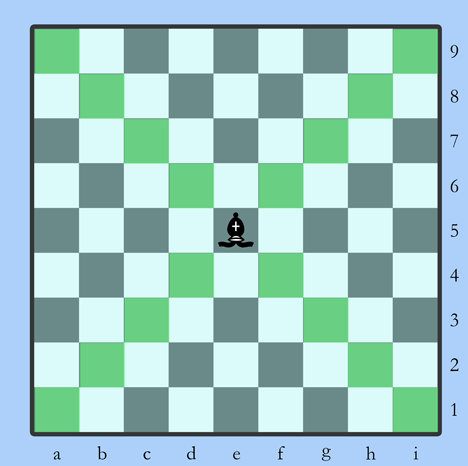
* **Cards**
  + Players accumulate cards through in-app events and transactions and keep them in a personal collection.
* **Decks** 
  + Players choose a selection of cards from their collection to build a deck that can be actively played during the game.
* **Hands**
  + Players start with five cards apiece, drawn from their shuffled deck. If a player has fewer than five cards in their hand at the beginning of their turn, they may draw one card from the top of the deck and add it to their hand.
* **Piece placement (square-restricted)**
  + Most cards are piece cards, which allow players to bring new pieces into battle. Most pieces can only be deployed in the player’s back rank, although some exceptions can be placed at different specific locations around the board.

### Other parts

* **Points**
  + All cards have a point cost that must be paid from the player’s bank in order to take effect. The amount of points in the bank can increase in two ways: a regular increment of 0.5 is added each turn, and half the value of a captured enemy piece is added upon capture.
* **Effects**
  + Some cards or pieces can influence the board or other pieces, often for a limited time or in a limited space. A freeze effect can immobilize a piece on a given square; a rock slide effect can create a square that is permanently impassible; a piece called the Bodyguard can prevent opposing pieces from crossing within one square of it; etc.
* **Treasure**
  + Treasure is a form of in-app currency that’s used to purchase card packs and pay tournament entry fees, as well as acquire accessories like custom piece skins. A prototype can easily go ahead without this, but it’s here for context.
* **Card trading (network)**
  + Players can make deals to trade cards with one another. This, also, is mainly here for description purposes.

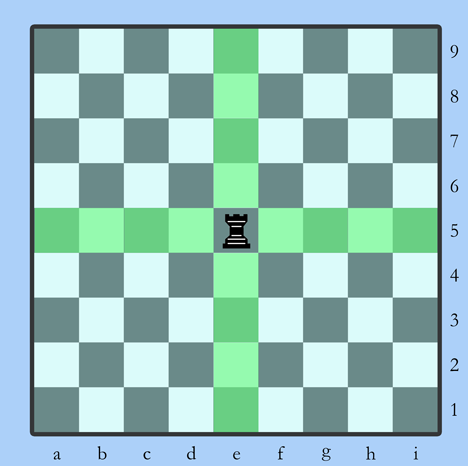
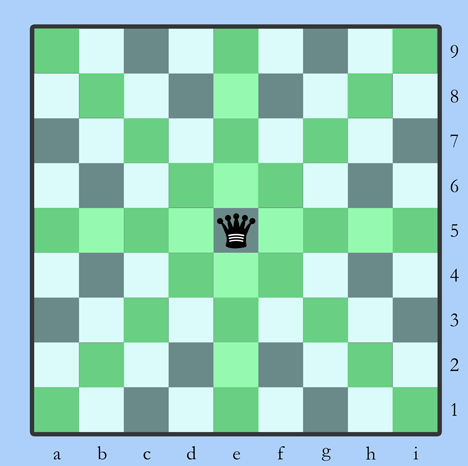
There will also, ultimately, need to be an AI capable of moving pieces and playing cards in a sensible way and user account system that tracks card collections and other player data, but that sort of thing is definitely not necessary for a working demo build.

# PIECE INDEX

**Knight**: moves and captures by leaping **Bishop**: moves and captures by sliding any

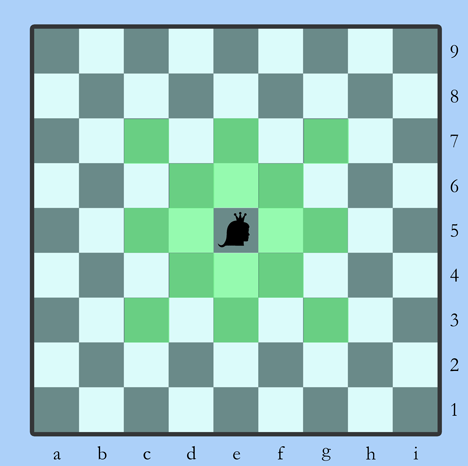
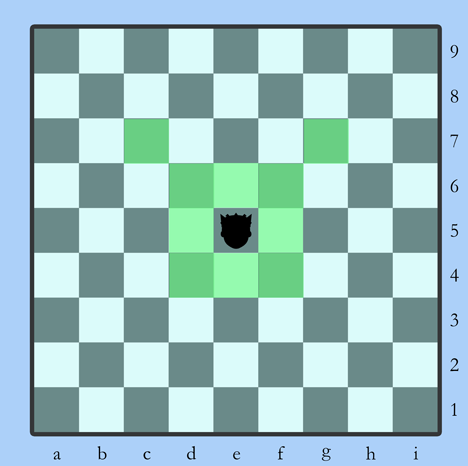
at a 2-1 right anglenumber of squares diagonally

