**Software Design  
Document**

for

Ferret Army Chess

Version 1.0 approved

Prepared by A. Maxwell, J. Guerrero, A. Romualdo, J. Cole, M. Yi

Ferret Army

February 2018

# 

# Table of Contents

Table of Contents i

Revisions i

1. Introduction 1

1.1 Purpose 1

1.2 System Overview 1

1.3 Definitions, Acronyms and Abbreviations 4

1.4 Supporting Materials 5

1.5 Document Overview 6

2. Architecture 6

2.1 Overview 6

2.2 User Interface (UI) 7

2.3 Game Engine (GE) 7

2.4 Artificial Intelligence Engine (AIE) 7

3. High-Level Design 8

3.1 View / Model Component 1..n 8

# 

# Revisions

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Primary Author(s) | Description of Version | Date Completed |
| 1.0 | Alexander Maxwell | Completion of *section 1* including sub sections 1.1, 1.2, 1.3, 1.4, and 1.5. | 02/19/2018 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Introduction

## Purpose

The purpose of this Software Design Document (SDD) is to provide a detailed description of the implementation of the "Ferret Army Chess (FAC)" software. It illustrates the design patterns, components, and units involved with creating an interactive version of chess called FAC. This document was generated according to the Software Requirement Specification (SRS) document *version 2.5* agreed upon with the client. The SRS document can be found included with the final deliverables for this project.

## System Overview

The FAC software is broken into three key components, *game engine* (GE), *user interface* (UI), and *artificial intelligence engine* (AIE). These components interact throughout the course of a chess game. The design of FAC largely revolves around the definitions, design, and implementation of GE, UI, and AIE. The architecture of the GE, UI, and AIE will be discussed later in *section 2.*

In accordance with the SRS the FAC software shall have no network functionality. The software shall run locally on desktop / mobile computers with either Windows or macOS operating systems. Due to the locality of the FAC software there will be no external interaction with other systems or actors beyond the *user*. A detailed description of the requirements can be seen in the SRS and will be discussed as required in later sections of this document. The general functional requirements can be seen, in bulleted form, below:

* FAC software shall provide users with three game modes including computer vs. computer, user vs. computer, user vs. user prior to gameplay.
* System shall implement a user interface (UI) allowing the user(s) to select game mode and settings.
* FAC software shall include functionality for limited artificial intelligence engine (AIE). Using randomization, the FAC software’s AIE shall move game pieces (GP) around the game board (GB).
* In addition to the standard chess moves the FAC software shall allow for three special moves including:
  + En passant – Special pawn capture move.
  + Pawn Promotion – From a player’s pawn to any other available game piece (GP). An available GP is defined as a GP previously captured by an opponent.
  + Castling – Both Queenside and Kingside.

These special moves shall be available to players only when specific conditions are met. The detailed description of each special move and the specific conditions to be met are covered in-depth in section 4.

* FAC Software shall provide functionality for user(s) to be able to move any game piece (GP) according to that GP’s specific game move (GM) attribute. These moves are detailed in-depth in TABLE 1.

TABLE 1.

*Game Pieces* and their associated *Game Moves*

|  |  |  |
| --- | --- | --- |
| **Game Piece** | **Game Move** | **Capture** |
| Pawn | **Forward 1 space**  **Forward 2 spaces** (Starting move only) – Movement cannot cause collision with another piece.  If the option for pawn promotion is chosen, then player can choose the piece the pawn will be promoted to after it reaches the last row of the opposing players side.  **Restrictions:**  Movement cannot extend past the edge of the game. | Left Diagonal 1 space  Right Diagonal 1 space  **Special case:**  En passant Capture – left or right diagonal 1 space  (See specifics in section 4.5) |
| Rook | **Forward 1-7 spaces**  **Backward 1-7 spaces**  **Left 1-7 spaces**  **Right 1-7 spaces**  **Restrictions:**  Movement is unrestricted until another game piece is encountered, or edge of game board is reached.  **Special case:**  Simultaneous movement with King is allowed for castling.  (See specifics detailed in section 4.7) | Same as game move until an opponent’s piece is captured. |
| Bishop | **Diagonal 1-7 spaces** on the game pieces color of origin  **Restrictions:**  Movement is unrestricted until another game piece is encountered, or end of game board is reached. | Same as game move until an opponent’s piece is captured. |
| Queen | **Diagonally 1-7 spaces**  **Vertically 1-7 spaces**  **Horizontally 1-7 spaces**  **Restrictions:**  Movement is unrestricted until another game piece is encountered, or end of game board is reached. | Same as game move until an opponent’s piece is captured. |
| King | **Diagonally 1 space**  **Vertically 1 space**  **Horizontally 1 space**  **Restrictions:**  Cannot move into a position that will place it within 1 space of the opponents King.  Cannot move into a position that will place it in check.  Movement cannot exceed the perimeter of the board.  **Special case:**  Castling will allow movement greater than 1 space along the 1st rank.  (See specifics in section 4.7) | Same as Game Move until an opponent’s piece is captured. Must not be in check when Capture completed. |

