AI Othello Players

0.1

Generated by Doxygen 1.8.10

Contents

Chapter 1

Class Index

1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 2

Class Documentation

2.1 TabuleiroOthello-fork.models.players.corner_player.CornerPlayer Class Reference

Public Member Functions

- def **__init__** (self, color)
- def **play** (self, board)
- def **getNearestCorner** (self, moves)

Public Attributes

- **color**

The documentation for this class was generated from the following file:

- /home/ambrozio/Development/Atom/IA/TabuleiroOthello-fork/models/players/corner_player.py

2.2 TabuleiroOthello-fork.models.players.manhattan_corner_player.ManhattanCorner↩ Player Class Reference

Public Member Functions

- def **__init__** (self, color)
- def **play** (self, board)
- def **getNearestCorner** (self, moves)

Public Attributes

- **color**

The documentation for this class was generated from the following file:

- /home/ambrozio/Development/Atom/IA/TabuleiroOthello-fork/models/players/manhattan_corner_player.py

2.3 TabuleiroOthello-fork.models.players.mixed_heuristic_player.Minimax1111Player Class Reference

Public Member Functions

- def __init__ (self, color)
- def play (self, board)
- def board_value_table (self, board, player_color)
- def board_value_pieces (self, board, player_color)
- def board_value_minimize (self, board, player_color)
- def max_move (self, board, player_color, depth, alpha, beta)
- def min_move (self, board, player_color, depth, alpha, beta)
- def minimax (self, board, player_color)
- def heuristic (self, move)
- def gameover (self, board)

Public Attributes

- **color**
- **max_depth**
- **heuristic_type**

2.3.1 Detailed Description

```
Documentation for the minimax with alpha-beta pruning AI player.

This class is the implementation of the minimax algorithm with alpha-beta pruning
for improved speed and greater depth search.
This class uses heuristic 4 (mixed).
```

2.3.2 Constructor & Destructor Documentation

2.3.2.1 def TabuleiroOthello-fork.models.players.mixed_heuristic_player.Minimax1111Player.__init__ (*self*, *color*)

```
Constructor.
```

2.3.3 Member Function Documentation

2.3.3.1 def TabuleiroOthello-fork.models.players.mixed_heuristic_player.Minimax1111Player.board_value_minimize (*self*, *board*, *player_color*)

```
This function returns the value of the board received as argument.
It is the number of valid moves for the opponent.
```

2.3.3.2 def TabuleiroOthello-fork.models.players.mixed_heuristic_player.Minimax1111Player.board_value_pieces (*self*, *board*, *player_color*)

```
This function returns the value of the board received as argument.
It is the sum of all positions owned by that player_color minus the sum of
positions owned by the opposition. Blank positions do not count.
```

2.3.3.3 def TabuleiroOthello-fork.models.players.mixed_heuristic_player.Minimax1111Player.board_value_table (*self*, *board*, *player_color*)

```
This function returns the value of the board received as argument.
It is the sum of all positions that player_color has minus the sum of
positions that the opposition has. Blank positions do not count.
```

2.3.3.4 def TabuleiroOthello-fork.models.players.mixed_heuristic_player.Minimax1111Player.gameover (*self*, *board*)

```
This function returns +1 if white player wins, -1 if black player wins,
0 if game is not over and +2 if game is a tie
```

2.3.3.5 def TabuleiroOthello-fork.models.players.mixed_heuristic_player.Minimax1111Player.heuristic (*self*, *move*)

```
This function returns the value of the move, according to the heuristic table below:
100,   0,   6,   5,   5,   6,   0, 100
  0,   0,   8,   3,   3,   8,   0,   0
  3,   7,   3,   2,   2,   3,   7,   3
  4,   3,   2,   1,   1,   2,   3,   4
  4,   3,   2,   1,   1,   2,   3,   4
  3,   7,   3,   2,   2,   3,   7,   3
  0,   0,   8,   3,   3,   8,   0,   0
100,   0,   6,   5,   5,   6,   0, 100
```

