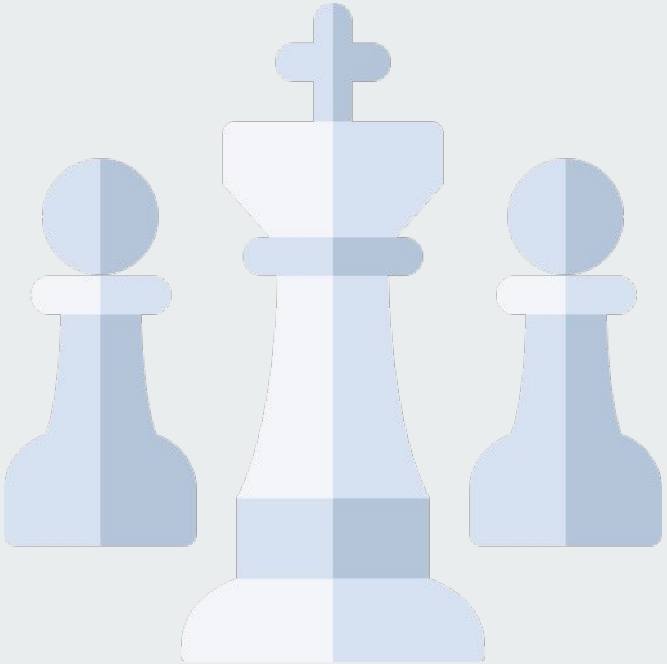
AI Chess Bot

By: Andrew Delgadillo and Brandon Fontenot



Introduction

- Chess is a game that has been played since sometime in the 6th century
- It is a game that is very simple to learn but very complex at the highest levels
- Both of us started playing not too long ago which lead us to being interested in making this chess bot

