IS597 Final Project: The C&C Game

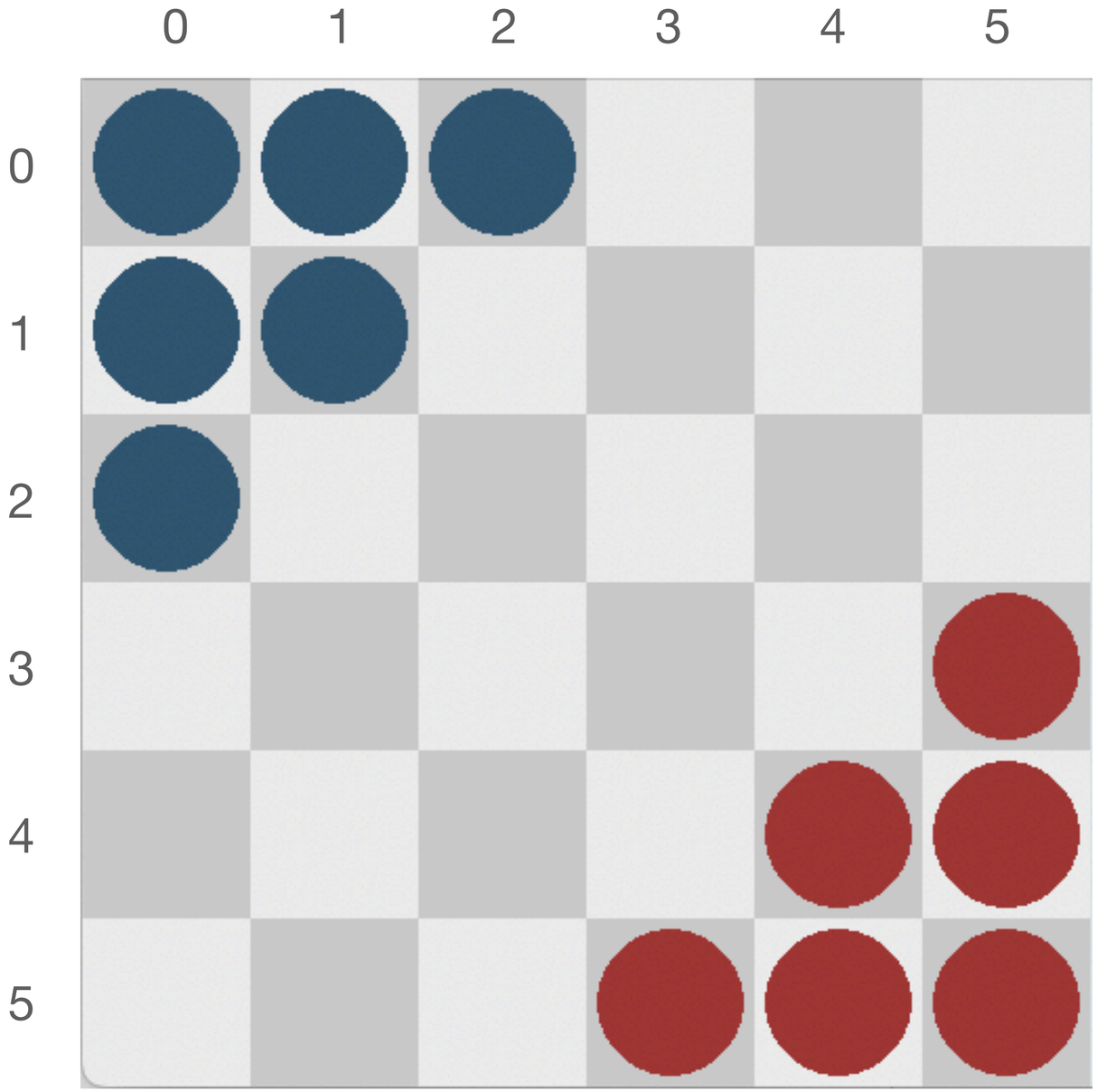
Yu-Wei Lai

April 29, 2022

Introduction

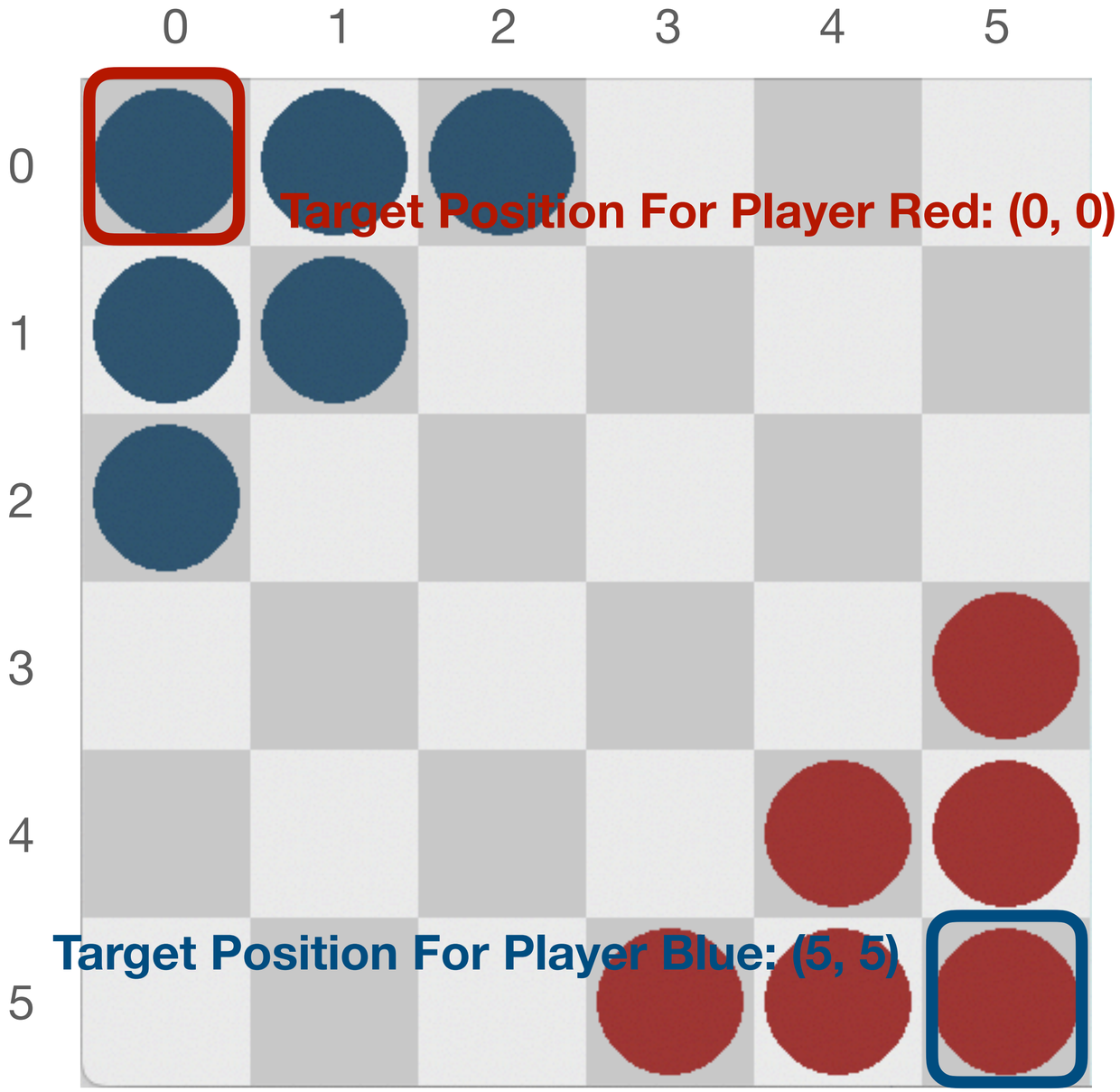
Conquering and Capturing

- **Ideas**
  - Chinese Checkers
  - Halma Game
  - English Checkers
- **Adjustments**
  - Jumping Moves: Continuously Jumping and stop at any valid position
  - Capturing Rule
  - Conquering Rule
- **Create a AI Player**



