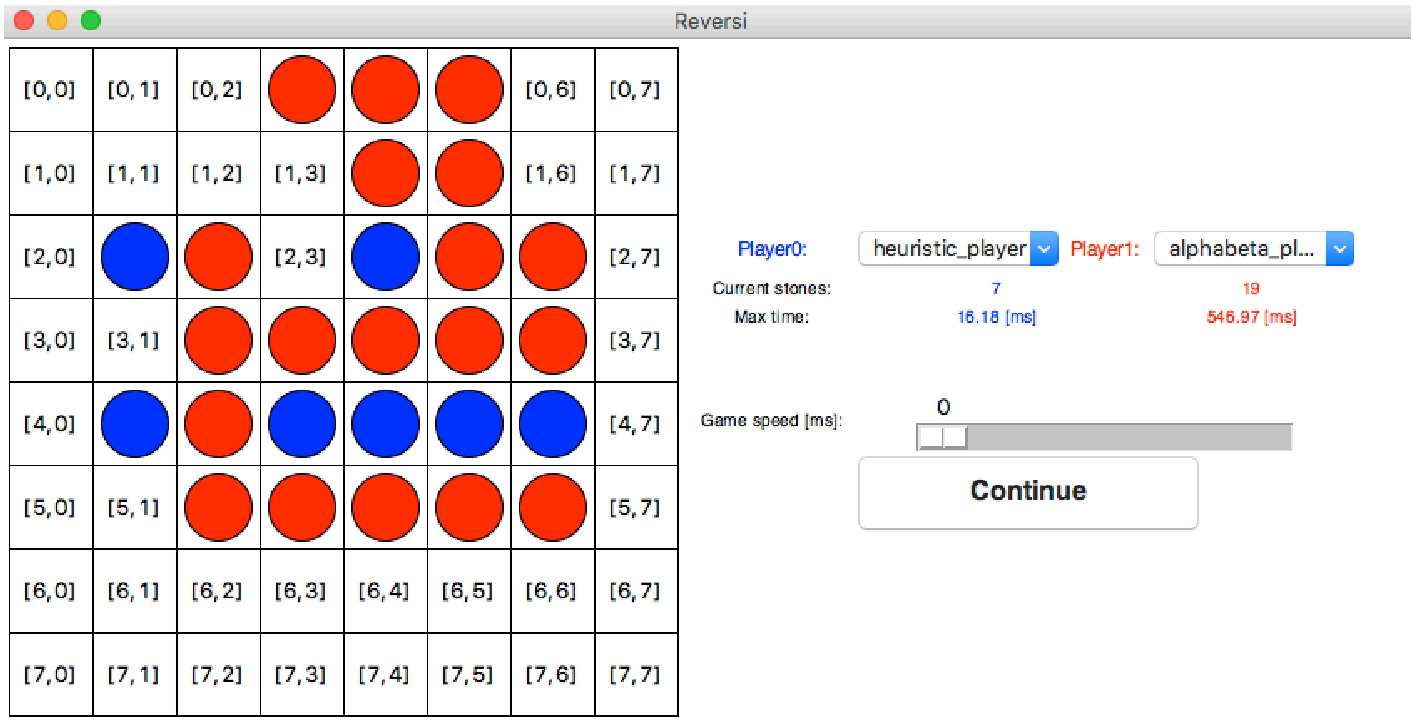
2. Reversi game

Rules: https://en.wikipedia.org/wiki/Reversi [https://en.wikipedia.org/wiki/Reversi]



Player0: heuristic_player Player1: alphabeta_pl...

	Player0	Player1
Current stones:	7	19
Max time:	16.18 [ms]	546.97 [ms]

Game speed [ms]: 0

[/wiki/_detail/courses/be5b33kui/labs/reversi/reversi-game.png?id=courses%3Abe5b33kui%3Asemtasks%3A02_reversi%3Astart]

Task

Your task is to ~~download the ZIP file~~ and alter the downloaded `player.py` file for the Reversi game and then play a Reversi tournament against the players of the rest of the class.