2.3.3.6 def TabuleiroOthello-fork.models.players.mixed_heuristic_player.Minimax1111Player.max_move (*self*, *board*, *player_color*, *depth*, *alpha*, *beta*)

```
This is the max part of the alpha-beta pruning. It plays every valid
move available for player_color (this player) until the game is over, if a move ends the game
or the depth is reached. It calls the min part of the alpha-beta algorithm.
During the recursive part, the return value (integer) of this function is the current best value.
If depth is 0, meaning the end of the recursive part, it returns the best move (Move object)
```

2.3.3.7 def TabuleiroOthello-fork.models.players.mixed_heuristic_player.Minimax1111Player.min_move (*self*, *board*, *player_color*, *depth*, *alpha*, *beta*)

```
This is the min part of the alpha-beta pruning. It plays every valid
move available for player_color (oppenent) until the game is over, if a move ends the game
or the depth is reached. It calls the max part of the alpha-beta algorithm.
Return value (integer) of this function is the current best value (beta).
Very similar with the max function.
```

2.3.3.8 def TabuleiroOthello-fork.models.players.mixed_heuristic_player.Minimax1111Player.minimax (*self*, *board*, *player_color*)

```
Wrapper function to call max and get the best move.
```

2.3.3.9 def TabuleiroOthello-fork.models.players.mixed_heuristic_player.Minimax1111Player.play (*self*, *board*)

```
This is the function called by the board controller. Returns a Move.
```

The documentation for this class was generated from the following file:

- /home/ambrozio/Development/Atom/IA/TabuleiroOthello-fork/models/players/mixed_heuristic_player.py

2.4 TabuleiroOthello-fork.models.players.minimize_moves_player.Minimax111Player Class Reference

Public Member Functions

- def __init__ (self, color)
- def play (self, board)
- def board_value (self, board, player_color)
- def max_move (self, board, player_color, depth, alpha, beta)
- def min_move (self, board, player_color, depth, alpha, beta)
- def minimax (self, board, player_color)
- def gameover (self, board)

Public Attributes

- **color**
- **max_depth**

2.4.1 Detailed Description

```
Documentation for the minimax with alpha-beta pruning AI player.

This class is the implementation of the minimax algorithm with alpha-beta pruning
for improved speed and greater depth search.
This class uses heuristic 3 (minimize).
```

2.4.2 Constructor & Destructor Documentation

2.4.2.1 def TabuleiroOthello-fork.models.players.minimize_moves_player.Minimax111Player.__init__ (*self*, *color*)

```
Constructor.
```

2.4.3 Member Function Documentation

2.4.3.1 def TabuleiroOthello-fork.models.players.minimize_moves_player.Minimax111Player.board_value (*self*, *board*, *player_color*)

```
This function returns the value of the board received as argument.
It is the number of valid moves for the opponent.
```

2.4.3.2 def TabuleiroOthello-fork.models.players.minimize_moves_player.Minimax111Player.gameover (*self*, *board*)

```
This function returns +1 if white player wins, -1 if black player wins,
0 if game is not over and +2 if game is a tie
```

2.4.3.3 def TabuleiroOthello-fork.models.players.minimize_moves_player.Minimax111Player.max_move (*self*, *board*, *player_color*, *depth*, *alpha*, *beta*)

```
This is the max part of the alpha-beta pruning. It plays every valid
move available for player_color (this player) until the game is over, if a move ends the game
or the depth is reached. It calls the min part of the alpha-beta algorithm.
During the recursive part, the return value (integer) of this function is the current best value.
If depth is 0, meaning the end of the recursive part, it returns the best move (Move object)
```

2.4.3.4 def TabuleiroOthello-fork.models.players.minimize_moves_player.Minimax111Player.min_move (*self*, *board*, *player_color*, *depth*, *alpha*, *beta*)

```
This is the min part of the alpha-beta pruning. It plays every valid
move available for player_color (oppenent) until the game is over, if a move ends the game
or the depth is reached. It calls the max part of the alpha-beta algorithm.
Return value (integer) of this function is the current best value (beta).
Very similar with the max function.
```

