REGIONAL UNIVERSITY NETWORK
EUROPEAN UNIVERSITY

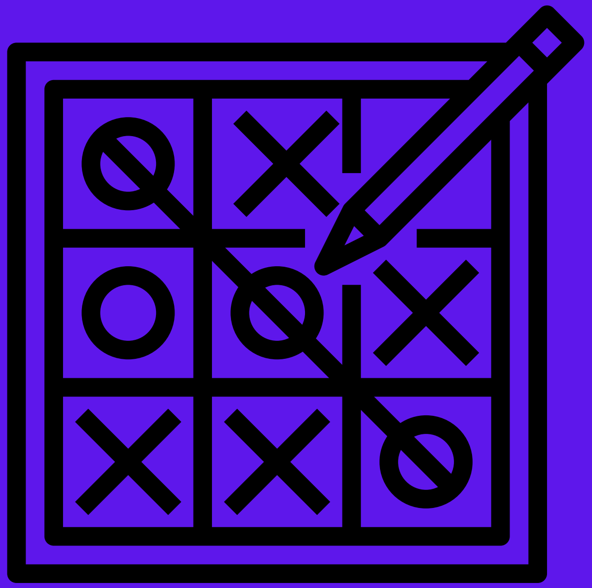
CHESS GAME

Python application

Adrian Florczak
Alberto Cuervo
Gloria Delgado