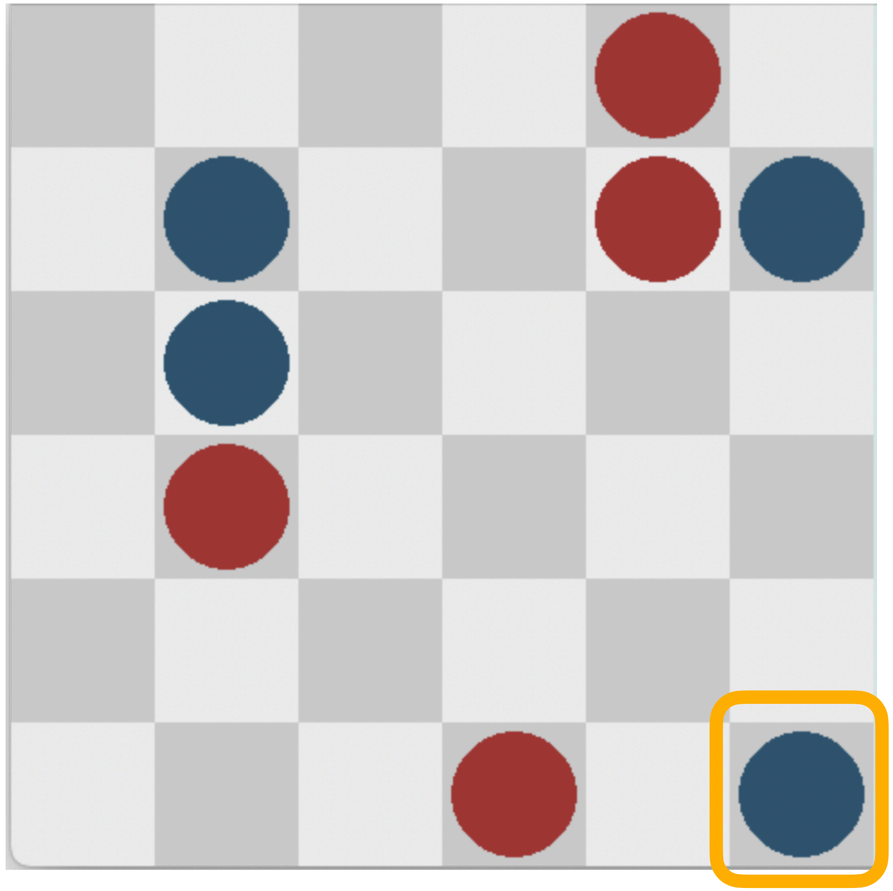
How to win a game?

- Occupying the target position
  - **For Blue Player: (5, 5)**
  - **For Red Player: (0, 0)**
- Make the opponent has only one piece left on the board