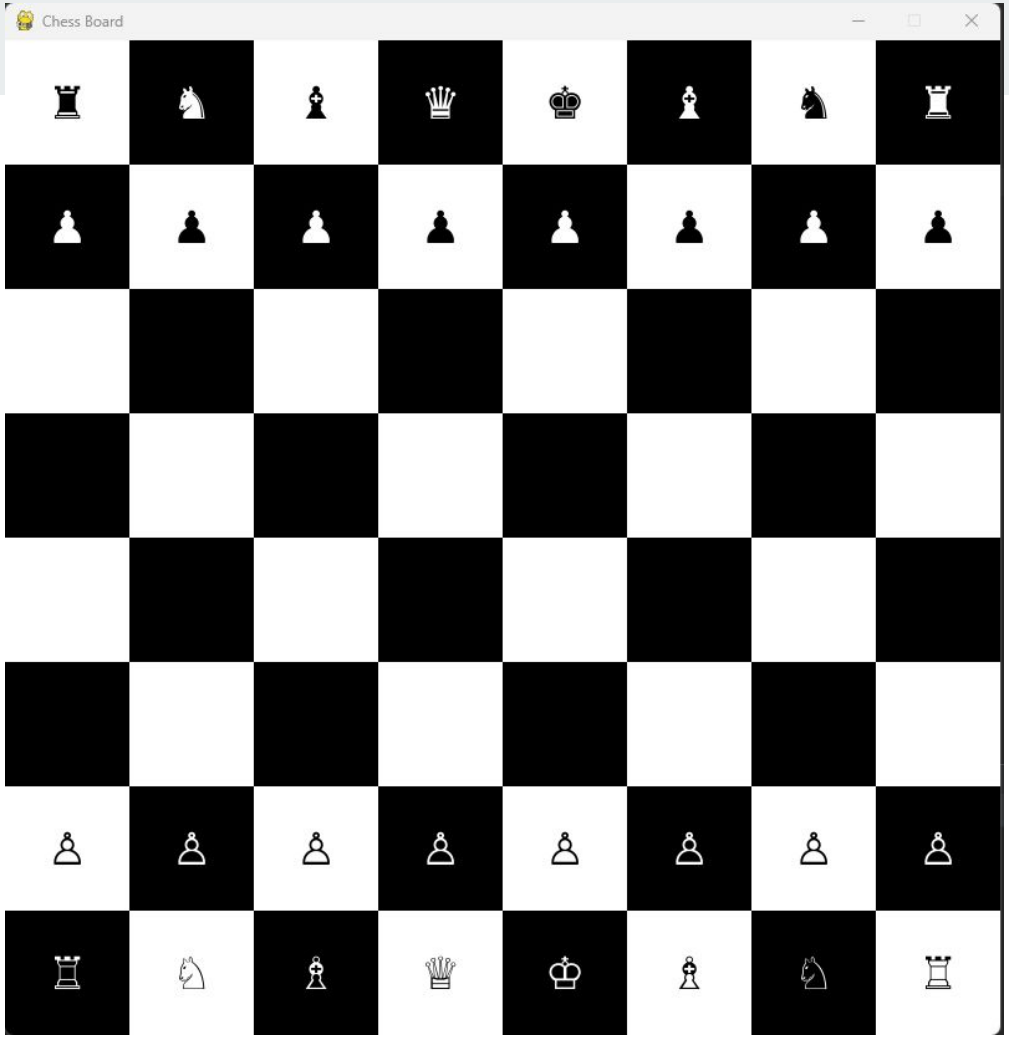
Problem Statement

- Develop an AI that is capable of playing chess at a competent level (able to beat us)
- This requires the bot be able to
  - Understand the rules of chess
  - Evaluate board positions
  - Understand piece development
  - Coordinate attacks

Quick Background

- The GUI and API we chose was from a Chess Coding Challenge in C#
- Challenge? To create a small chess bot in C#
- All Credit goes to **Sebastian Lague**
  - ***Other than the bot we made of course***

Chess Board



Black: MyBot

Time: 00:53.6

Human vs MyBot

MyBot vs MyBot

MyBot vs EvilBot

Save Games

Rules & Help

Documentation

Submission Page

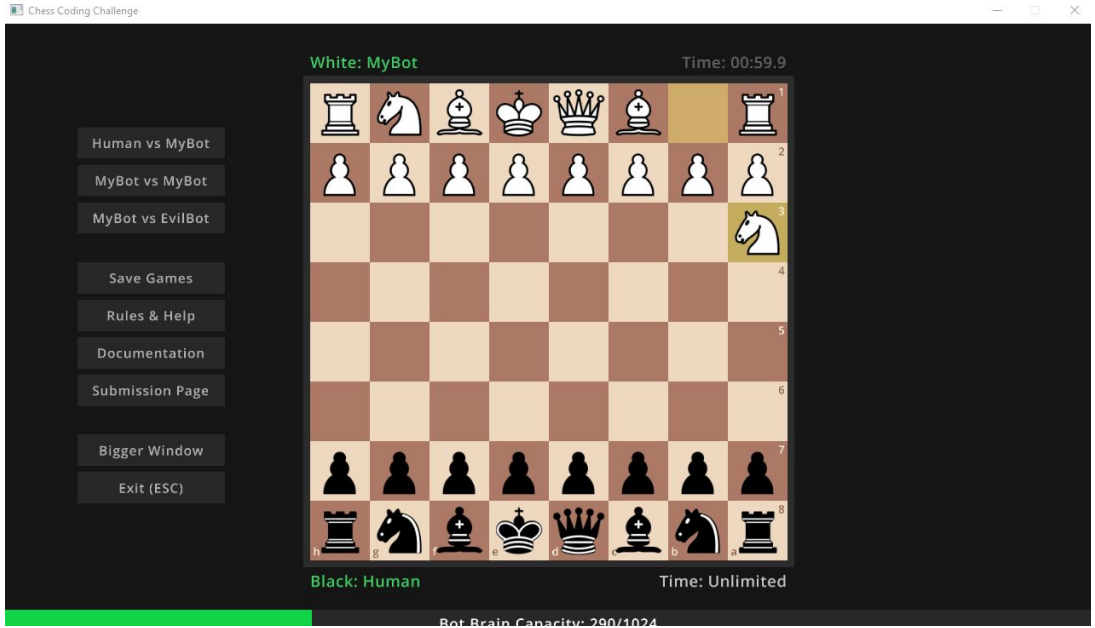
Bigger Window

Exit (ESC)