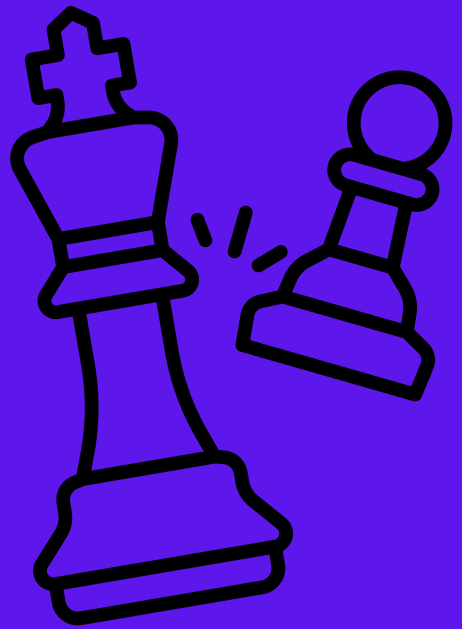
Our initial ideas & research

Tic Tac Toe



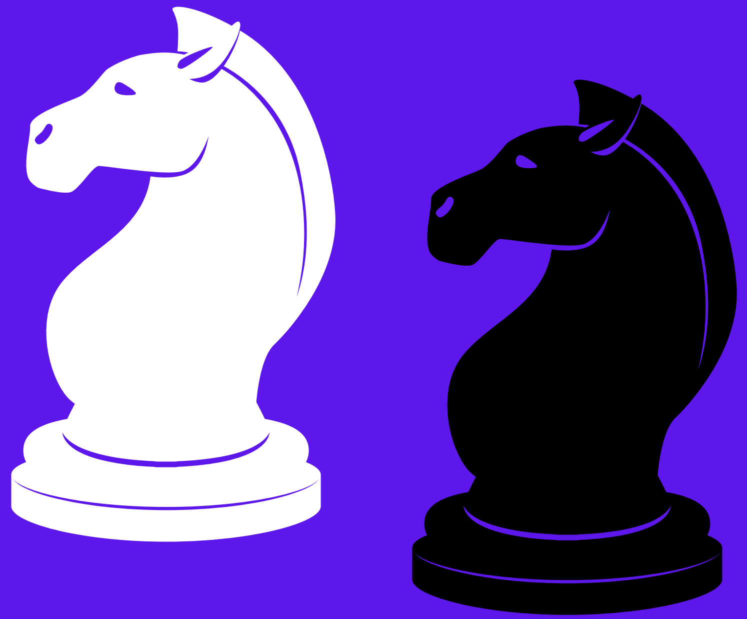
Most resources

Chess



Bigger challlange

Other chess-like game