Learning outcomes

This task will teach you about:

- Adversarial search
  - Combinatorial explosion
  - Search algorithm effectivity

The solution can be generalized to other classical games, although in some cases this strategy does not lead to optimal solutions and you would use different approaches (e.g.: Go [https://en.wikipedia.org/wiki/Computer_Go]).

ZIP file content

See `README_EN.txt` in reversi.zip [/wiki/_media/courses/b3b33kui/cviceni/reversi/reversi_2022l.zip]).

The package contains also the file `game_board.py` that contains the class of the gameboard. You can also test your player in a Reversi game using the `reversi_creator.py` or `headless_reversi_creator.py` scripts.

Player specification

The file `player.py` contains the `MyPlayer` class with a few methods. You can use the method that is already implemented in the `MPlayer` class called `get_all_valid_moves` in order to get all the valid moves and thus focus only on your strategy and decision how to play the game.

- The code must be **Python 3** compatible, otherwise it can fail during automatic evaluation!

method	input	output	explanation
`__init__`	`my_color` , `opponent_color` , `board_size`	`None`	Creates a player and its opponent. The board size is in the constructor - you can use it for any heuristics calculation.
`get_all_valid_moves`	`board`	list of coordinate tuples	Method returns for a given board all the valid moves.
`move`	`board` (n x n game board)	r , c (row, column) a tuple - coordinates of your move	The input is 2D `list` - current game board. Method should return a valid move. Example: (0,0) means putting your stone to the position `board[r][c]` . If no valid move is possible return `None` . `board` values -1 for empty space and 0/1 for the stone color. The max time spent within the `move` method is 5 secs. **This is the method you are supposed to alter.**

MyPlayer properties

- docstring should be informative enough
- **attribute name contains the username of the student**

Example of MyPlayer implementation

```
class MyPlayer:
    '''super-smart indeed'''
    def __init__(self, my_color,opponent_color):
        self.name = 'username' #username of the studen

    def move(self,board):
        return (*,*)
```

Evaluation

Late submissions do not take part in the tournament and receive 0 points.

Points are given in the following way:

1. The player must work and generate correct moves (e.g.: not outside the board).
2. Manual evaluation is done by the teachers with regards to your code quality.
3. The automatic evaluation gives points based on automatic code testing and the rank of the player in the tournament.

Evaluation	min	max	note
Algorithm quality	0	1	Evaluated based on automatic evaluation whether the player follows the Reversi game rules.
Code quality	0	4	Comments, code structure, cleanliness, proper naming... We will take the complexity of the algorithm used into account as well - e.g. minimax with alpha-beta pruning vs. heuristics as a look-up table only.
Rank in the tournament	0	6	The tournament truly pits your code against your classmates'. Based on the ranking in the tournament, we distinguish for each sub-tournament (based on the board size) five levels and give them their respected point count: 1-2-3-4-6. Your final score is the maximum of the points gained in the subtournaments. If your player does not finish the tournament because of its mistake, you get 0 points.

Submission

Upload a ZIP archive with the module `player.py` and any other possible modules you created to the Upload system [https://cw.felk.cvut.cz/upload/]. **All the files must be in the archive's root folder! The archive cannot contain any other folders!**

Information about the tournament:

- you play against all the other players on boards of the size $n \times n$, $n \in \{6, 8, 10\}$ in three separate subtournaments
- every match consists of two games - first, player "A" starts, then player "B" starts
- there are 2 points for winning the match, 1 for a draw, 0 for losing both games
- 5 seconds limit per move - try it out in Brute
- illegal moves results in losing a game
- exceeding the time limit means losing the game
- creating the player (constructor) takes at most 60s
- don't use multithreading

Results

TBA

courses/be5b33kui/semtasks/02_reversi/start.txt · Last modified: 2024/02/20 10:27 by xposik