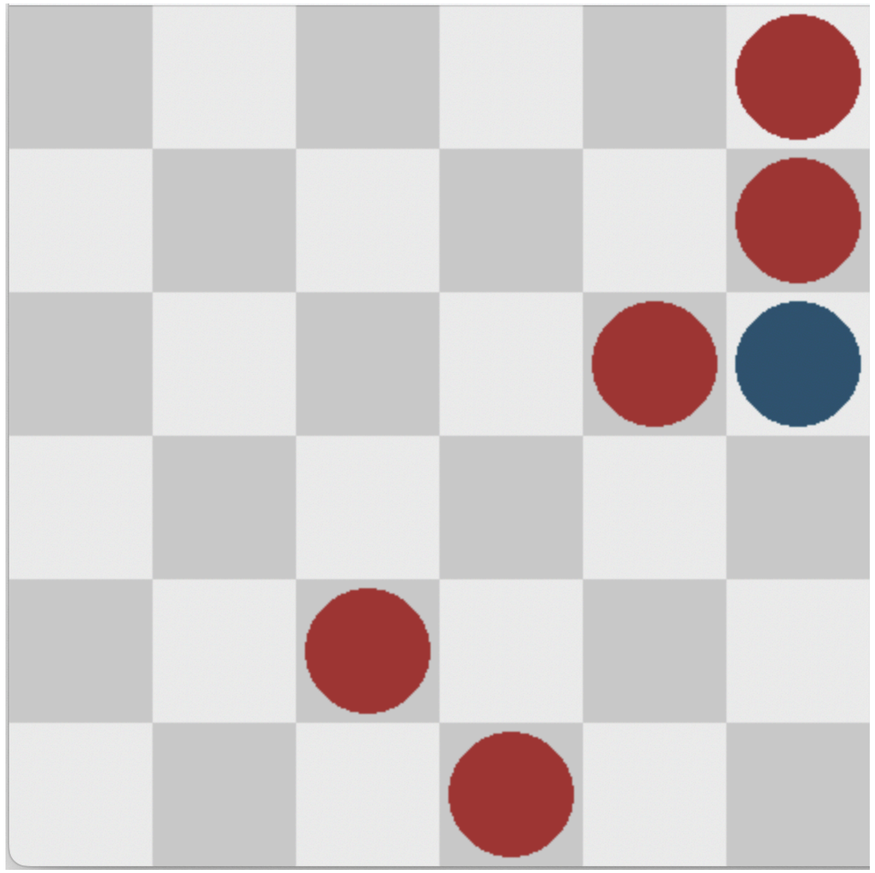


How to win a game?

Winning Condition 1:
Occupy the target position (Blue Wins)



Winning Condition 2:
Opponent has only one piece left (Red Wins)



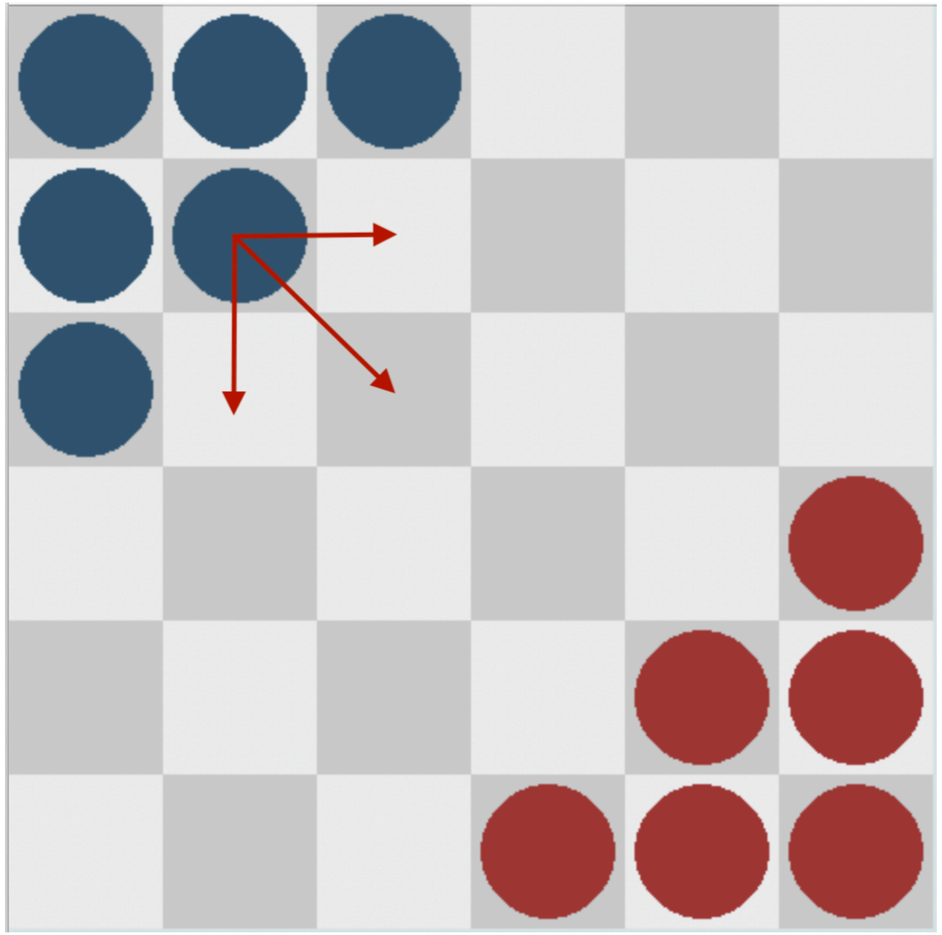
How to make a move?