White: Human

Time: Unlimited

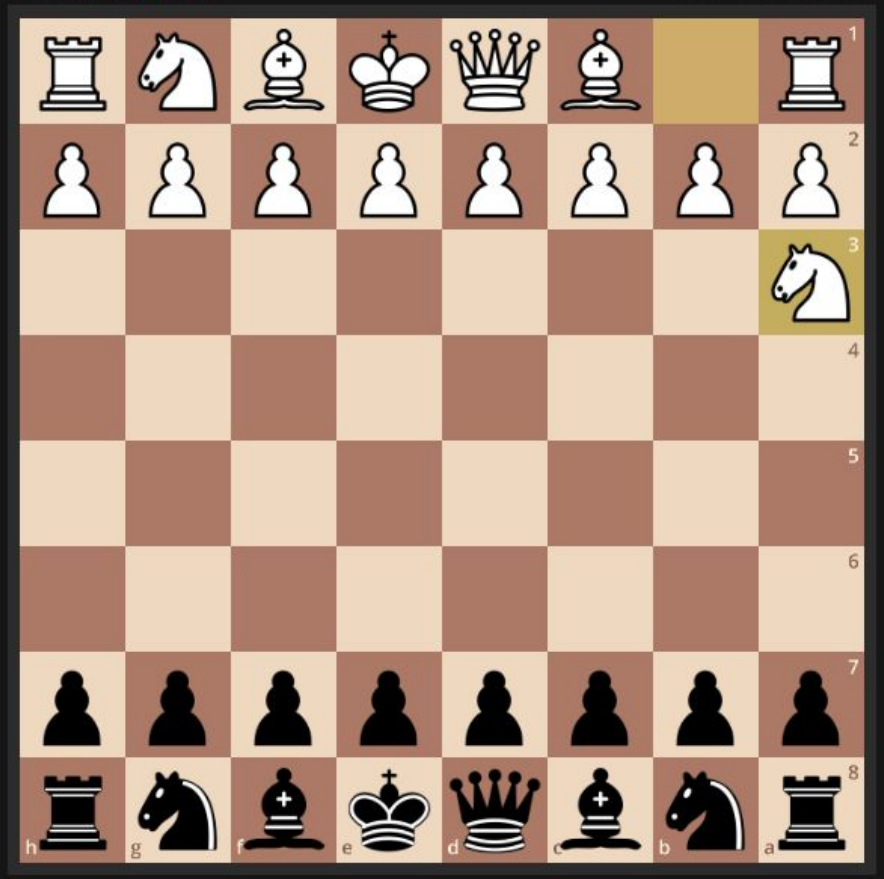
Use-Case Scenarios



Chess Coding Challenge

White: MyBot

Time: 00:59.9



Black: Human

Time: Unlimited

Human vs MyBot

MyBot vs MyBot

MyBot vs EvilBot

Save Games

Rules & Help

Documentation

Submission Page

Bigger Window

Exit (ESC)

Chess Coding Challenge

White: MyBot

Time: 00:59.9

Black: Human

Time: Unlimited

Human vs MyBot

MyBot vs MyBot

MyBot vs EvilBot

Save Games

Rules & Help

Documentation

Submission Page

Bigger Window

Exit (ESC)

Bot Brain Capacity: 290/1024

Chess Coding Challenge

White: MyBot

Time: 00:59.9

Black: Human

Time: Unlimited

Human vs MyBot

MyBot vs MyBot

MyBot vs EvilBot

Save Games

Rules & Help

Documentation

Submission Page

Bigger Window

Exit (ESC)

Bot Brain Capacity: 290/1024

Chess Coding Challenge

White: MyBot

Time: 00:59.8

Black: Human

Time: Unlimited

Human vs MyBot

MyBot vs MyBot

MyBot vs EvilBot

Save Games

Rules & Help

Documentation

Submission Page

Bigger Window

Exit (ESC)

Bot Brain Capacity: 290/1024

Chess Coding Challenge

White: MyBot

Time: 00:59.8

Human vs MyBot

MyBot vs MyBot

MyBot vs EvilBot

Save Games

Rules & Help

Documentation

Submission Page

Bigger Window

Exit (ESC)

Black: Human

Time: Unlimited

Chess Coding Challenge

White: MyBot

Time: 00:59.8

Human vs MyBot

MyBot vs MyBot

MyBot vs EvilBot

Save Games

Rules & Help

Documentation

Submission Page

Bigger Window

Exit (ESC)