* The GB shall include game coordinates, so a user can submit moves using a coordinate on the GB. This will be represented as numbers for the rank (horizontal coordinates) and letters for the file (vertical coordinates).
* Users shall have the option to quit an ongoing game at any time. It is not necessary for both players to agree before an individual player quits a game.
* FAC software shall provider user(s) with the ability to enable a timer to control the flow of the game. The time shall have functionality to support the following operations:
  + Game Time Limit – Set duration for the length of a single game.
    - Winner will be determined by the total points accumulated from the capture of the opposing teams game pieces. See section 4.8 for more on stalemate resolution.
  + Turn Time Limit – Set a duration for the length of a single turn.

## Definitions, Acronyms and Abbreviations

TABLE 2.

Definitions, Acronyms, and Abbreviations

|  |  |
| --- | --- |
| **Term** | **Definition** |
| AIE | Refers to artificial intelligence engine that makes moves for the computer. |
| Bystander | A user who is observing a computer vs. computer game without making a game move. |
| Capture | The act of a player removing another player’s game piece by replacing their opponent’s game piece with the attacking game piece thus Capturing said game piece. |
| Check | Refers to a game move where a player’s King is under attack from another player whether a user or the computer. |
| Checkmate | Refers to a game move where a player’s King has no remaining moves where said game piece is not under attack from another player whether a user or the computer. |
| Computer | The system that represents the artificial intelligence of which a user can compete against. |
| FAC | Refers to Ferret Army Chess the software under development. |
| File | The columns of the chessboard that run vertically and are referred to by letters. |
| GB | Refers to the Game Board which is comprised of an 8 square by 8 square board with alternating colors which total 64 possible squares a game piece may occupy. |
| GE | Refers to the game engine which is collectively the code that runs the game pieces, game moves, and Game Board. |
| GM | Refers to game move which is the act of moving a game piece on the Board. |
| GP | Refers to any of the game piece(s) which may be a Pawn, Rook, Bishop, Knight, Queen, or King |
| GPA | Refers to the game piece array that hold the active state of each piece in a FAC game. |
| Major Piece | Refers to specifically to the queen or rook game pieces. |
| Minor Piece | Refers to specifically to the bishop or knight game pieces. |
| Player | A user who has initiated a game against either another user or the computer. |
| Rank | The rows that go from side to side across the chessboard and are referred to by numbers. |
| Reinfeld Value | The numeric value assigned to each game piece the values are as follows: Pawn (1), Bishop (3), Knight (3), Rook (5), Queen (9) |
| SDD | Software Design Document |
| TP | Refers to the Test Plan used to test the functionality of FAC. |
| UI | Refers to interface by which the user interacts with the FAC software. |
| User | A person interacting with the FAC software. |
| X-COR | Refers to the X Coordinate in the coordinate pair (X,Y) that represents a game pieces location on the game board. |
| Y-COR | Refers to the Y Coordinate in the coordinate pair (X,Y) that represents a game pieces location on the game board. |

## Supporting Materials

<Note any references or related materials here.

## Document Overview

This Software Design Document (SDD) is divided into three sections with various subsections. The major sections of this SDD are as follows:

1. Introduction – Structure of the SDD.
2. Architecture – In-depth look at the structure of the FAC software.
3. High Level Design – Overview of each component.

In *section 1. – Introduction*, the purpose, methodology, and reason for this Software Design Document is introduced. In *section 2. – Architecture,* the system design, component design, sub-component design, and class design is discussed in-depth. In *section 3. – High Level Design*, an overview of the structural and functional decomposition of the systems components is discussed. In addition, the general interaction between the system’s components and the *user* is discussed at an abstract level.

# Architecture

## Overview

As previously mentioned in section 1.2 the FAC software is broken down into three key components, the *game engine* (GE), user *interface* (UI), and artificial *intelligence engine* (AIE). Each of these primary components can be further divided into several sub-components. An overview of the FAC software at a component/sub-component level can be seen in figure 1.