**Rook**: Moves and captures by sliding any number **Queen**: moves or captures any number of

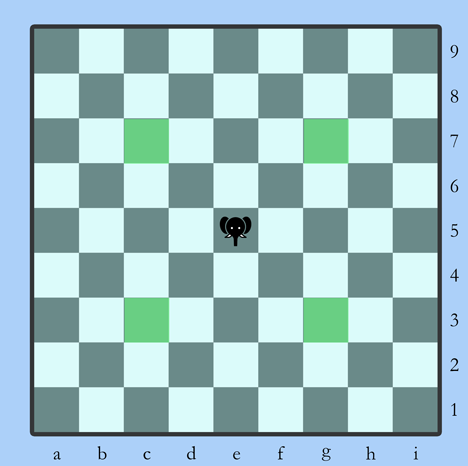
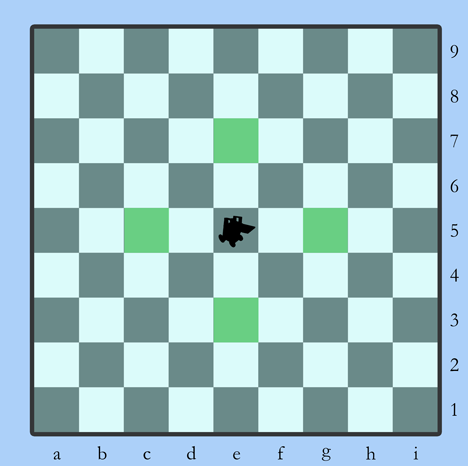
of squares orthogonally (sideways, forward or squares in any direction

backward)

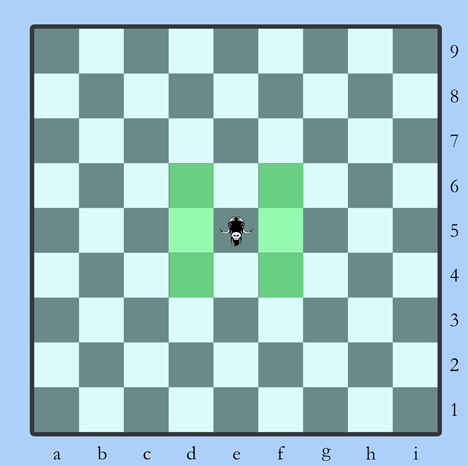
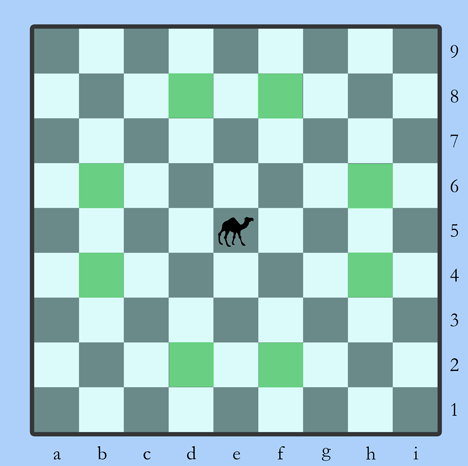
**Princess**: moves and captures up to two squares **Prince**: moves and captures one square in any

in any direction direction or two squares diagonally forward

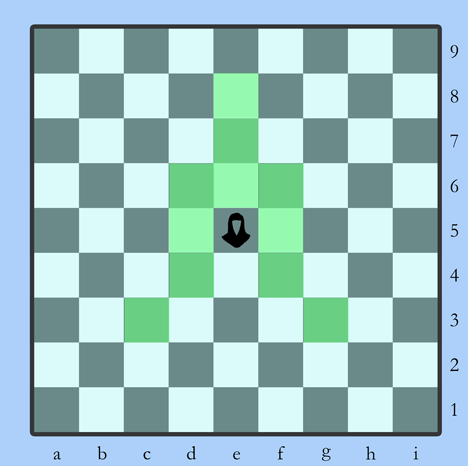
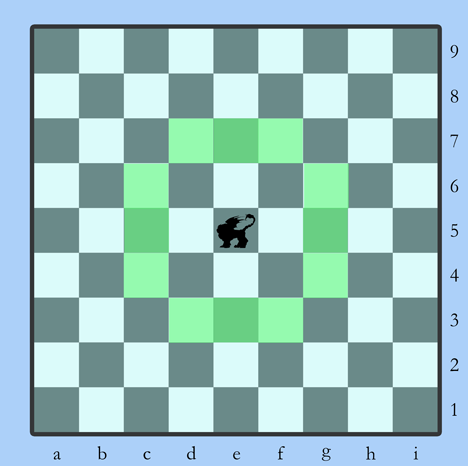
**Alfil**: moves and captures by leaping exactly **Dabbaba**: moves and captures by leaping