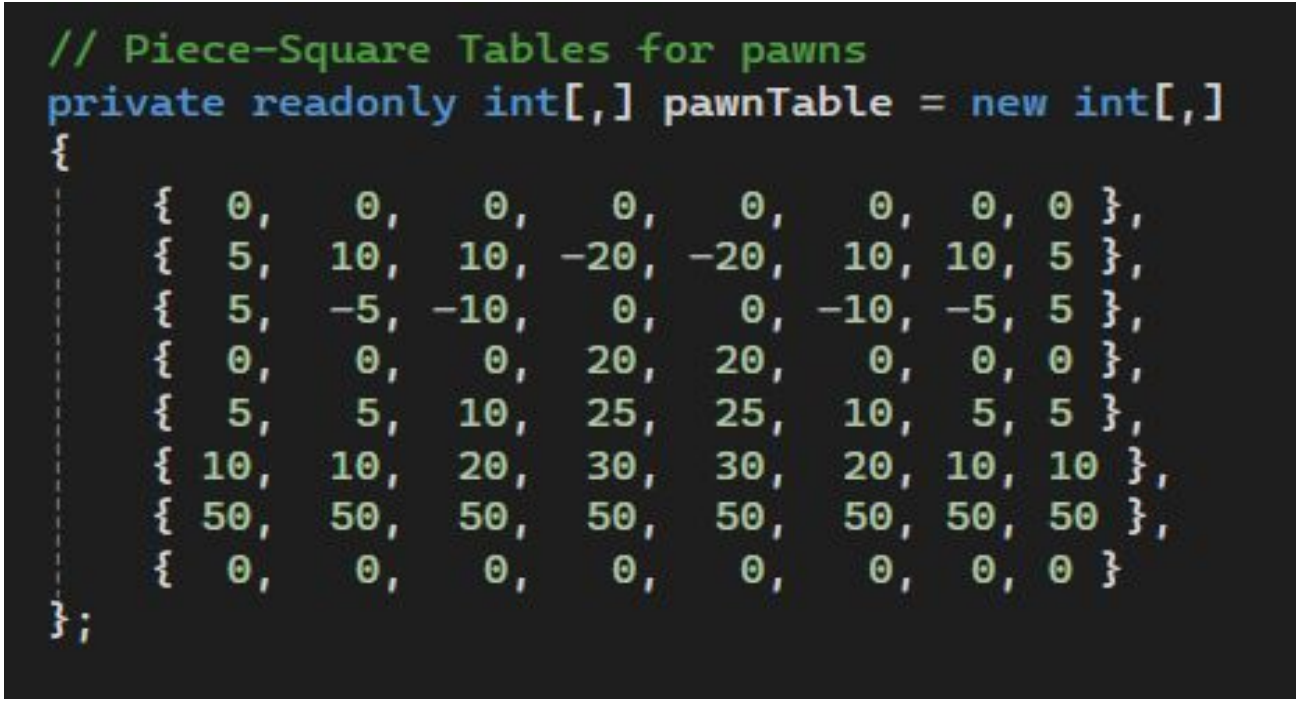
Black: Human

Time: Unlimited

Bot Brain Capacity: 200/1024

Our model

- Piece-Square Tables: For the pawns and knights, assigns values to pieces based on their positions on the board.
- Alpha-Beta Pruning / Minimax: Reducing the number of nodes evaluated in the minimax algorithm
- Evaluation Function: Calculating the score based on the positions and types of pieces on the board, Pawn isolation, Danger positions


// Piece-Square Tables for pawns
private readonly int[,] pawnTable = new int[,]
{
{ 0, 0, 0, 0, 0, 0, 0, 0 },
{ 5, 10, 10, -20, -20, 10, 10, 5 },
{ 5, -5, -10, 0, 0, -10, -5, 5 },
{ 0, 0, 0, 20, 20, 0, 0, 0 },
{ 5, 5, 10, 25, 25, 10, 5, 5 },
{ 10, 10, 20, 30, 30, 20, 10, 10 },
{ 50, 50, 50, 50, 50, 50, 50, 50 },
{ 0, 0, 0, 0, 0, 0, 0, 0 }
};

```
int[,] knightTable = new int[,]
{
{-50, -40, -30, -30, -30, -30, -40, -50},
{-40, -20,   0,   0,   0,   0, -20, -40},
{-30,   0,  10,  15,  15,  10,   0, -30},
{-30,   5,  15,  20,  20,  15,   5, -30},
{-30,   0,  15,  20,  20,  15,   0, -30},
{-30,   5,  10,  15,  15,  10,   5, -30},
{-40, -20,   0,   5,   5,   0, -20, -40},
{-50, -40, -30, -30, -30, -30, -40, -50}
};
```

Demo

Lesson Learned

- Most of the time is spent tweaking the heuristics for evaluation functions
- It’s very important to balance risk taking
- The biggest limiting factor is computational resources

Questions?