2.4.3.5 def TabuleiroOthello-fork.models.players.minimize_moves_player.Minimax111Player.minimax (*self*, *board*, *player_color*)

```
Wrapper function to call max and get the best move.
```

2.4.3.6 def TabuleiroOthello-fork.models.players.minimize_moves_player.Minimax111Player.play (*self*, *board*)

```
This is the function called by the board controller. Returns a Move.
```

The documentation for this class was generated from the following file:

- /home/ambrozio/Development/Atom/IA/TabuleiroOthello-fork/models/players/minimize_moves_player.py

2.5 TabuleiroOthello-fork.models.players.pieces_quantity_alpha_beta_player.Minimax11↩ Player Class Reference

Public Member Functions

- def __init__ (self, color)
- def play (self, board)
- def board_value (self, board, player_color)
- def max_move (self, board, player_color, depth, alpha, beta)
- def min_move (self, board, player_color, depth, alpha, beta)
- def minimax (self, board, player_color)
- def gameover (self, board)

Public Attributes

- **color**
- **max_depth**

2.5.1 Detailed Description

```
Documentation for the minimax with alpha-beta pruning AI player.

This class is the implementation of the minimax algorithm with alpha-beta pruning
for improved speed and greater depth search.
This class uses heuristic 2 (pieces).
```

2.5.2 Constructor & Destructor Documentation

2.5.2.1 def TabuleiroOthello-fork.models.players.pieces_quantity_alpha_beta_player.Minimax11Player.__init__ (*self*, *color*)

```
Constructor.
```

2.5.3 Member Function Documentation

2.5.3.1 def TabuleiroOthello-fork.models.players.pieces_quantity_alpha_beta_player.Minimax11Player.board_value (*self,* *board,* *player_color*)

```
This function returns the value of the board received as argument.
It is the sum of all positions owned by that player_color minus the sum of
positions owned by the opposition. Blank positions do not count.
```

2.5.3.2 def TabuleiroOthello-fork.models.players.pieces_quantity_alpha_beta_player.Minimax11Player.gameover (*self,* *board*)

```
This function returns +1 if white player wins, -1 if black player wins,
0 if game is not over and +2 if game is a tie
```

2.5.3.3 def TabuleiroOthello-fork.models.players.pieces_quantity_alpha_beta_player.Minimax11Player.max_move (*self,* *board,* *player_color,* *depth,* *alpha,* *beta*)

```
This is the max part of the alpha-beta pruning. It plays every valid
move available for player_color (this player) until the game is over, if a move ends the game
or the depth is reached. It calls the min part of the alpha-beta algorithm.
During the recursive part, the return value (integer) of this function is the current best value.
If depth is 0, meaning the end of the recursive part, it returns the best move (Move object)
```

2.5.3.4 def TabuleiroOthello-fork.models.players.pieces_quantity_alpha_beta_player.Minimax11Player.min_move (*self,* *board,* *player_color,* *depth,* *alpha,* *beta*)

```
This is the min part of the alpha-beta pruning. It plays every valid
move available for player_color (oppenent) until the game is over, if a move ends the game
or the depth is reached. It calls the max part of the alpha-beta algorithm.
Return value (integer) of this function is the current best value (beta).
Very similar with the max function.
```

2.5.3.5 def TabuleiroOthello-fork.models.players.pieces_quantity_alpha_beta_player.Minimax11Player.minimax (*self,* *board,* *player_color*)

```
Wrapper function to call max and get the best move.
```

2.5.3.6 def TabuleiroOthello-fork.models.players.pieces_quantity_alpha_beta_player.Minimax11Player.play (*self,* *board*)

```
This is the function called by the board controller. Returns a Move.
```

The documentation for this class was generated from the following file:

- /home/ambrozio/Development/Atom/IA/TabuleiroOthello-fork/models/players/pieces_quantity_alpha_beta↩ _player.py

2.6 TabuleiroOthello-fork.models.players.table_minimax_alpha_beta_player.Minimax1↩ Player Class Reference

Public Member Functions

- def __init__ (self, color)