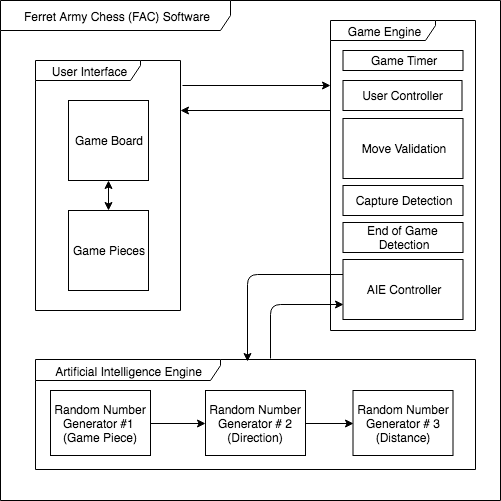


Figure 1. FAC Software Overview

The FAC software was decomposed into the GE, UI, and AIE for several reasons. In order to play a game of chess there are several key components including: *game board* (GB), *game pieces* (GP), *players* (2 *users*), the *computer*, and associated game winning mechanics. Of the key components *users* need only be aware of the GB and GP’s to play FAC. The remaining key components are the “under-the-hood” elements supporting the transaction between the GB, GP’s, and *user*.

Since FAC shall allow a *player* to be either a *human user* or a *computer user* the need arises for the FAC software to support some form of artificial intelligence when a *human user* is playing a game against a *computer user*. In an effort to support modularity it was beneficial to break the *user interface* (UI) elements from the game mechanics including the artificial intelligence engine thus leading to the UI, GE, and AI.

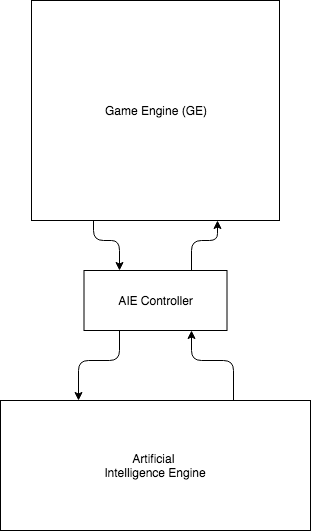


Figure 2. AIE Controller

To support future improvements to the artificial intelligence of FAC the AIE was separated from the GE. Since it is highly unlikely that interactions between the UI and GE will change during maintenance the compartmentalization of the AIE leads to greater maintainability. When the AIE ability is improved there will be little refactoring involved since all interaction between the GE and the AIE run through the AIE controller. This means that changes to the AIE will almost be “plug-in-play” since all calls from the GE to the AIE will remain unchanged. This type of interaction leads to an adapter design pattern for the AIE and GE. The interactions between the GE and AIE will be discussed in detailed later in this document.