two squares diagonally exactly two squares orthogonally

**Ox**: moves and captures one square sideways **Camel**: moves and captures by leaping at a 3/1

or diagonally right angle

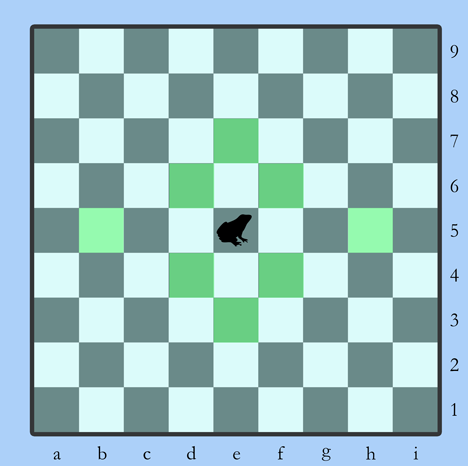
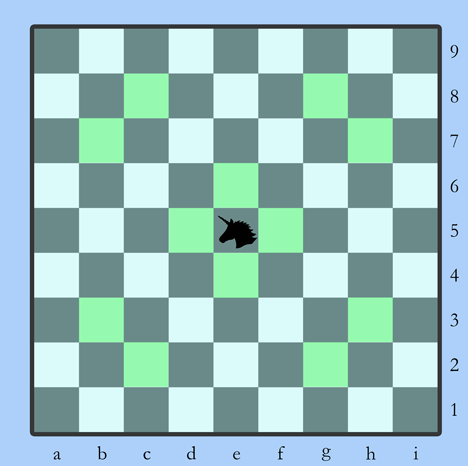
 

**Zealot**: moves and captures up to three squares **Manticore**: moves and captures like a knight or

straight forward, two squares diagonally dabbaba

backward or one square diagonally forward or

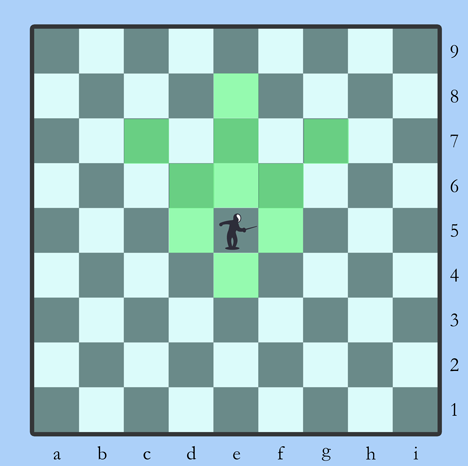
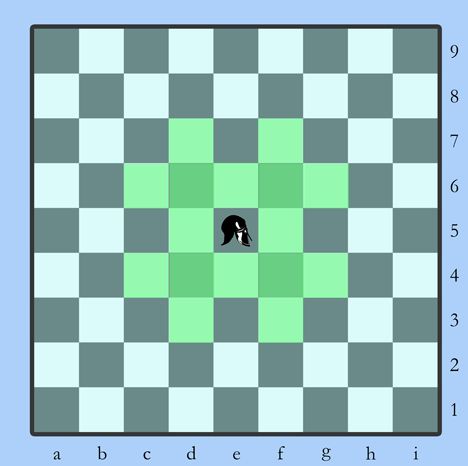
sideways

**Frog**: moves and captures by leaping one square **Unicorn**: moves and captures by leaping at a

diagonally, exactly two squares forward and 3/2 right angle or one square orthogonally

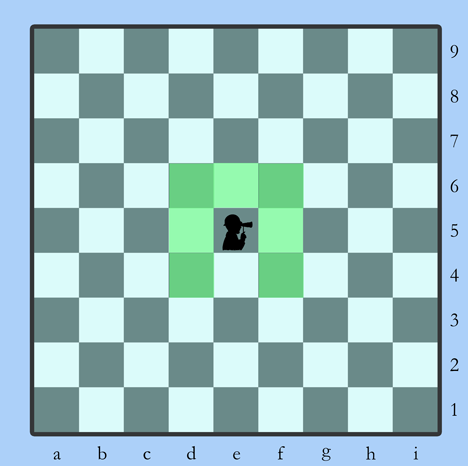
backward or exactly three squares sideways

**Fencer**: moves and captures up to three squares **Veteran**: moves and captures like a king or a

straight forward, up to two squares diagonally knight

forward or one square backwards or sideways



**Analyst**: moves and captures one square in any

direction except straight backward; pawns in a

one square radius may move (but not capture)

like their designated promotion piece(s) for

one turn