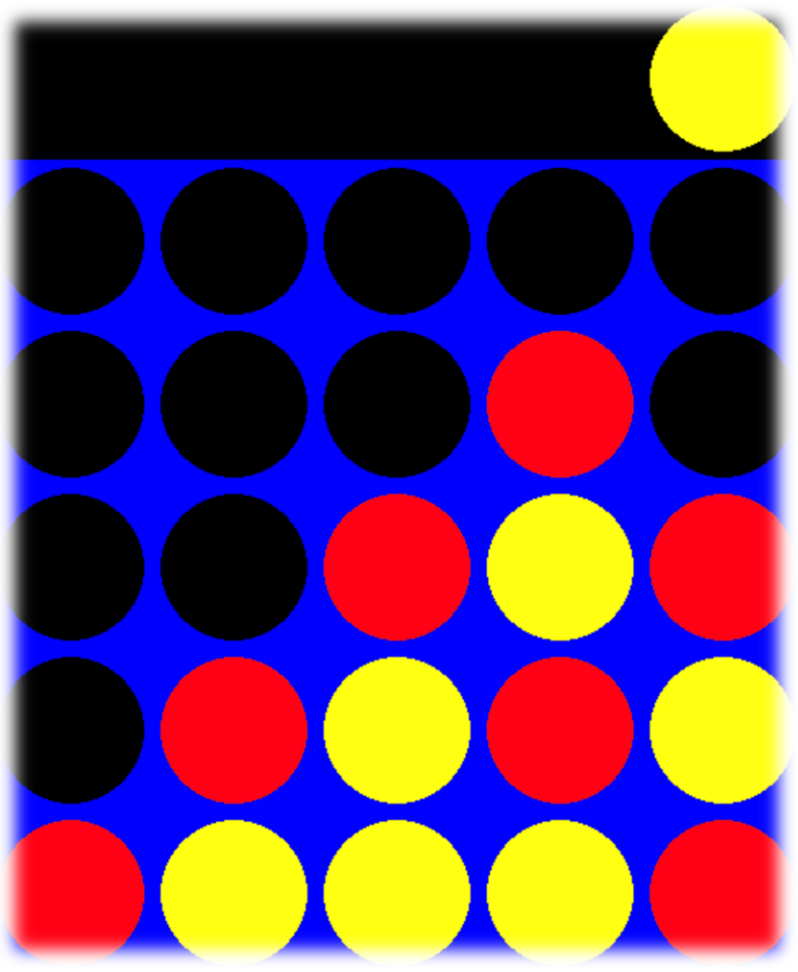
Connect–N AI Game



Abstract

Building upon the classic two-player strategy game Connect 4, one arrives on a generalized '*Connect-N'*. Instead of being limited to a classic 7x6 board and requiring four tokens in a row to win, Connect-N allows for customizable board sizes and the number of connections needed to win.

Our implementation of Connect-N gives the user full control over customization of the board. One can choose the number of columns, the number of rows, and the number of tokens required for winning on the board. The game is played between two, in two modes: **Player vs Player (PvP)** and **Player vs Computer (PvC)**, in which they take turns and compete to form the sequence of 'N'. The AI opponent, in the Player vs Computer mode, is powered by the **Minimax** algorithm with **Alpha-Beta Pruning** for efficient decision-making. The code has **difficulty levels** incorporated too, containing easy, medium, and hard difficulty modes.

Idea of the Code

The code first starts with the prompts for entering the number of columns, rows and the winning condition; followed up by the setting up of the Connect-N board. Then comes the function ‘winning_move’, which checks if the winning condition is met, to crown the winner; followed by the ‘draw_board’ function which specifies specific colors for different situations, such as: green, for highlighting the winning tokens.
Then follow the functions based on computer’s working in the PvC mode, i.e. those based on artificial intelligence algorithms in the code; the **‘minimax’** function, the **‘evaluate_board’** function, and the **‘evaluate_window’** (based on **heuristics**) function. Then there’s a ‘is_board_full’ function which as the name suggests, checks if the board is full. Then there are prompts for selecting the game modes and the difficulties; followed by the setting up of the working of the Connect-N game for these different modes and difficulties.
The game then finally ends, and so does the code.

The AI Algorithmic Functions

The ‘minimax’ function finds the next available row in a column, simulates dropping a piece, and recursively calls minimax with reduced depth, switching to the minimizing player to evaluate outcomes. So, overall it simulates all possible moves to a certain depth; alternating at each step between the maximizing player (chooses moves to maximise the score for AI), and the minimizing player (chooses moves to minimise the score for the opponent). Alpha-Beta Pruning is implemented here too. It updates the best score and column if a better score is found. It implements this pruning to limit the search space: alpha (maximizer's best) and beta (minimizer's best). It prunes unnecessary branches of the search tree to save computational time. After the recursion reaches the base case (depth == 0 or board full), it evaluates the board using ‘evaluate_board’ function.