## User Interface (UI)

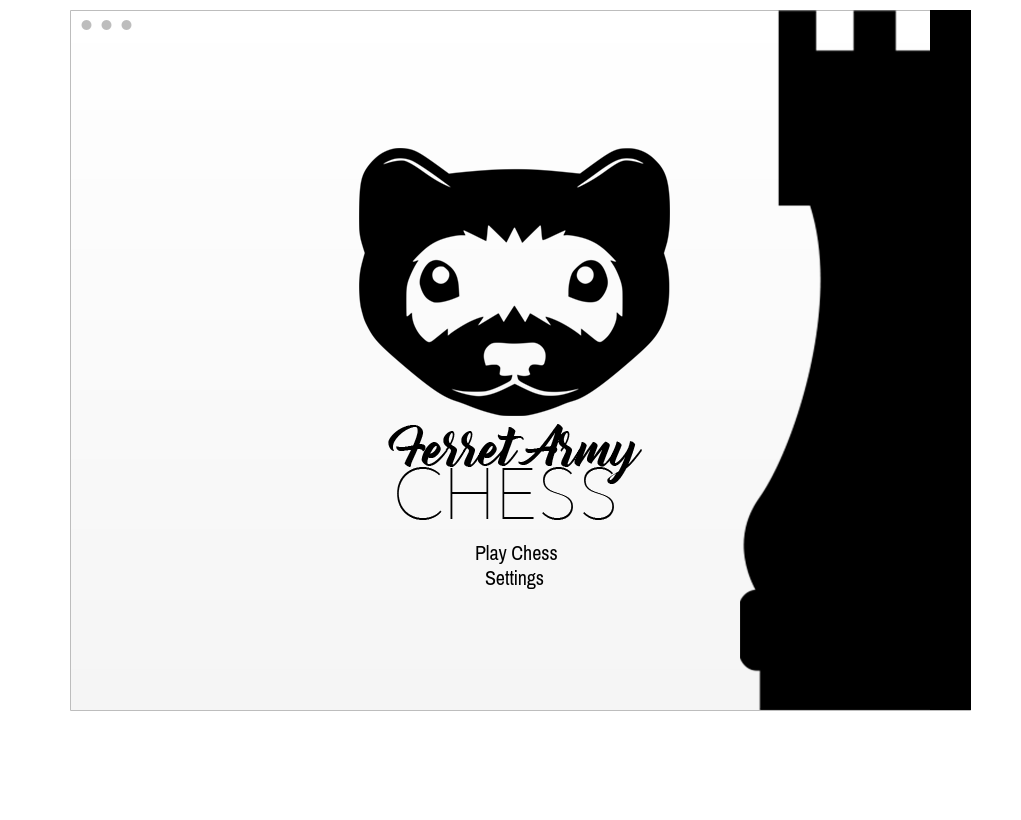


Figure 3. Landing Page

The *user interface* (UI) is broken into several key sub-components. As previously mentioned the two primary components are the *game board* (GB) and the *game pieces* (GP’s). Additionally, there are several other UI elements that encompass the FAC software. These include the:

* *landing page,* which is the first screen a user sees when launching the FAC software, see figure 3.

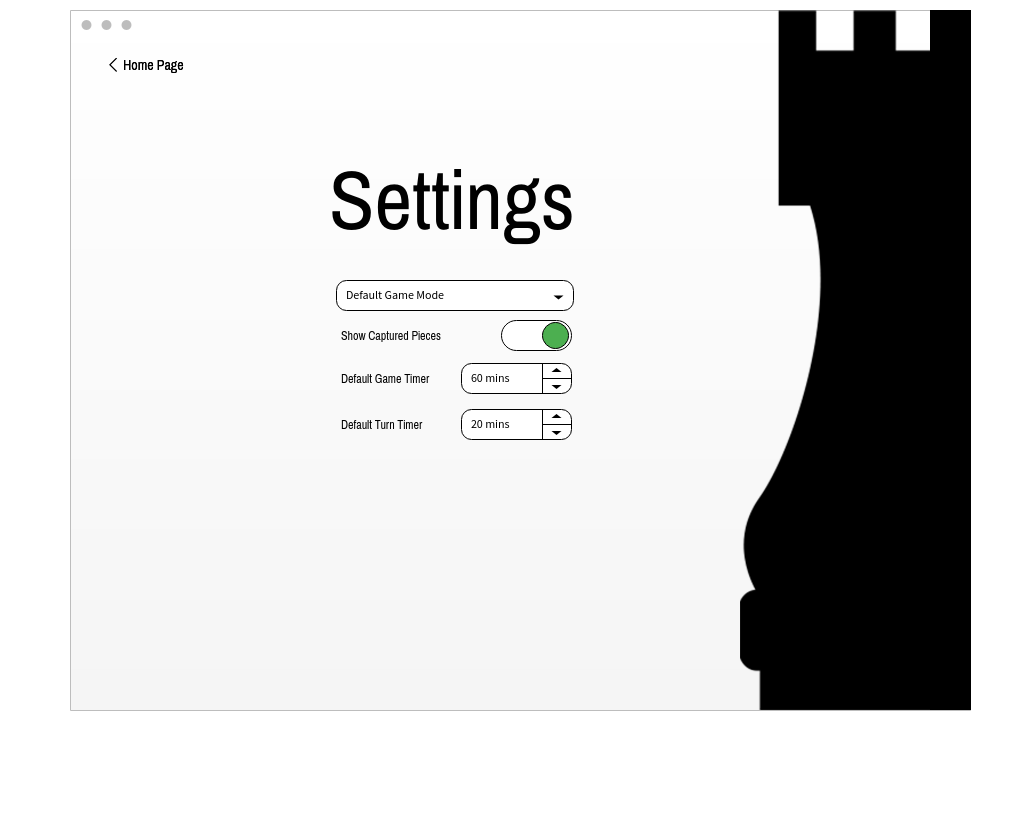


Figure 4. Settings Page

* *settings page* where FAC shall provide the *user* with the ability to change default values of the *game timer*, *turn timer*, enable/disable *captured piece tracking*, and choose a default *game mode*, see figure 4. The settings shall be saved between instances of the FAC software. This will be completed using a .xml, .text, or .json file.
* *game setup page* where FAC shall provide *user(s)* with the ability to provide a *username* and override the default values of the *game timer*, *turn timer*, and *game mode*, see figure 5. The *game page* is also where a player starts a game thus causing the FAC software to initiate the GB, GP’s, *game timer, turn timer,* and *game page* encompasses the GB and all 32 GP’s as well as the *game timer*, *turn timer*,



Figure 5. Game Setup Page

and *captured pieces* count.