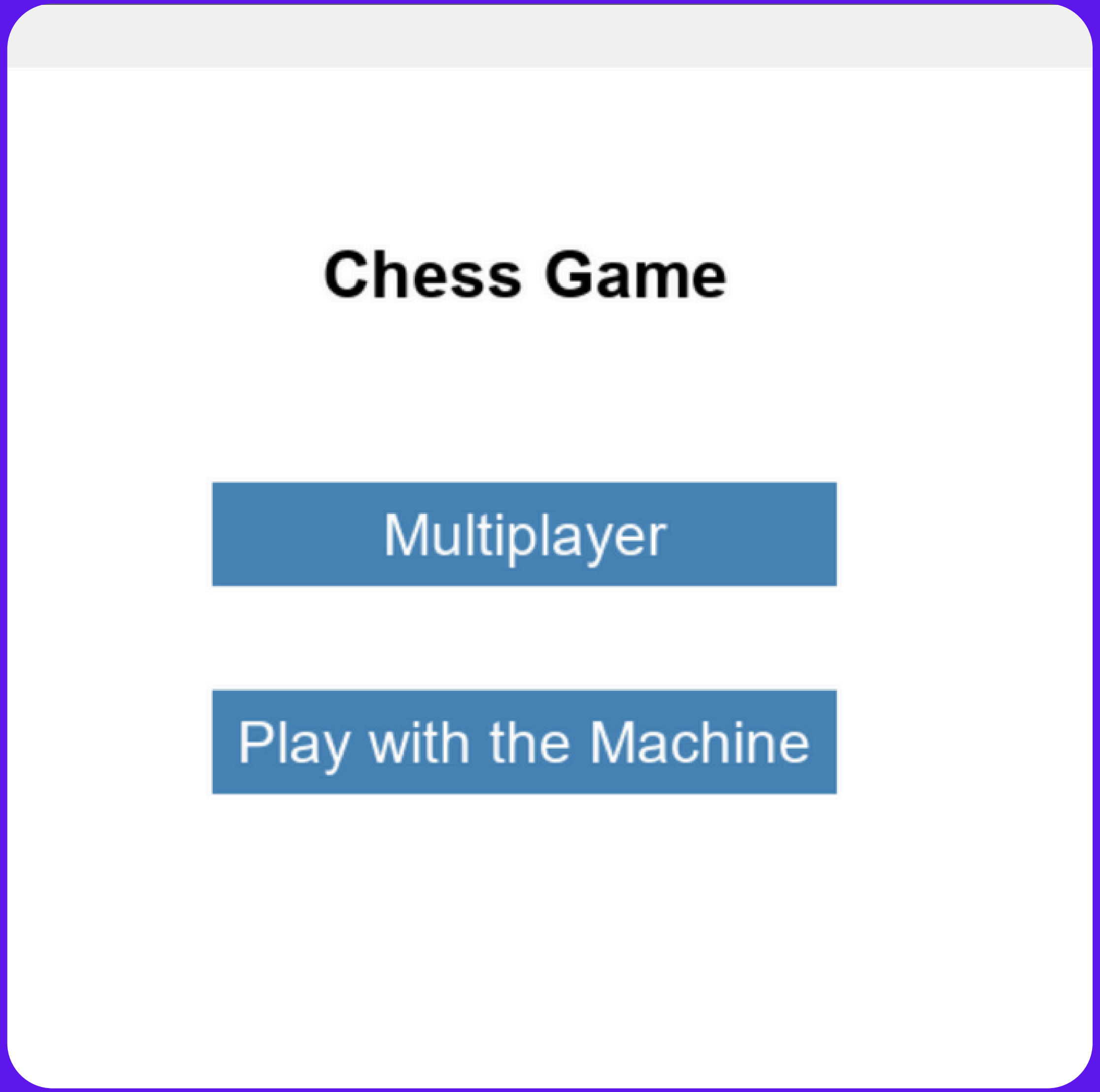


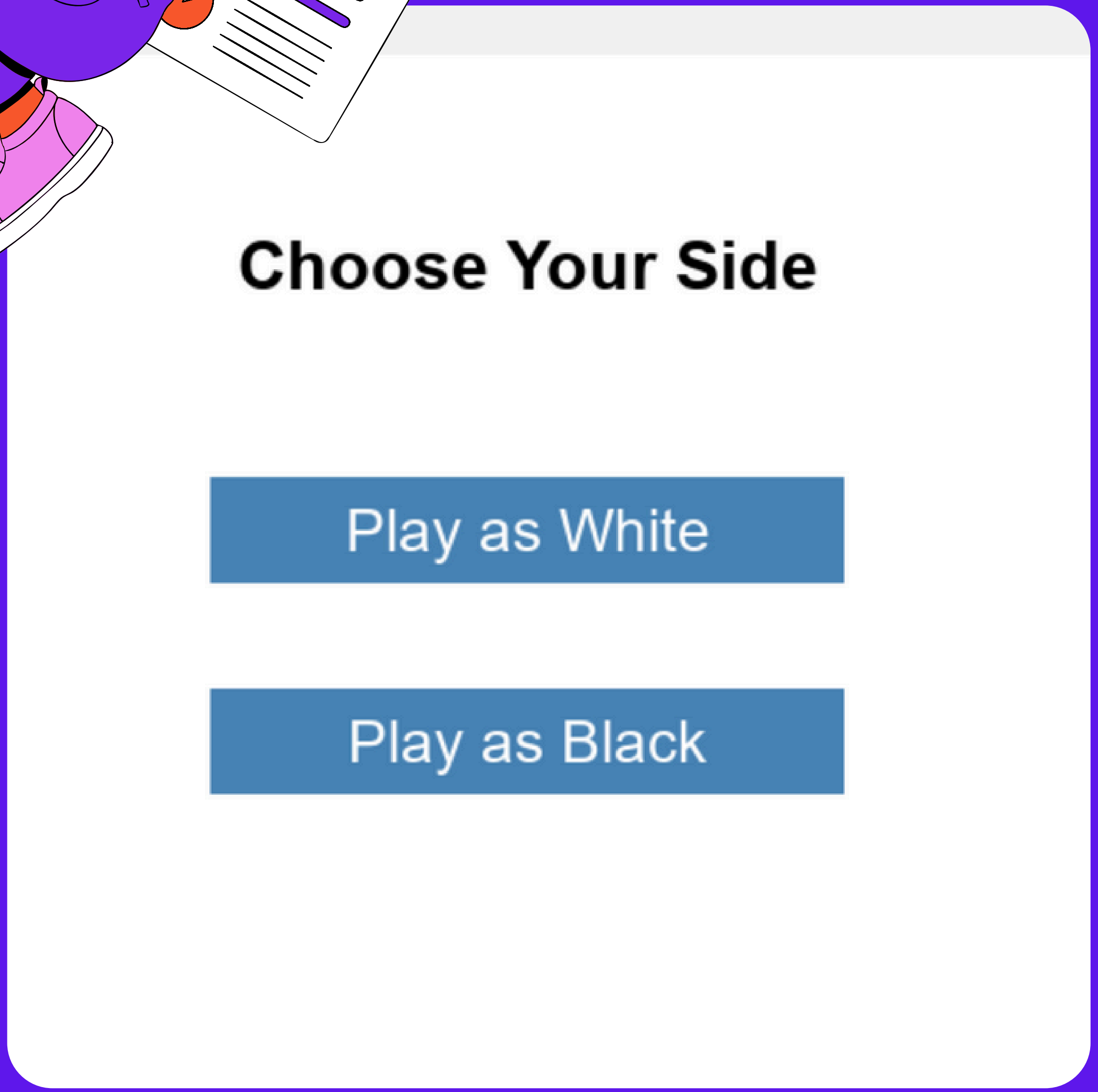
More ambitious

Main features

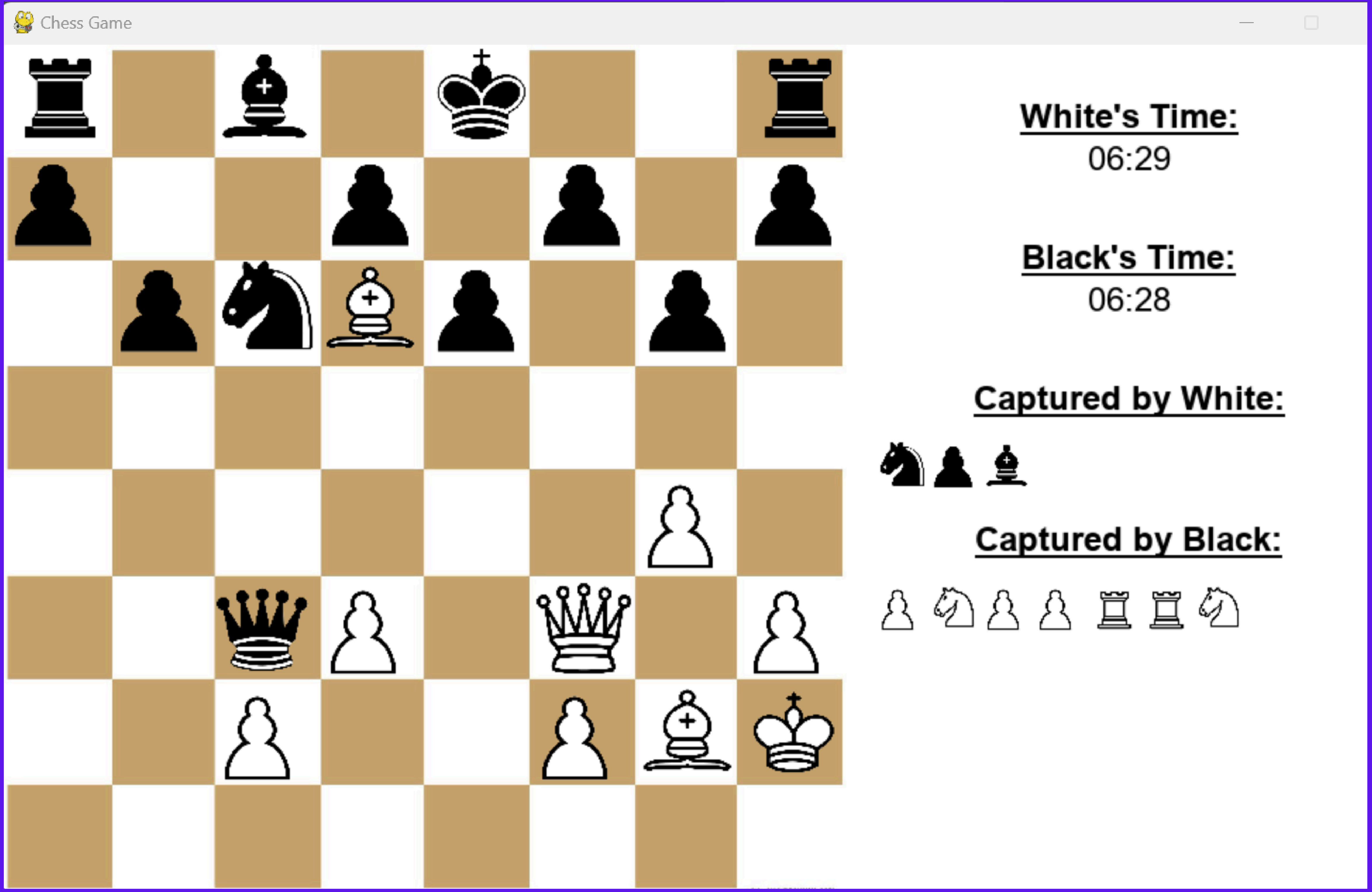
- Local multiplayer
- Playing against computer

INTERFACE


Chess Game


Choose Your Side

GAME EXPERIENCE



White's Time:
06:29

Black's Time:
06:28

Captured by White:

Captured by Black:

