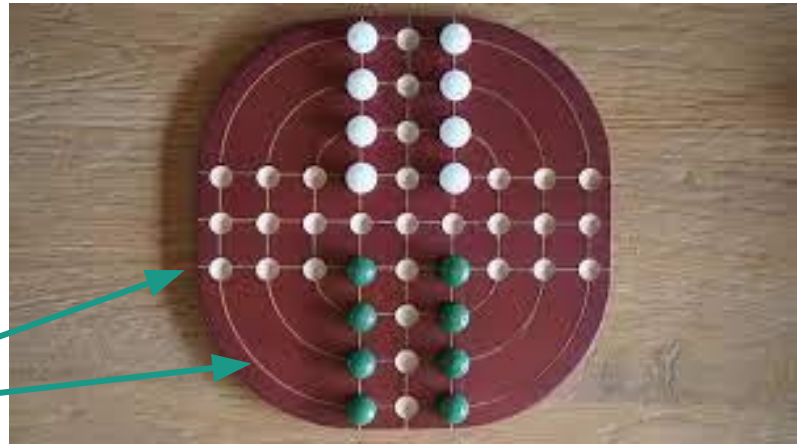
Artificial Intelligence: Wana

Wana - Game Description

- Wana is a 2 player board game.
- Each player has 8 pieces.
- The game ends when one player manages to block all moves of a single opponent piece
- Pieces can move in any direction, as long as there isn't another piece blocking its path
- Pieces can move along the lines leading to the outside of the board, and appear on the opposite side
- Pieces can move along the circular lines



Problem Specification

State Representation: matrix, where each element $\in$ {OUT, EMPTY, PLAYER1, PLAYER2}

Initial State: Matrix is initialized according to the game rules

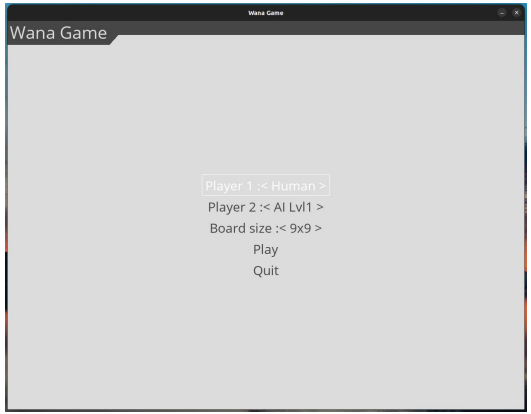
Objective Test: Check if there is a piece with no possible moves

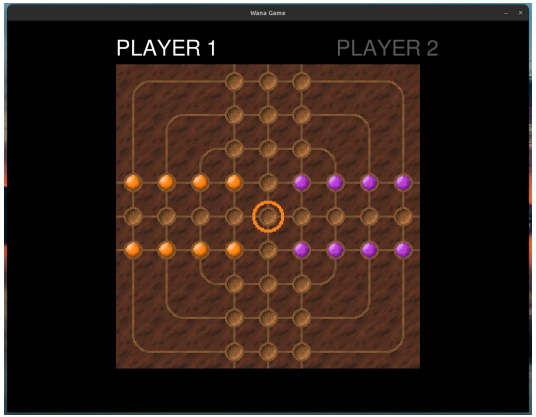
Operator: Move(X, Y)

- Preconditions: position Y, must be empty
- Effect: move the piece from X to Y
- Cost: 1 (turn)

Project Development

Programming Language: Python, using pygame and pygame_menu
Development Environment: VS Code
Data Structures: Lists of integers are used to represent the board
Progress: A lot of work has been completed, but there is always room for improvement





Implemented Algorithms

- Random
- Minimax
  - 5 different heuristics used
  - Depth between 2 and 3
- Monte Carlo Tree Search
  - With 10, 100, 500 and 1000 iterations

Heuristics

- Heuristic 1 = Number of moves that the player has for all pieces
- Heuristic 2 = - Number of moves that the opponent has
- Heuristic 3 = Heuristic 1 + Heuristic 2
- Heuristic 4 = 25 * (X2 - X1) + 5 * (Y2 - Y1)
    - X1 - Number of pieces of player with 0 moves
    - Y1 - Number of pieces of player with 1 move
    - X2 - Number of pieces of opponent with 0 moves
    - Y2 - Number of pieces of opponent with 1 move
- Heuristic 5 = If found losing/winning move then -inf/inf
  Else Y2 - Y1

Experimental Results

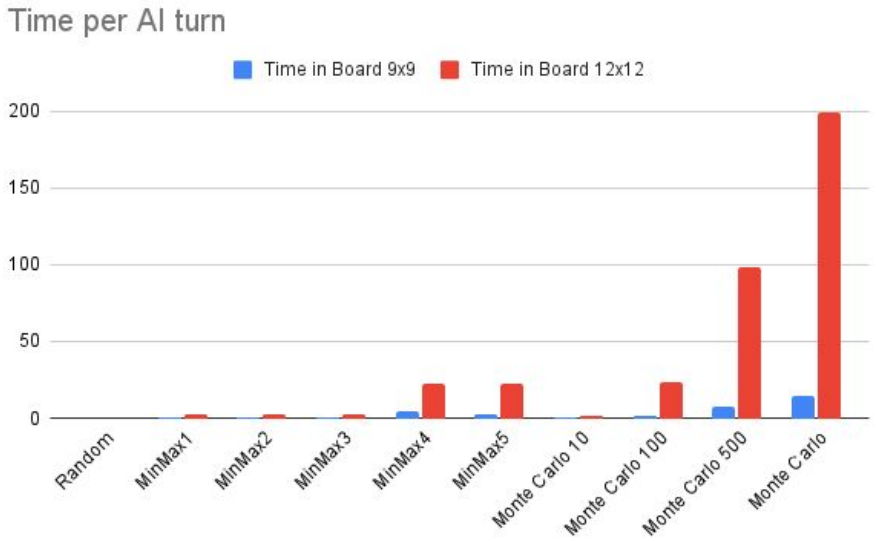
- **AI Random** - Obviously the worst one, and loses with every single other AI
- **AI MinMax 1/2/3**
  - Use of heuristics 1 to 3, and depth 2. They play the game a lot better than the random algorithm, but they are still limited in what they can do.
- **AI MinMax 4/5**
  - Use of heuristics 4 and 5 with depth 3. Since they use slightly better heuristics and have an extra level of depth, they play better than the previous AI's. However, they take much longer to think, and sometimes they even end up in an infinite cycle with lower level MinMax AI's.
- **AI Monte Carlo**
  - Consistent increase of "intelligence" as the number of iterations increases. Furthermore, it won't end up in cycle like MinMax, since it's not deterministic.
  - However, even with 1000 iterations, it usually doesn't win against lower level MinMax.

Experimental Results

As we can see in the graphic, most of the AI's calculations were relatively quick in the 9x9 board.

However, in the 12x12 board, the MinMax 4 and 5 suffered a significant increase in time due to its depth.

The Monte Carlo Algorithm had an even more drastic increase. This is probably due to the usage of a large amount of memory to store the Monte Carlo Tree.



Conclusions

In conclusion, this project allowed us to explore different kinds of algorithms, and use them to create an AI for our game, Wana. In our Project, Minmax was the most successful algorithm, despite its limitations with larger game trees.

The Monte Carlo Algorithm was indeed promising, but unfortunately it wasn't able to perform as well as we had hoped. There are ways to improve it, but it would require a more advanced knowledge and experimentation.

Overall, we believe this project helped us develop our basic understanding of the used algorithms. It also provided us with valuable insights into the challenges of implementing AI in a game.

References