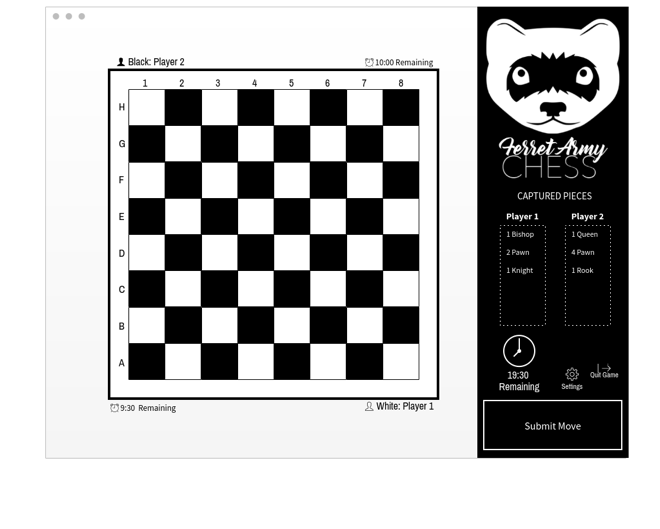


Figure 6. Game Page

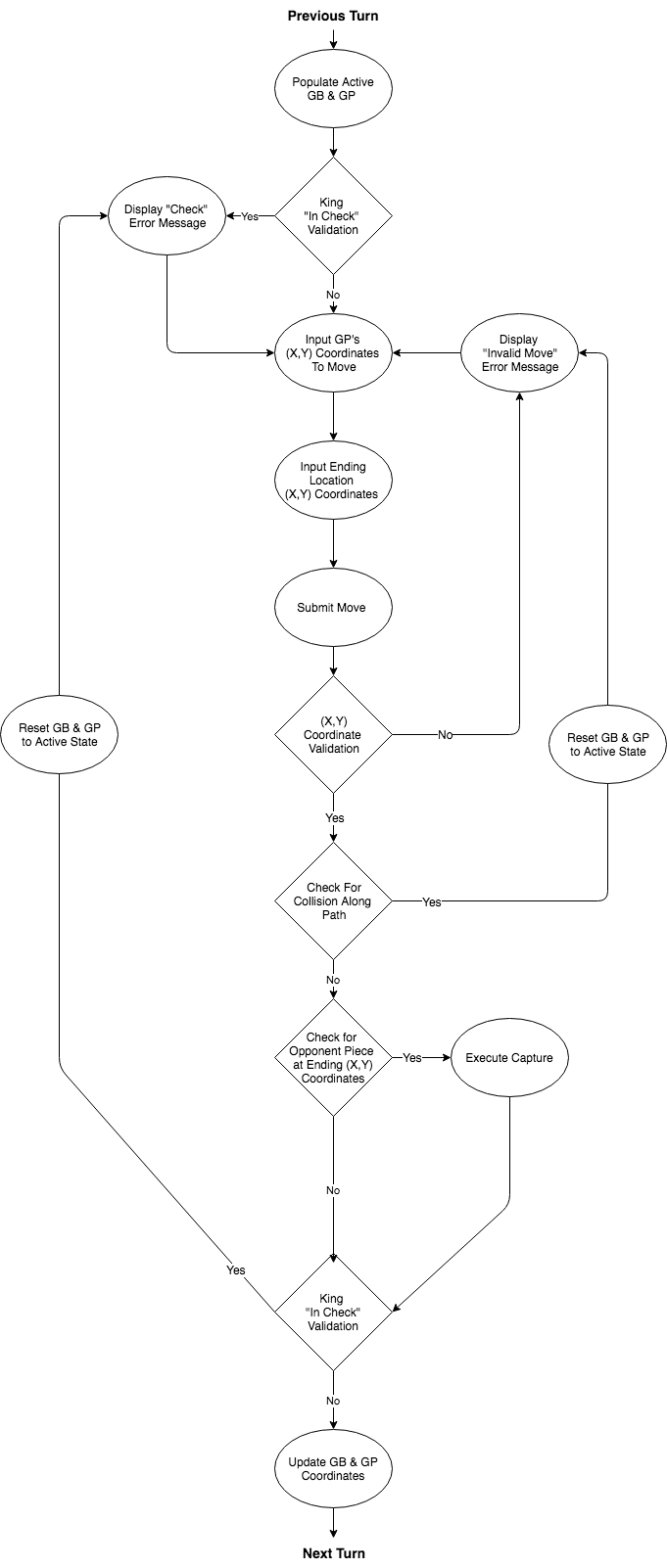
* The *game page* is the primary page *users* see while playing FAC, see figure 6. The *game page* shall accurately show the current *game state*, position of the GP’s, allow *players* to submit a move, and notify *players* if a move is *invalid*, leads to *capture*, or leads to *check* or *checkmate*.