Normal Moves and Jumping Moves

- **Normal Moves**

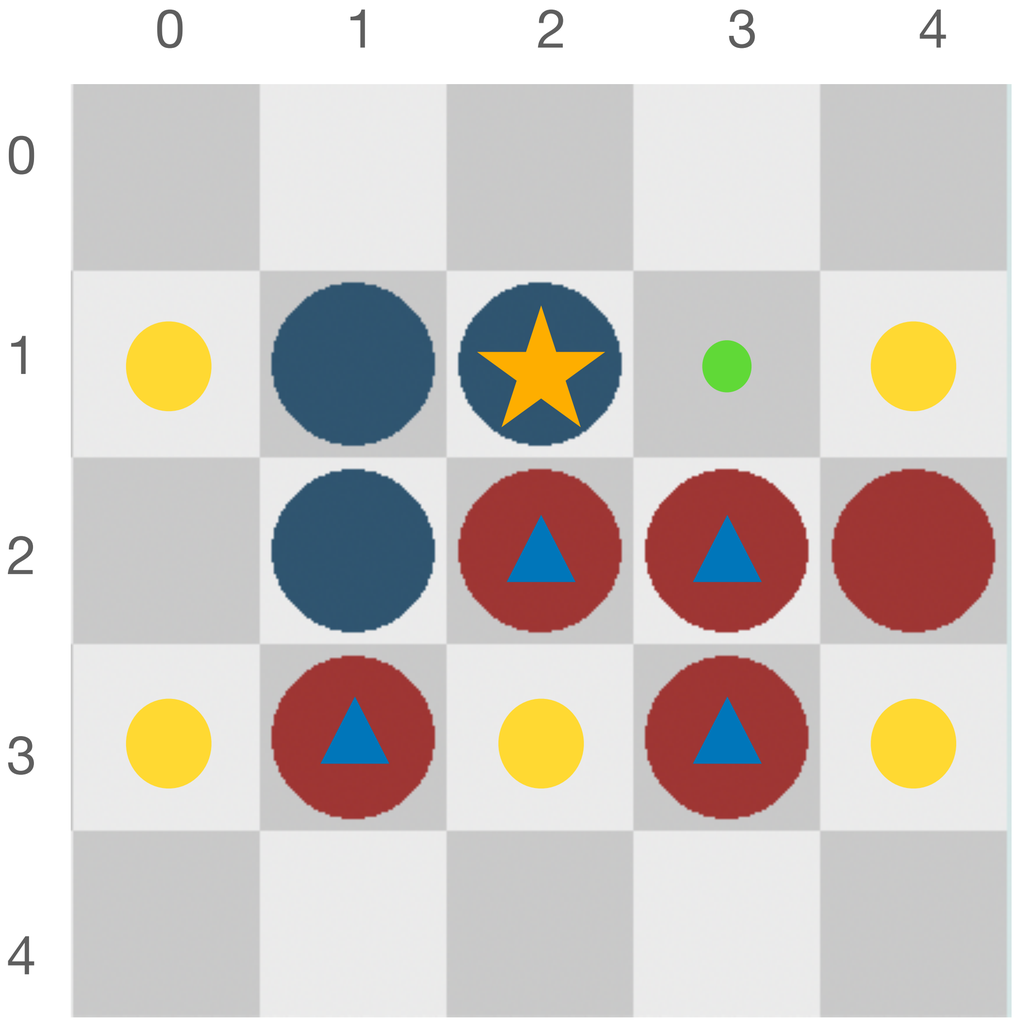
Three possible directions - the end positions should always getting closer to the target

- **Jumping Moves**
  - Like English Checkers: but can continue
  - Like Chinese Checkers: but in 8 directions
  - **Capturing:** When jumping over opponent's pieces, the first jumped over opponent's piece will be removed.



How to make a move?

Example of a jumping move



Possible Jumping Moves For Blue(1, 2):

- (1, 2) -> (3, 2) — Capture (2, 2)
- (1, 2) -> (3, 2) -> (3, 0) — Capture (2, 2)
- (1, 2) -> (3, 2) -> (3, 4) — Capture (2, 2)
- (1, 2) -> (3, 2) -> (3, 4) -> (1, 4) — Capture (2, 2)
- (1, 2) -> (3, 0)
- (1, 2) -> (3, 0) -> (3, 2) — Capture (3, 1)
- (1, 2) -> (3, 0) -> (3, 2) -> (3, 4) — Capture (3, 1)
- (1, 2) -> (3, 0) -> (3, 2) -> (3, 4) -> (1, 4) — Capture (3, 1)
- (1, 2) -> (1, 0)
- (1, 2) -> (1, 0) -> (3, 2)
- (1, 2) -> (1, 0) -> (3, 2) -> (3, 4) — Capture (3, 3)
- (1, 2) -> (1, 0) -> (3, 2) -> (3, 4) -> (1, 4) — Capture (3, 3)
- (1, 2) -> (3, 4) — Capture (3, 2)
- (1, 2) -> (3, 4) -> (1, 4) — Capture (3, 2)