In the ‘evaluate_board’ function, iteration happens through row; checking different subsections and identifying advantageous horizontal, vertical and diagonal patterns. It also adds a bonus score for pieces in the center column, as the pieces of the center column are more likely to contribute to horizontal, vertical, and diagonal connections. This center column is prioritized to maximize future connection opportunities. Positive contributions indicate an advantage for player 1, while

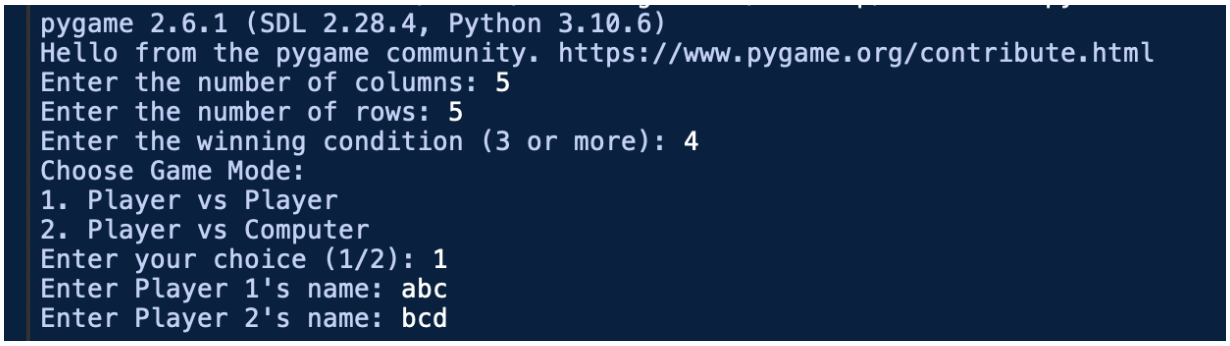
negative contributions indicate an advantage for player 2. Each such subsection/‘window’ is then checked for patterns using the helper function ‘evaluate_window’.
The ‘evaluate_window’ function is the core of the ‘evaluate_board’ scoring system.

The ‘evaluate_window’ function assigns different scores to the window, based on favourability. It gives positive scores for good setups for the player (e.g., near-complete connections) along with giving negative scores for setups favouring the opponent, especially close threats, such as: it penalizes the score if the opponent has a near win (-50 points) or a potential connection (-10 points). And so it encourages the AI to block opponent threats, prioritizing imminent dangers. This guides the AI into giving preference to advantageous or promising moves.

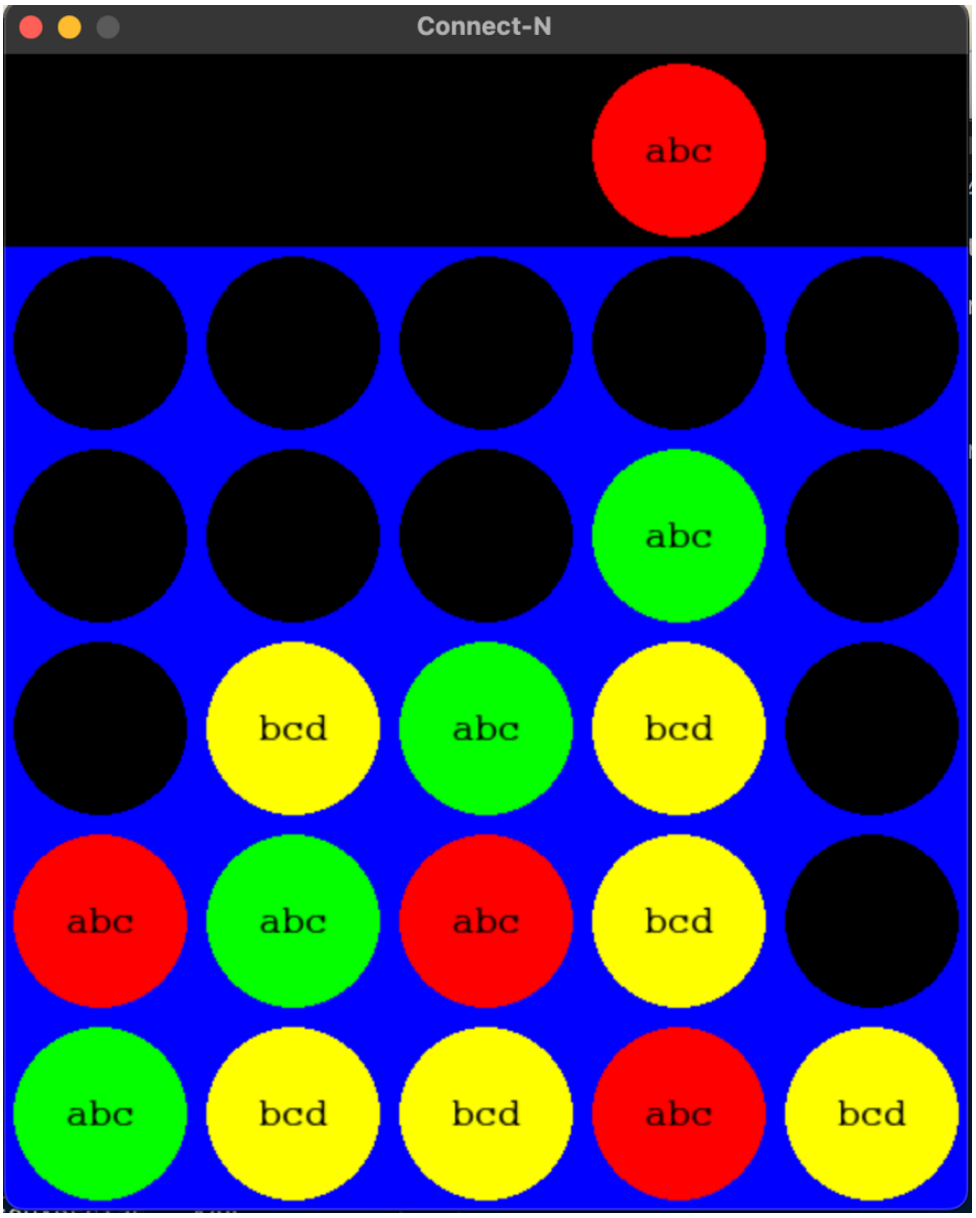
The different difficulty modes are based on these functions too. In the easy mode, random moves are executed. In the medium mode, minimax with alpha-beta pruning is utilised, with **depth 3**. In the hard mode too, minimax with alpha-beta pruning is utilised, but with **depth 7**, which makes the computer’s playing more efficient and thereby increases difficulty for the player.

