```
# prompt: code for naughts and crushes with alpha beta pruning and one player as ai and also show move of ai

import random

def print_board(board):
    print("-------------")
    for i in range(3):
        print("|", board[i*3], "|", board[i*3 + 1], "|", board[i*3 + 2], "|")
        print("-------------")

def check_win(board, player):
    win_combinations = [(0, 1, 2), (3, 4, 5), (6, 7, 8),
                        (0, 3, 6), (1, 4, 7), (2, 5, 8),
                        (0, 4, 8), (2, 4, 6)]
    for combo in win_combinations:
        if all(board[i] == player for i in combo):
            return True
    return False

def check_draw(board):
    return all(cell != ' ' for cell in board)

def evaluate(board):
    if check_win(board, 'X'):
        return 1
    elif check_win(board, 'O'):
        return -1
    else:
        return 0

def minimax(board, depth, maximizing_player, alpha, beta):
    if depth == 0 or check_win(board, 'X') or check_win(board, 'O') or check_draw(board):
        return evaluate(board), None

    if maximizing_player:
        max_eval = -float('inf')
        best_move = None
        for i in range(9):
            if board[i] == ' ':
                board[i] = 'X'
                eval, _ = minimax(board, depth - 1, False, alpha, beta)
                board[i] = ' '
                if eval > max_eval:
                    max_eval = eval
                    best_move = i
                alpha = max(alpha, eval)
                if beta <= alpha:
                    break
        return max_eval, best_move
    else:
        min_eval = float('inf')
        best_move = None
        for i in range(9):
            if board[i] == ' ':
                board[i] = 'O'
                eval, _ = minimax(board, depth - 1, True, alpha, beta)
                board[i] = ' '
                if eval < min_eval:
                    min_eval = eval
                    best_move = i
                beta = min(beta, eval)
                if beta <= alpha:
                    break
        return min_eval, best_move


def ai_move(board):
  _, move = minimax(board, 9, True, -float('inf'), float('inf'))
  return move

def play_game():
    board = [' '] * 9
    current_player = 'X'

    while True:
        print board(board)
```

```
        if current_player == 'X':
            move = ai_move(board)
            print(f"AI plays at position {move + 1}")
            board[move] = current_player
        else:
            while True:
                try:
                    move = int(input("Enter your move (1-9): ")) - 1
                    if 0 <= move <= 8 and board[move] == ' ':
                        board[move] = current_player
                        break
                    else:
                        print("Invalid move. Try again.")
                except ValueError:
                    print("Invalid input. Please enter a number.")

        if check_win(board, current_player):
            print_board(board)
            print(f"{current_player} wins!")
            break
        elif check_draw(board):
            print_board(board)
            print("It's a draw!")
            break

        current_player = 'O' if current_player == 'X' else 'X'

if _name_ == "_main_":
    play_game()
```

Analyze files with Gemini