AI-m of the Game
Final Showcase

Instructors: Brandon Wang and Brandon Wei
Members: Philip, Rushil, Subham, Victor, Vikram, Varun, Vishaal

Overview of AI-m of the Game

- AI-m of the Game is designed to increase our knowledge about a technology that is increasingly becoming more pervasive, AI. The purpose is to equip us with the basic knowledge to create our own optimal AI's in games and have some fun with it! It's not all Terminator and Skynet.
- We picked the checkers game because it was the perfect mix of challenging game, yet not impossible to do in the timeframe of the project.
- To build up to the checkers game, we learned about the different types of games and how AI is related to that. We observed different strategies to play each game optimally so that we could emulate that in the final project.

Problem to Solve

- Our Goal
  - Using AI and algorithm functions to solve and play games
- The project
  - Checkers is a universally played game with players on every single continent
- Choosing the game
  - Our team looked through a couple of popular board games and decided on Checkers
    - Easy to understand and play
  - Easy to make online and with a team in this COVID environment
- Why is it important
  - Advancing AI will help propel humans into a rapidly digitizing world
  - AI helps to improve the overall gaming experience; making it more immersive and interesting for players
  - AI is trained to always be better than the player - both parties are improving

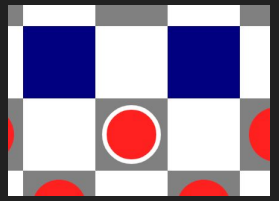
Team Efforts

- Team members:
  - Philip
  - Rushil
  - Subham
  - Varun
  - Victor
  - Vikram
  - Vishaal
- Advisors:
  - Brandon Wang
  - Brandon Wei
- Collaboration Tools
  - GitHub
  - Shared Codesandbox
  - Regular Zoom meetings

Project Progress

```
function evaluateBoard(board, color) {
  var score = 0;

  var i, j, piece, sign, val;
  for (i = 0; i < boardPieces; i++) {
    for (j = 0; j < boardPieces; j++) {
      piece = board[i][j];
      if (!piece) continue;
      // sign = piece % 2 === color % 2 ? 1 : -1;
      // val = piece > 2 ? 5 : 3;
      // score += val * sign;
      sign = normalize(piece) === color ? 1 : -1;
      val = piece > 2 ? 5 : 3;
      score += val * sign;
    }
  }
  return score;
}
```

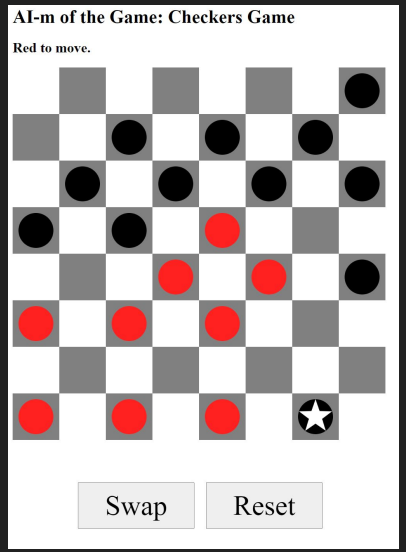


Features

- UI Layout - Javascript + HTML
- Game Mechanics - Jumps, showing valid moves, captures, promotions
- Problem: Multi Jumps. Tried recursion projections, etc. Solved with turn system
- Computer AI - controlling a player; adding functions
- Custom game analysis tools - Evaluation Function
- Minimax recursion using depth-first search

Optimizations and Challenges

- Optimization: Alpha Beta Pruning
- Challenge: Projecting Multi Jumps
- Challenge: Project Endstates
- Challenge: Version coordination
- Final: Optimized to see six moves ahead



MVP & The Final Result

- MVP - Minimum Viable Product
  - A checkers game that has the basic gameplay functionalities as well as an AI that uses minimax and alpha-beta pruning to solve the game
  - Our checkers AI is very difficult to beat w/ a depth of 6
    - Future improvements → optimizations to make code more efficient/faster, adjust weights to make AI play smarter, multi-jump prevention, trading pieces
- Improvements upon MVP made during workshops
  - Added eval function
  - Added depth of 6

https://csb-xuydb.netlify.app/

Checkers Project Demo!