It shall also display a message when the game timer “runs outs” triggering a call to the GE to initiate the *stalemate* *resolution*. The *stalemate resolution* feature will be discussed in depth later in this document.

## Game Engine (GE)

The *game engine* (GE) is the most crucial component of the FAC software. It is responsible for handling a majority of the interactions between the UI, *users*, and where applicable the AIE (via the *AIE* *controller*). Additionally, the GE is the component that handles all chess logic associated with a *player* making a move, validating said move, checking for an *end game* state, and finally sending a message to the GB (via the UI) to reflect the change to the *player*. A standard interaction for the GE with a *player* can be seen in figure 7.

Figure 7. User Moving a Piece



As previously mentioned a primary function the GE serves beyond interfacing with the user through the UI is sending is controlling the AIE through the *AIE controller* which is the name given to an instance of the AIE during a game of FAC. The *AIE Controller* is responsible for querying the AIE for a potential *computer* related move when a *user* is acting as the role of a *player* in the *user vs. computer* game mode, or when a *user* is acting as the role of a *bystander* in the *computer vs. computer* game mode.

## Artificial Intelligence Engine (AIE)

The *artificial intelligence engine* (AIE) is arguably the second most crucial component of the FAC software. It provides all logic necessary to generate a move on the behalf of the *computer*. This shall be completed using 3 random number generators. The first random generator shall choose the piece, the second random number generator shall choose the direction the GP will attempt to move, and the third random generator shall choose the distance from the GP’s current position to the desired location.

Although not intelligent the AIE will seemingly appear “smart” as it will choose a piece, direction, and length of move each time it is called. Once the AIE generates a move the GE will validate the move. If the move is valid then the GE will reflect the move on the GB. If the move is not valid then the GE will query the AIE for a new move (this includes a new piece, direction, and distance). This process will repeat until a valid move is reflected on the GB. The process of generating the numbers to calculate the move will be discussed in-depth in section 3.

# High-Level Design

## Game Board (GB) / User Interface Element

Like a traditional game of chess FAC shall implement an 8-by-8 *game board* (GB). The GB shall be colored with alternating colors with the *white player* starting on the bottom of the GB, and the *black player* starting on the top of the GB. To represent the GB a 2-dimensional array shall be used as it naturally facilitates (X,Y) coordinates, with both X-COR and Y-COR being bounded from 0-7. In an effort to ease the calculation of each GP’s offset the coordinate pattern shown in figure 8. shall be used for the 2-dimensional array.