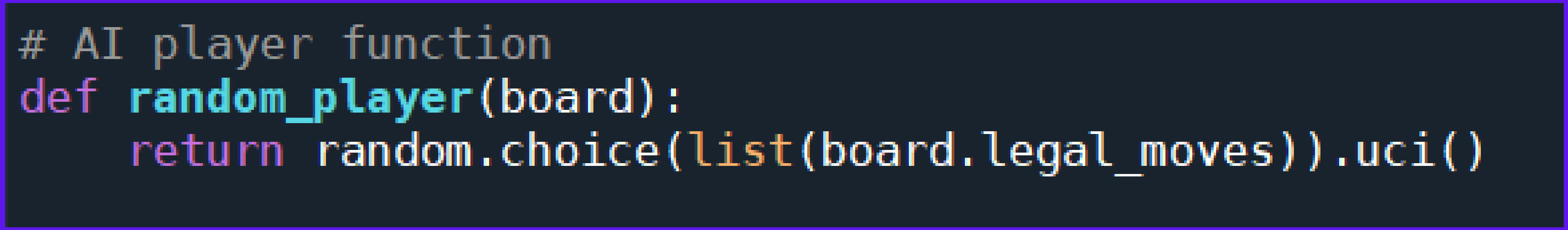
Computer game

- Random player
- Alphabeta algorithm

RANDOM PLAYER

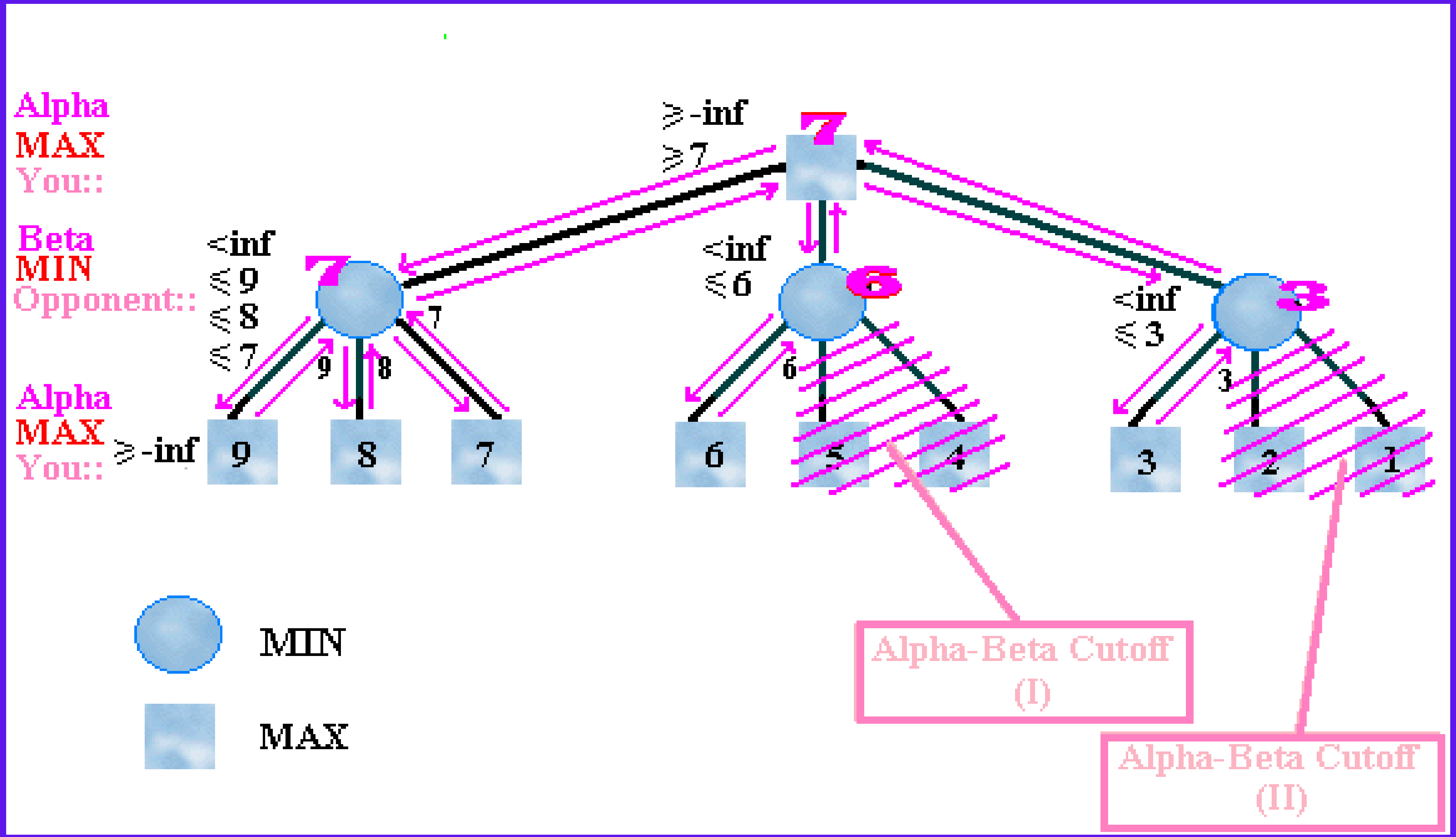
Problems found:

- Very random moves with out logic
- Some weird loops of movement
- As it is ‘random’ it allways started the same ways



```python
# AI player function
def random_player(board):
    return random.choice(list(board.legal_moves)).uci()
```

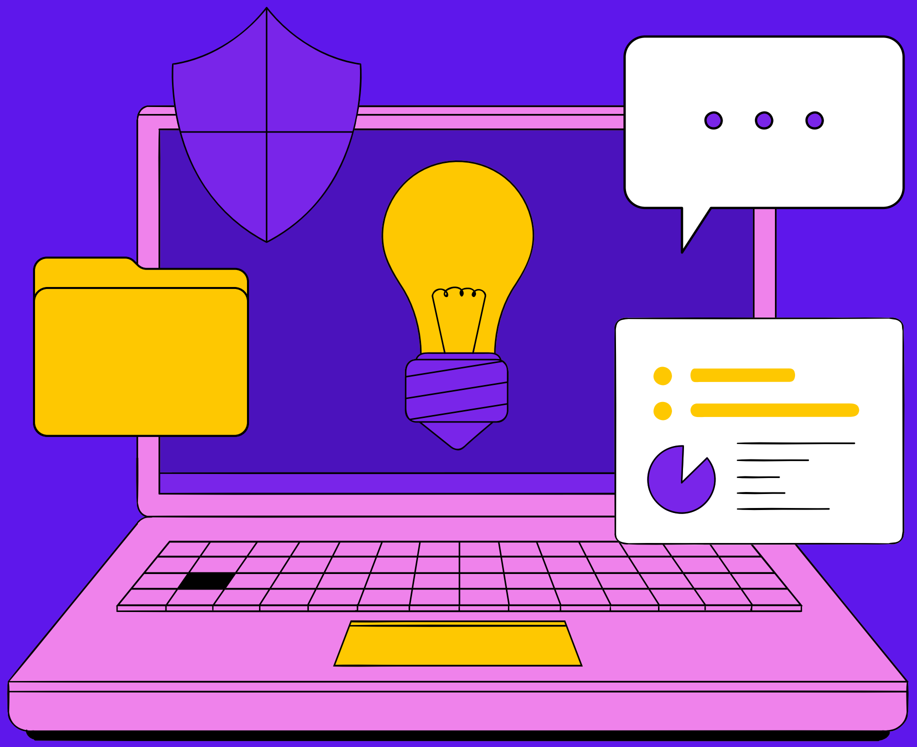
ALPHABETA ALGORITHM

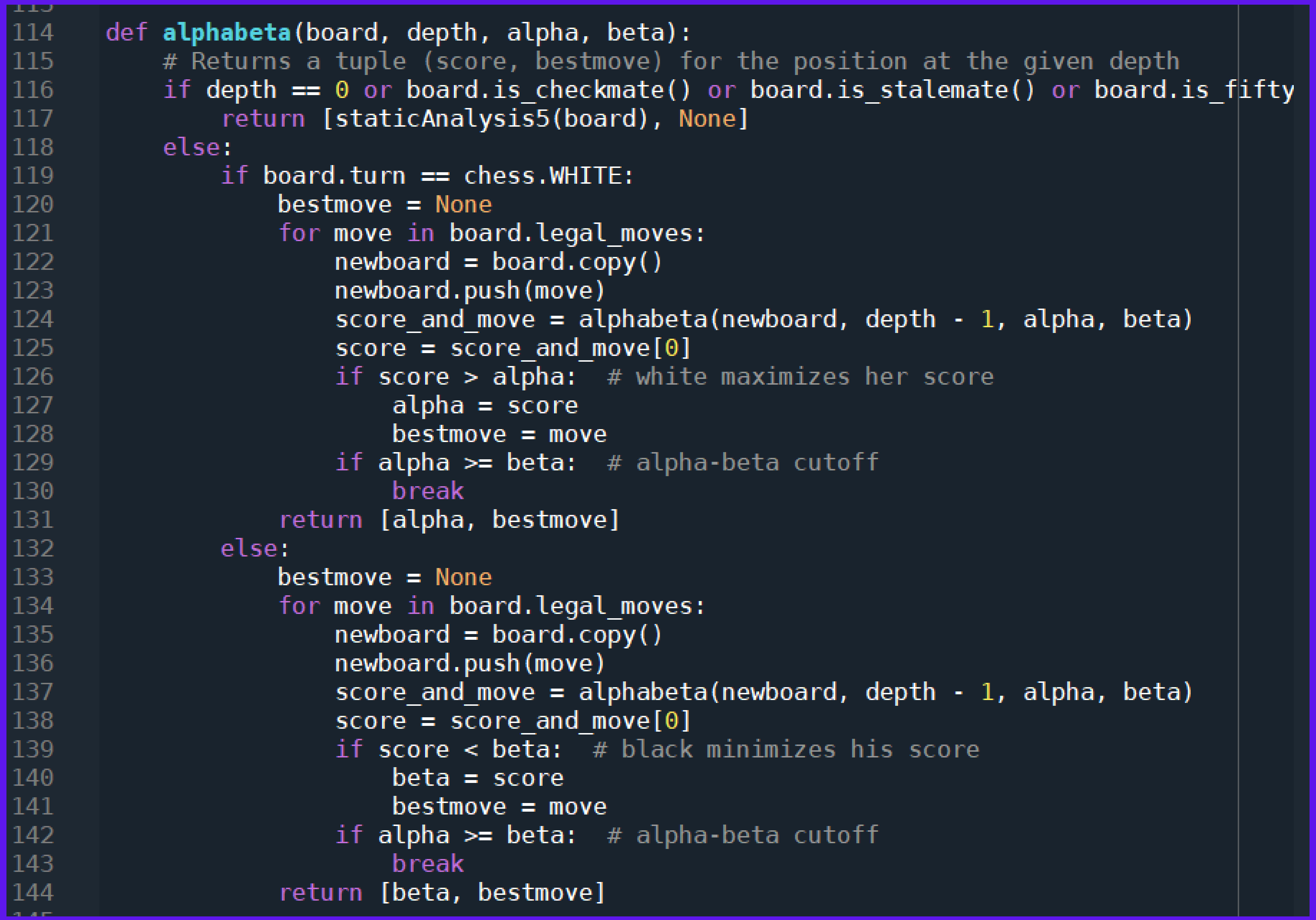


Resaerching some more we came across the minimax algorithm and that brought us to the alpha beta algorithm wich is an imprvement of it

ALPHABETA ALGORITHM

Implementation of the algorithm into our code





```
def alphabeta(board, depth, alpha, beta):
    # Returns a tuple (score, bestmove) for the position at the given depth
    if depth == 0 or board.is_checkmate() or board.is_stalemate() or board.is_fifty
        return [staticAnalysis5(board), None]
    else:
        if board.turn == chess.WHITE:
            bestmove = None
            for move in board.legal_moves:
                newboard = board.copy()
                newboard.push(move)
                score_and_move = alphabeta(newboard, depth - 1, alpha, beta)
                score = score_and_move[0]
                if score > alpha:  # white maximizes her score
                    alpha = score
                    bestmove = move
                if alpha >= beta:  # alpha-beta cutoff
                    break
            return [alpha, bestmove]
        else:
            bestmove = None
            for move in board.legal_moves:
                newboard = board.copy()
                newboard.push(move)
                score_and_move = alphabeta(newboard, depth - 1, alpha, beta)
                score = score_and_move[0]
                if score < beta:  # black minimizes his score
                    beta = score
                    bestmove = move
                if alpha >= beta:  # alpha-beta cutoff
                    break
            return [beta, bestmove]
```

Live demonstration

Special thanks to
OUR TESTERS

Some of the testers were defeated by our computer algorithm:

As European Law the tester has been censored for their own privacy
Regulation (EU) 2016/679

REGIONAL UNIVERSITY NETWORK
EUROPEAN UNIVERSITY

THE END