



- **Peguy Rusty Kana Donwoung**
- **Nkwanga Nkwanga Mansvell**
- **Atoundem Sonfack Auguste**
- **Nya Ken Ulrich**

Aufgabe 1 (Strategie– Vier Gewinnt). Spielen Sie in Ihrer Gruppe das Spiel „Vier Gewinnt“ und versuchen Sie, Strategien bzw. Heuristiken für gute Spielzüge zu entwickeln und zu notieren. Implementieren Sie Ihre Heuristiken als Bewertungsfunktion für eine KI, die damit aus allen möglichen nächsten Zügen den besten Zug auswählt (anstelle des Zufalls). Verwenden Sie dafür die bereits angefangene Vier-Gewinnt-Implementierung auf Moodle. Unter <https://kimaster.mni.thm.de/> finden Sie eine Online-Plattform für das Spiel. Hinweis: Ein Kriterium für eine gute Heuristik ist z.B. die Anzahl an erweiterbaren Dreierreihen.

```
def score_board(self):
    """
    Evaluates the current board state and returns a score.

    What it does:
    - Assigns a positive score if Player 1 is in a good position.
    - Assigns a negative score if Player 2 is in a good position.
    - Favors moves in the center column since these give more opportunities to connect four.

    Returns:
    - A score that tells how good the board is for the current player.
    """

    score = 0

    # Prüfe, ob der Gegner im nächsten Zug gewinnen kann
    for move in self.get_valid_moves():
        row = self._get_row_for_move(move)
        self.board[row, move] = -self.players_turn
        if self.__check_win(row, move): # Prüfe, ob der nächste Gegner gewinnen kann
            score -= 10000 # massive Strafe
            self.board[row, move] = 0 # den simulierten Zug löschen

    # die mittlere Spalte fördern
    center_col = self.board[:, self.n_col // 2]
    center_count = np.count_nonzero(center_col == self.players_turn)
    score += center_count * 20 # Weight for the central column

    # Analysiert die Möglichkeiten von dem Spieler
    score += self._evaluate_lines(self.players_turn)

    # Reduziert die Möglichkeiten von dem Gegner
    score -= self._evaluate_lines(-self.players_turn)

    return score

def _evaluate_lines(self, player):
    score = 0
    directions = [(0, 1), (1, 0), (1, 1), (1, -1)] # Horizontal, Vertikal, Diagonalen
    for row in range(self.n_row):
        for col in range(self.n_col):
            if self.board[row, col] == player:
                for dx, dy in directions:
                    count = self._count_consecutive_discs(row, col, dx, dy)
                    if count == 2:
                        score += 30 # für zwei Züge erhöhen
                    elif count == 3:
                        score += 300 # Für 3 priorisieren
                    elif count >= 4:
                        score += 10000 # den Erfolg sichern

    return score

ConnectFour._evaluate_lines = _evaluate_lines
ConnectFour.score_board = score_board
```



Aufgabe 2 (Text-Animation– Vier Gewinnt).Erweitern Sie Ihre Vier Gewinnt-Implementierung derart, dass der Zustand des „Spielbrettes“ in der Textkonsole ausgegeben werden kann.

```
def display_board(self):
    """
    Displays the current state of the Connect Four board in the console.
    """
    # Numerisiert die Werte
    symbol_map = {0: '.', self.PLAYER_1: 'X', self.PLAYER_2: 'O'}

    # die Abbildung erstellen
    for row in self.board:
        print(' '.join(symbol_map[cell] for cell in row))

    # Zeile für die Spalten abbilden
    print('-' * (self.n_col * 2 - 1)) #Trennungszeile
    print(' '.join(str(i) for i in range(self.n_col))) # Zahl der Zeile
ConnectFour.display_board=display_board
```


Ausgabe:

KI (Minimax) chooses column 3

.
.
.
.
.	.	.	X	.	.	.
0	.	.	X	.	.	.

0	1	2	3	4	5	6

Aufgabe 3 (Suchbaum). Ihre KI ist derzeit nur in der Lage, den nächsten Zug zu bewerten. Implementieren Sie eine Funktion, die auch Folgezüge bewertet und mithilfe eines Suchbaums Entscheidungen trifft. Hinweis: Nutzen Sie das Pseudo-Code-Beispiel zum α - β -Pruning aus der Vorlesung und erweitern Sie das jupyterlab-Beispiel auf Moodle (einige Funktionen haben zur Zeit nur eine textuelle Beschreibung, aber keinen Code).

```
def beta_max(self, depth, alpha, beta):
    if depth == 0 or self.__check_win_state(): # Basisfall: Max. Tiefe oder Spielende
        return self.score_board()

    valid_moves = self.get_valid_moves()
    max_eval = float('-inf') # Initialisierung
    for move in valid_moves:
        row = self._get_row_for_move(move)
        self.board[row, move] = self.PLAYER_1 # Simuliere den Zug
        eval = self.beta_min(depth - 1, alpha, beta) # Rufe beta_min auf
        self.board[row, move] = 0 # Rückgängig machen des Zugs

        max_eval = max(max_eval, eval) # Aktualisiere max_eval
        alpha = max(alpha, eval) # Aktualisiere alpha
        if beta <= alpha: # Pruning
            break

    return max_eval

def beta_min(self, depth, alpha, beta):
    if depth == 0 or self.__check_win_state(): # Basisfall: Max. Tiefe oder Spielende
        return self.score_board()

    valid_moves = self.get_valid_moves()
    min_eval = np.inf # Initialisierung
    for move in valid_moves:
        row = self._get_row_for_move(move)
        self.board[row, move] = self.PLAYER_2 # Simuliere den Zug
        eval = self.beta_max(depth - 1, alpha, beta) # Rufe beta_max auf
        self.board[row, move] = 0 # Rückgängig machen des Zugs

        min_eval = min(min_eval, eval) # Aktualisiere min_eval
        beta = min(beta, eval) # Aktualisiere beta
        if beta <= alpha: # Pruning
            break

    return min_eval

ConnectFour.beta_max = beta_max
ConnectFour.beta_min = beta_min
```


Aufgabe 4 (Testen Ihrer KI). Lassen Sie Ihre KI gegen die KI auf der Webseite <https://kimaster.mni.thm.de/> spielen. Hierzu müssen Sie die Web Socket-API einbinden und die richtigen Kommandos ausführen. Passende Funktionen finden Sie schon in dem Code-Beispiel aus Aufgabe 2. Wie gut ist Ihre KI im Vergleich zu den verschiedenen Schwierigkeitsstufen?



DANKE