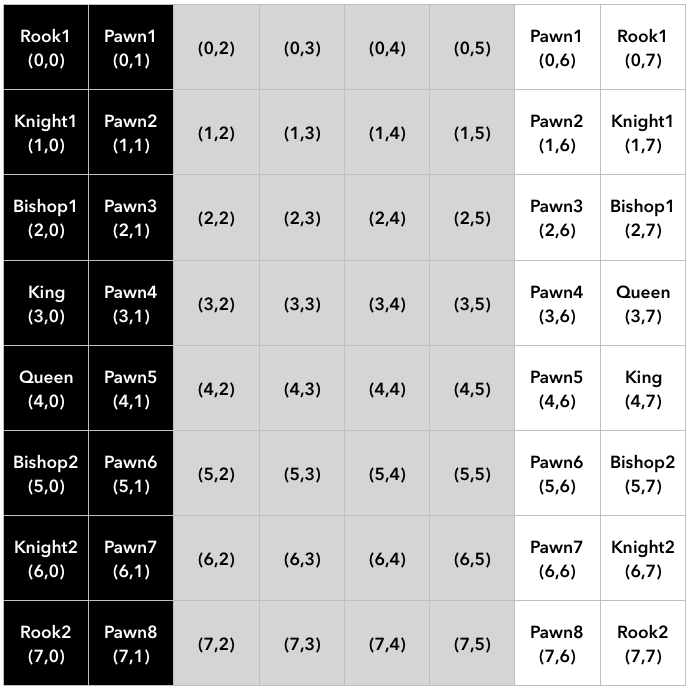


Figure 8. Game Board with Coordinate Listing

The reason for aligning the black and white GP’s vertically instead of horizontally is to hold to the principle of “*keep the common case fast*”. Thus, calculating horizontal moves is a matter of subtraction or addition along a single row. Vertical moves are a matter of subtraction or addition along a single column. Diagonal moves require calculating offsets from the GP’s current position. The offsets for *vertical*, *horizontal*, and *diagonal* moves can be seen in figure 9. It is important to note that the offsets shown in figure 9. only apply if the coordinate labeling in figure 8. is implemented.

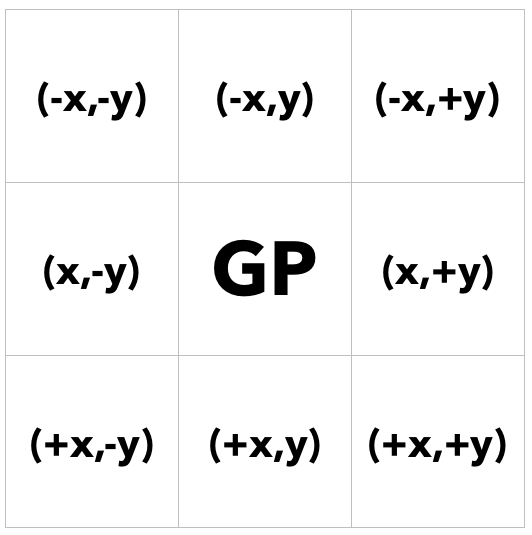
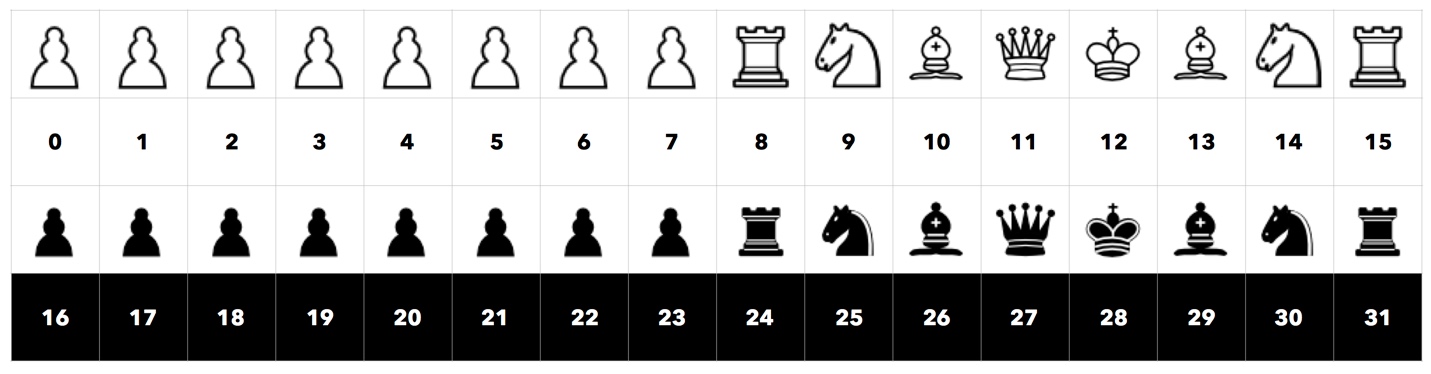


Figure 9. Offset Calculations

## Game Piece (GP) / User Interface Element

In a game of FAC there are 32 individual *game pieces* (GP). With both playable colors (white and black) having 8-pawns, 2-rooks, 2-knights, 2-bishops, 1-queen, and 1-king for a total of 16 GP’s per *player.* GP’s may have one of two states, *active* or *non-active*. Since each space on the GB will have a valid corresponding (X,Y) coordinate to signify a capture an invalid coordinate of (-1,-1) will be assigned. The GE owns the list of *active* and *non-active* (captured) GP’s for both the *white player* and *black player.* This shall be implemented using a 1-dimensional array. The alignment of pieces, by index, in the array including color collation can be seen in figure 10. with *white player’s* pawns stored in indices 0-7 and *white player’s* value pieces stored in indices 8-15, *black player’s* pawns in 16-23 and *black player’s* value pieces stored in indices 24-31.