Demo of the Code

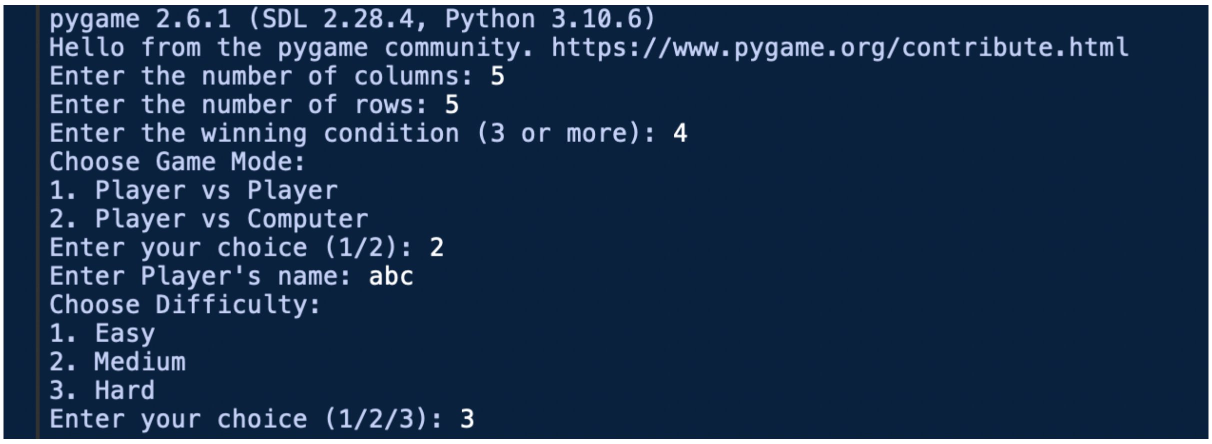
PvP mode:



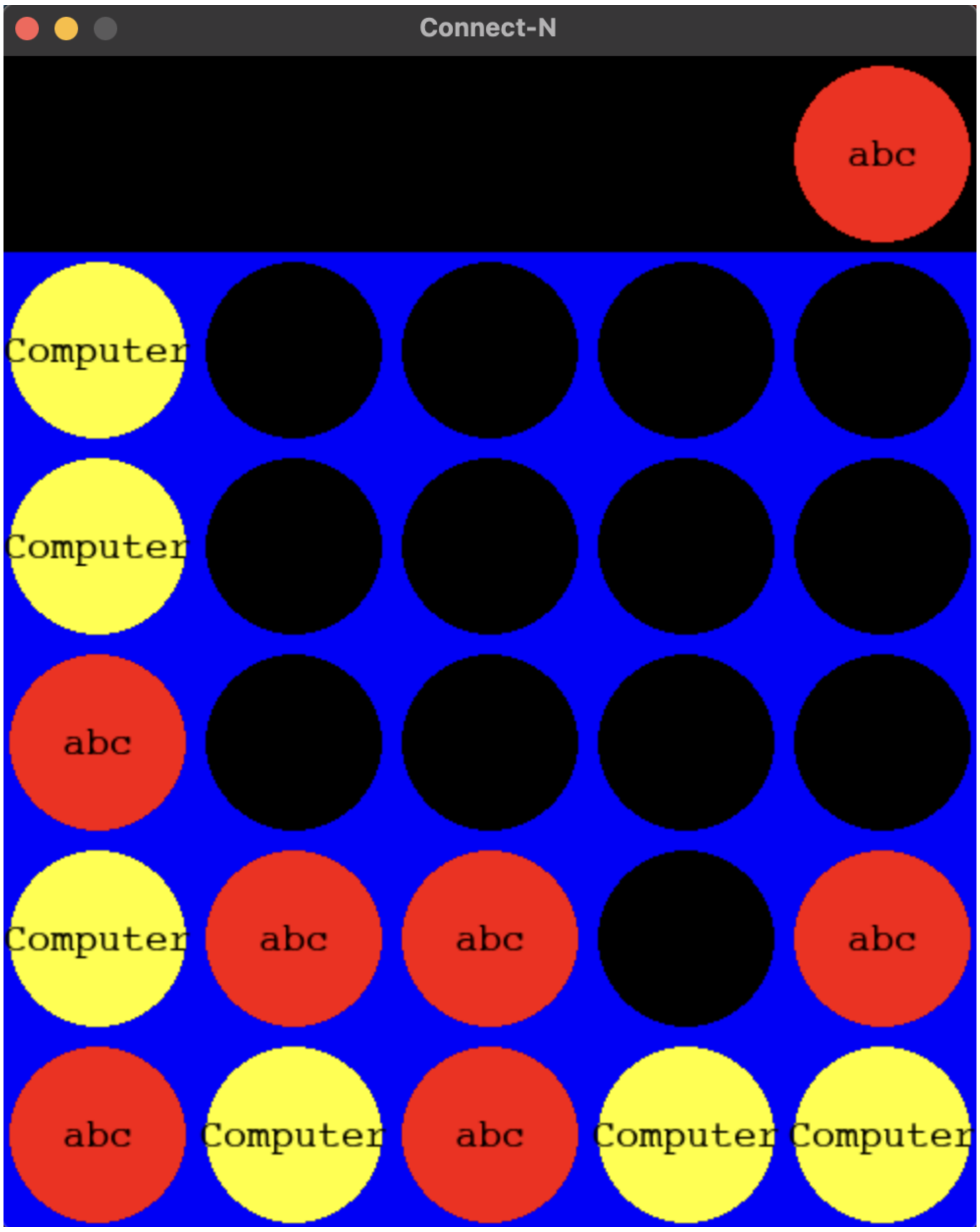
```
pygame 2.6.1 (SDL 2.28.4, Python 3.10.6)
Hello from the pygame community. https://www.pygame.org/contribute.html
Enter the number of columns: 5
Enter the number of rows: 5
Enter the winning condition (3 or more): 4
Choose Game Mode:
1. Player vs Player
2. Player vs Computer
Enter your choice (1/2): 1
Enter Player 1's name: abc
Enter Player 2's name: bcd
```



PvC mode:



```
pygame 2.6.1 (SDL 2.28.4, Python 3.10.6)
Hello from the pygame community. https://www.pygame.org/contribute.html
Enter the number of columns: 5
Enter the number of rows: 5
Enter the winning condition (3 or more): 4
Choose Game Mode:
1. Player vs Player
2. Player vs Computer
Enter your choice (1/2): 2
Enter Player's name: abc
Choose Difficulty:
1. Easy
2. Medium
3. Hard
Enter your choice (1/2/3): 3
```



Suggestions and Improvements

The AI algorithmic functions could be made more adaptive in nature. For ex: the computer's difficulty could be dynamically adjusted based on player performance. Instead of fixed difficulty levels, a provision could be made where specific AI instructions are provided provide e.g., offensive, defensive, or random, etc
A tutorial mode could be introduced for someone who’s new to the game, in which hints or some other form of visual could be provided.
Some other game-specific features like a timer to keep track of time or an undo/redo button could be facilitated too.

References

https://www.mililink.com/upload/article/279817393aams_vol_216_april_2022_a25_p3303-3313_kavita_sheoran,_et_al..pdf
http://www.warse.org/IJATCSE/static/pdf/file/ijatcse181022021.pdf
https://web.stanford.edu/class/archive/cs/cs221/cs221.1192/2018/restricted/posters/yuex/poster.pdf
https://roadtolarissa.com/connect-4-ai-how-it-works/
https://en.wikipedia.org/wiki/Connect_Four
https://www.lovetoknow.com/life/lifestyle/connect-four-games

Thank you.