Important Implementations in the Program

Functions and Classes for Game Operating

- Class: Move - For recording moves
- Class: GameBoard - For recording and dealing game status
  - is_winning, is_loosing => test if a game is end
- Function: **generate_moves( )** - **O(n^2)** generate all valid move for a board
- Function: **is_valid( )** - **O(1)** check if a move is valid
- Function: **make_a_move( )** - **O(n)**

  The function will generate a new game status => next round (Using **deepcopy( )** which is O(n))

Important Implementations in the Program

Moves Generating - in $O(n^2)$

The function **generate_moves( )** will generate all possible moves for a game status

- Algorithms for **Normal Moves**
  - Complexity: $O(1)$
  - The function **generate_moves( )** where dealing with normal moves by trying three neighbor directions.
- Algorithms for **Jumping Moves**
  - Complexity: $O(n^2)$
  - The function **generate_moves( )** where dealing with the jumping move use a **brute force searching process** to get all valid end positions.

Important Implementations in the Program

Functions and Classes for Players Playing Games

- Class: **HumanPlayer**
- Class: **RandomPlayer**
- Class: **SmartPlayer**
- Class Method: **choose_a_move( )**
  For each of the three players classes, there is a function choose_a_move( ) for **choosing moves in the valid set of the moves** in different methods
  - For HumanPlayer - **O(n)** list out all options to human players, and get user's input
  - For RandomPlayer - **O(1)** randomly pick a move by indexes
  - For SmartPlayer - **$O(n^{depth})$** discuss it in the next slide…

Important Implementations in the Program

Smart Player to choose a move - in $O(n^{depth})$

- **A mini-max-like Algorithm**
  Consider further moves by self and opponents and evaluate the moves
- Includes a depth search for making further moves
- Complexity: $O(n^{depth})$ (default depth is 3)
- Evaluations: Based on three strategies
  1. Prefer to reduce opponents' pieces
  2. Prefer to get closer to the target position
  3. Prevent itself from being captured by the opponent

Version 1: Smart Player - Performance

Plays better than random

- Total Testing Games: 50 games
- Average time per round: 22 secs
  (Compare to random player: 0.1 secs per round)
- Performance - The Smart Player **wins 31 games**

Is there first player advantage in the game?

Compete Random v.s. Random for 1000 rounds

- Total Games: 1000
- Total Time: 100.71785899999999
- Player 1 Total Wins: 510
- Player 2 Total Wins: 490
- Draws: 0

The GUI

Play the game!

- Built by **PyGame**
- Currently, we still need to use command line for interacting…
- Process:
  1. **Choose Players' Types**
  2. Game starts!
  3. If playing as a **Human Player**:
     Input a **move option**
  4. When Game Ends:
     Choose to start a new game or end the app

```
----------------------------
Please Choose Player Types:
1: Human Player
2: Smart Player
3: Random Player
----------------------------
Please Choose the Player Type for Player 1:
```