Figure 10. Game Piece Array



This 1-dimensional array will also be called upon heavily when the AIE randomly selects a GP to move. The logic of the AIE and its use of the *game piece array* (GPA)will be discussed in-depth in the next section.

## Random Generator #1 / Artificial Intelligence Engine (AIE)

As previously mentioned the AIE is made up of 3 random number generators. Random generator #1 will generate a number between 0-31 corresponding to the 1-dimensional array outlined in section 3.2. When it’s the *white player’s* random generator #1 shall be seeded with a number between 0-15 as shown in figure 11. When it’s the *black player’s* turn random generator #1 shall be seeded with a number between 16-31 as shown in figure 12.

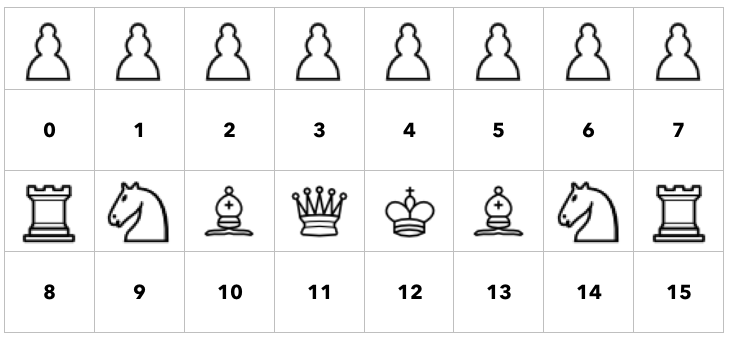
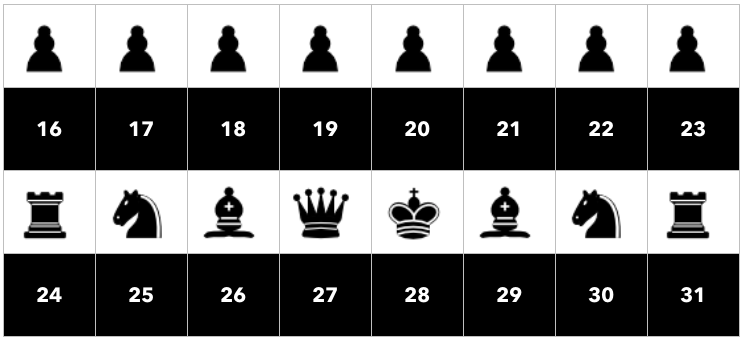


Figure 11. Game Piece Array (White Only)

Figure 12. Game Piece Array (Black Only)



After a random number is generated 0-15 or 16-31 the piece at that location shall have its active coordinates checked to determine if they are (-1,-1) or another value. As described if the active coordinates for the selected GP in the *game piece array* are (-1,-1) then that piece has already been captured during a previous turn. In this case a new random number is generated until an active GP is found.

If it is white’s turn then a number between 0-15 will be generated. If it is black’s turn then a number between 16-31 will be generated. In order to signify a captured piece the (X,Y) coordinate (-1, -1) will be used. Every time a number 0-15 or 16-31 is generated the AIE will check the selected GP’s current coordinate to determine if that piece is captured or active. .

The second random number generator shall generate a number from 0-7 representing the direction the GP will move from its current location. In order to generate an (X,Y) coordinate for the chosen direction a numeric system shall be assigned to each unique GP’s available direction of movement. Since a queen and king have the most available directions of movement: *forward*, *backward,* *left*, *right*, *diagonally-left-upper, diagonally-right-upper, diagonally-left-lower, diagonally-right-lower* the numbers 0-7 will be used to correspond to the available directions starting vertically moving counter-clockwise.

The third random number generator shall generate a number from 0-7 representing the Y-COR in the (X,Y) coordinate pair. The calculated coordinate pair (X,Y) shall then be returned back to the GE to validate the move in accordance with that specific GP’s available moves. If the move is deemed invalid, then the GE will make another call to the AIE to generate a new move, again this only occurs in *computer* vs. *computer* and *computer* vs. *human* game modes.

## View / Model Component 1..n