**Modules**

* **watch YourBack**: This module contains functions that help run the game. Functions like placePiece(), movePiece(), comparePiece() help manage the pieces on the board. The board is managed by helper functions that return it size and shrinks the board when necessary. The module also consists of functions that help in keeping the piece inside the boundaries of board and helps us check if there is something that can kill us or can be attacked.
* **minimax**: This module implements the alpha beta pruning algorithm. The algorithm starts with a fixed depth of 3 and then we increase the depth as the board shrinks. The increase is done with the shrinking board because so that we can evaluate more states while usinga similar amount of memory as we were doing initially.
* **evaluationFunctions:** This module is used by our implementation of minimax. It gives each state a value so that the minimax algorithm can choose the best state and hence make the best move. The values returned by all functions in this module are normalized i.e. [-1,1].
* **strategy:** This module implements our strategy of attacking and defending alternatively. It basically gives the evaluation functions different weights as we switch between attacking and defending.
* **Player:** This module implements the player which player the game. The player uses all the modules to play the game. The player gets its move using the minimax algorithm which is governed by the evaluation functions that we have used.

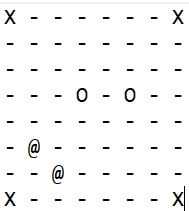
**Search strategy**

We search for the best possible next move by using the alpha-beta pruning algorithm. The algorithm goes through possible states until a depth of 3 and picks up the best state possible.

We also tried increasing the depth of our minimax tree up to 5 but it took a lot of time and space to run.

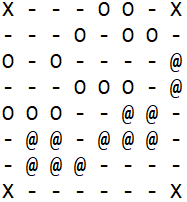
**Evaluation function**

* evalPartnerWarriorDiag(): This function tries to place or move a piece to an empty square which is diagonally surrounded by the most number of our pieces on the board.



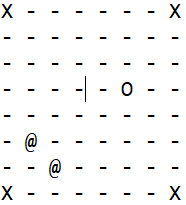
(if you are a white player) It will evaluate the state with the most score that either places a white piece at (4,2) or (4,4).

* PieceAdvantage() : This function returns the difference between the number of our players to the number of opponent players that are currently on the board.



(if you are a white player) The function will return 1 (13-12).

* distanceLine(): This function returns the distance of a white piece from the line (0,2) or the distance of the black piece from the line (0,6). This function is useful in the placing the phase because say for example you are player white then one the safest places on the board for you during the placing phase is when you are closest to the line (0.2) as a black piece cannot be placed anywhere above that line. Thus, it will give any state that places the white piece on (0,3) the highest value.
* evalFromCentre(): This function gives the maximum value to the state in which our player can be placed or move to an empty square that is near the center of the board e.g. (3.5,3.5) initially.
* deathtrap(): This function tries to place o move a piece such that two of our players have a gap on one square between them. This function might help us kill naïve players that might move in the between two of our players, hence the name deathtrap.



(if you are a white player and we are in the placing phase), this function will try placing a white piece at (3,3).

**Creative aspect**

The player uses the following strategies:

* + Attack: The attack mode is set when we have at least 4 more players than our opponent. The attack strategy gives different weights to different evaluation function, so the player chooses more of the attacking moves.
  + Defend: The defend mode is set through out when our (number of players+3) is less than the number of opponent players. The defense strategy sets weights to evaluation functions such that the player makes defensive moves.
  + GodMode: This mode tries to place or move all the pieces of our players to the squares (3,3), (4,3), (3,4), (4,4). As these squares are in the center of the board and all our pieces will be next to each other then there is no way the opponent can kill us. This is the best strategy to be in when the board has shrunk down the smallest size.

The depth of the tree of our minimax algorithm increases as the board shrinks which helps us make better and more efficient moves.

Thus, the strategies using the different evaluation functions which are weighted according to the strategies and exploring a large number of possible moves through alpha-beta pruning helps the player to choose the best possible move in a short period of time.