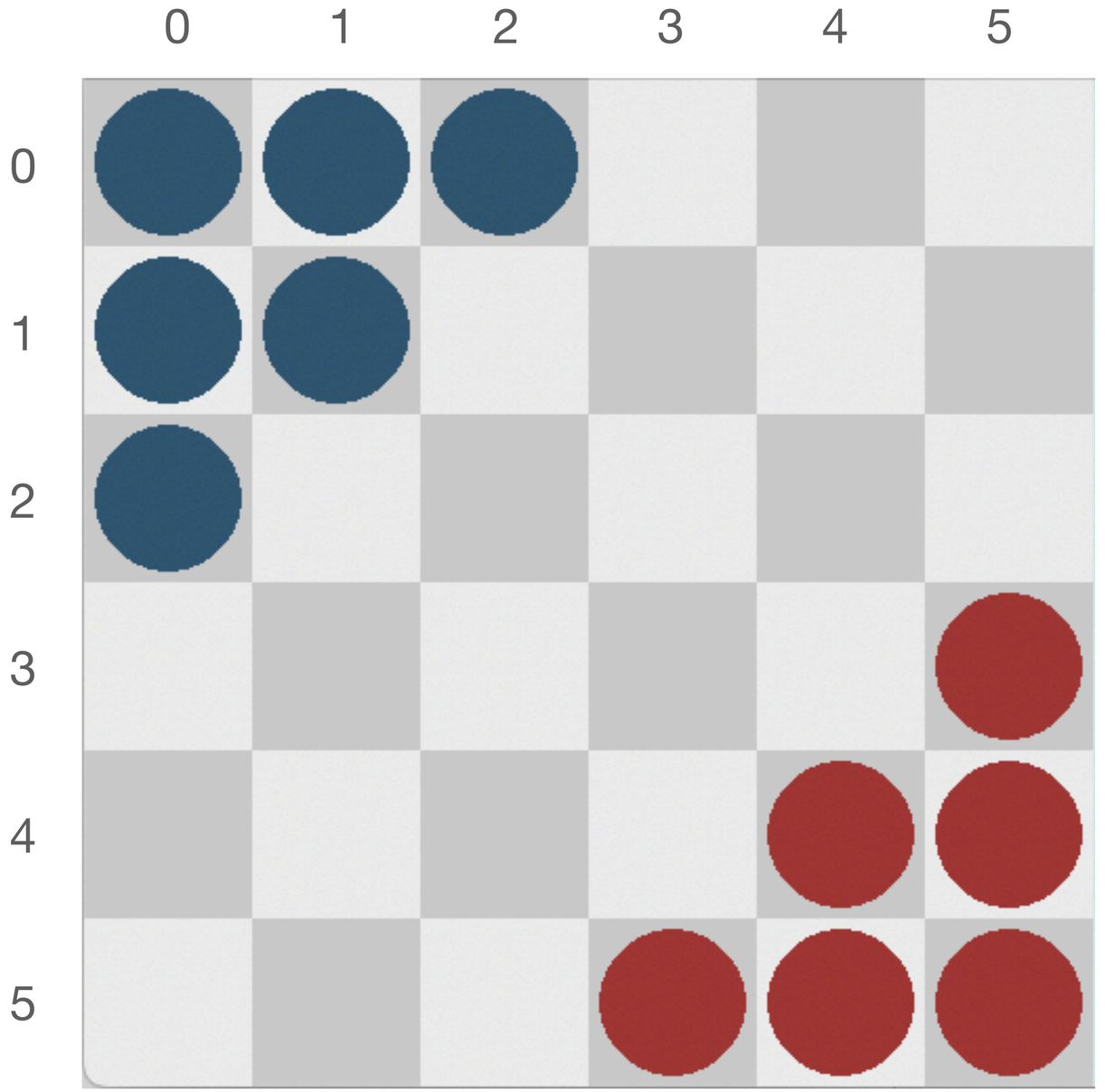
```
Human Player's Options:
Option: 0 - (0, 1)->(1, 2); Capture: None
Option: 1 - (0, 1)->(0, 3); Capture: None
Option: 2 - (0, 1)->(2, 1); Capture: None
Option: 3 - (0, 0)->(2, 2); Capture: None
Option: 4 - (1, 1)->(2, 1); Capture: None
Option: 5 - (1, 1)->(1, 2); Capture: None
Option: 6 - (1, 1)->(2, 2); Capture: None
Option: 7 - (2, 0)->(3, 0); Capture: None
Option: 8 - (2, 0)->(2, 1); Capture: None
Option: 9 - (2, 0)->(3, 1); Capture: None
Option: 10 - (0, 2)->(1, 2); Capture: None
Option: 11 - (0, 2)->(0, 3); Capture: None
Option: 12 - (0, 2)->(1, 3); Capture: None
Option: 13 - (1, 0)->(2, 1); Capture: None
Option: 14 - (1, 0)->(1, 2); Capture: None
Option: 15 - (1, 0)->(3, 0); Capture: None

Please Input A Move Option:
```

```
----------------------------
Restart the game?
----------------------------

1: Restart a game; 2: End the session.
Your Option:
```

App Demo



Improvement

- **The Smart Player**
  - Currently: just perform slightly better than random player
  - Better strategies: less depth and run faster
  - A better evaluation function
  - **Further strategies**
    1. Prevent the opponent from getting closer to the target point
    2. Prefer a stronger structure with connection
- **GUI**
  - Let users play without command line

Thanks!