- def play (self, board)
- def board_value (self, board, player_color)
- def max_move (self, board, player_color, depth, alpha, beta)
- def min_move (self, board, player_color, depth, alpha, beta)
- def minimax (self, board, player_color)
- def heuristic (self, move)
- def gameover (self, board)

Public Attributes

- **color**
- **max_depth**

2.6.1 Detailed Description

```
Documentation for the minimax with alpha-beta pruning AI player.

This class is the implementation of the minimax algorithm with alpha-beta pruning
for improved speed and greater depth search.
This class uses heuristic 1 (table).
```

2.6.2 Constructor & Destructor Documentation

2.6.2.1 def TabuleiroOthello-fork.models.players.table_minimax_alpha_beta_player.Minimax1Player.__init__ (*self*, *color*)

```
Constructor.
```

2.6.3 Member Function Documentation

2.6.3.1 def TabuleiroOthello-fork.models.players.table_minimax_alpha_beta_player.Minimax1Player.board_value (*self*, *board*, *player_color*)

```
This function returns the value of the board received as argument.
It is the sum of all positions that player_color has minus the sum of
positions that the opposition has. Blank positions do not count.
```

2.6.3.2 def TabuleiroOthello-fork.models.players.table_minimax_alpha_beta_player.Minimax1Player.gameover (*self*, *board*)

```
This function returns +1 if white player wins, -1 if black player wins,
0 if game is not over and +2 if game is a tie
```

2.6.3.3 def TabuleiroOthello-fork.models.players.table_minimax_alpha_beta_player.Minimax1Player.heuristic (*self*, *move*)

```
This function returns the value of the move, according to the heuristic table below:
100,   0,   6,   5,   5,   6,   0, 100
  0,   0,   8,   3,   3,   8,   0,   0
  3,   7,   3,   2,   2,   3,   7,   3
  4,   3,   2,   1,   1,   2,   3,   4
  4,   3,   2,   1,   1,   2,   3,   4
  3,   7,   3,   2,   2,   3,   7,   3
  0,   0,   8,   3,   3,   8,   0,   0
100,   0,   6,   5,   5,   6,   0, 100
```

2.6.3.4 def TabuleiroOthello-fork.models.players.table_minimax_alpha_beta_player.Minimax1Player.max_move (*self, board, player_color, depth, alpha, beta*)

```
This is the max part of the alpha-beta pruning. It plays every valid
move available for player_color (this player) until the game is over, if a move ends the game
or the depth is reached. It calls the min part of the alpha-beta algorithm.
During the recursive part, the return value (integer) of this function is the current best value.
If depth is 0, meaning the end of the recursive part, it returns the best move (Move object)
```

2.6.3.5 def TabuleiroOthello-fork.models.players.table_minimax_alpha_beta_player.Minimax1Player.min_move (*self, board, player_color, depth, alpha, beta*)

```
This is the min part of the alpha-beta pruning. It plays every valid
move available for player_color (oppenent) until the game is over, if a move ends the game
or the depth is reached. It calls the max part of the alpha-beta algorithm.
Return value (integer) of this function is the current best value (beta).
Very similar with the max function.
```

2.6.3.6 def TabuleiroOthello-fork.models.players.table_minimax_alpha_beta_player.Minimax1Player.minimax (*self, board, player_color*)

```
Wrapper function to call max and get the best move.
```

2.6.3.7 def TabuleiroOthello-fork.models.players.table_minimax_alpha_beta_player.Minimax1Player.play (*self, board*)

```
This is the function called by the board controller. Returns a Move.
```

The documentation for this class was generated from the following file:

- /home/ambrozio/Development/Atom/IA/TabuleiroOthello-fork/models/players/table_minimax_alpha_beta_↩ player.py

2.7 TabuleiroOthello-fork.models.players.random_player.RandomPlayer Class Reference

Public Member Functions

- def **__init__** (self, color)
- def **play** (self, board)

Public Attributes

- **color**

The documentation for this class was generated from the following file:

- /home/ambrozio/Development/Atom/IA/TabuleiroOthello-fork/models/players/random_player.py

Index