Gobblet AI Player

In this project, we will explore the game-playing of **Gobblet,** Gobblet is an abstract game played on a 4x4 grid with each of the two players having twelve pieces that can nest on top of one another to create three stacks of four pieces.

Your goal in Gobblet is to place four of your pieces in a horizontal, vertical, or diagonal row. Your pieces start nested off the board. On a turn, you either play one exposed piece from your three off-the-board piles or move one piece on the board to any other spot on the board where it fits. A larger piece can cover any smaller piece. A piece being played from off the board may not cover an opponent's piece unless it's in a row where your opponent has three of his color.

Your memory is tested as you try to remember which color one of your larger pieces is covering before you move it. As soon as a player has four like-colored pieces in a row, he wins — except in one case: If you lift your piece and reveal an opponent's piece that finishes a four-in-a-row, you don't immediately lose; you can't return the piece to its starting location, but if you can place it over one of the opponent's three other pieces in that row, the game continues..

Components

16-square playing board
12 white Gobblets
12 black Gobblets

Game rules: RULES OF THE GAME
Video: https://www.youtube.com/watch?v=aSaAjQY8_b0

The GUI

You will need to create a GUI that allows for human vs. human, human vs. computer, and computer vs. computer gameplay. The GUI must include the following main features:

1. Board: Display the current game board and the current pieces on the board.
2. Move input: Allow human players to input their moves by clicking on the board
3. Game status: Whose turn it is, and if the game is over.
4. Game options: Allow players to choose different game modes (e.g., human vs. human, human vs. computer, computer vs. computer), choose the AI player difficulty level (for each AI player), and start/restart a new game.
5. The Project can be built using any programming language and framework of your choice.

Game Playing Algorithms

1. The minimax algorithm: a basic search algorithm that examines all possible moves from a given position and selects the move that leads to the best outcome for the current player
2. Alpha-beta pruning: an improvement on the minimax algorithm that can reduce the number of nodes that need to be searched.
3. Alpha-beta pruning with iterative deepening (depth is increased iteratively in the search tree until the timing constraints are violated)

Heuristics

Use at least one heuristic, you need to add the heuristic description and approximations taken in your documentation, You can also feel free to add more heuristics.

Collaboration

- Teams of 6-8 maximum.
- GitHub must be used.
- The GitHub repo must include a README file that illustrates all the project features and includes a user manual.
- Clear commits and comments for each team member are mandatory, to show the collaboration.
- The GitHub repo must be private till the submission date.

Plagiarism

You cannot copy code / external work and claim it as yours (even with slight modifications like changing variable names). Plagiarism will not be tolerated. Your work will be checked for plagiarism:

- Manually.
- Using software tools.

Results Evaluation and Submission

Game EXE (make sure it's working without any dependencies).
Pdf to the LMS that includes the following:

- The table that includes the team members, their IDs, and their contributions.
- GitHub link (don't forget to turn it into a public repo after submission time).
- The chosen programming language and framework of your choice.
- UML diagrams that illustrate the design (class, sequence, state).
- Game-playing supported algorithms.
- Used heuristics, their description, and the benefits of using them.
- Supported features.
- Maximum difficulty level supported (it's expected that the difficulty level reflects a deeper search in the tree).
- The level at which it becomes tough to play against the AI (the AI will win every time).
- Link to 2 minute YouTube video that shows the following game modes:
    - Human vs low difficulty level AI (Human must win).
    - Human vs high difficulty level AI (AI must win).

Evaluation: Based on the above submissions, contributions, and the code clearness (well written and commented).