HIVE AI Player

In this project, we will explore the game-playing of HIVE, a game of placing and moving bugs, represented by tiles, to surround your opponent's queen bee.
Players either place a new piece from their collection onto the table on each turn or move one they placed in a previous turn. Each piece—a beetle, ant, or grasshopper—has a unique way of moving on the table, and when a piece moves, it must not leave any other pieces stranded from the group ("breaking the hive"). Here:
https://en.boardgamearena.com/gamepanel?game=hive
you can play the game for yourself to understand it better and also explore the how to play section for more details on the game rules.
Game rules: https://www.ultraboardgames.com/hive/game-rules.php

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: Display the current score, 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.

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)

Results Evaluation

You should do it similarly to what is described in the paper linked previously (refer to section 6 in the paper)
You can use any other paper if needed (don't forget to include it in the submissions).

Collaboration

- Teams of 6-10 maximum.
- GitHub must be used.
- The GitHub repo must include a README file that illustrates all the project features and includes a user manual.
- To show collaboration, clear commits and comments for each team member are mandatory.
- 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.

However, you can learn the ideas from external sources, and then write your code.

Submission

Compressed file containing:

1. Game EXE (make sure it's working without any dependencies).
2. Pdf to the LMS that includes the following:
    - A 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 very hard 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).