PyNeutron

Filip Lew

Jan 04, 2020

CONTENTS:

1 src **1**

1.1 gui module 1
1.2 main module 1
1.3 neutron module 1
1.4 player module 3
1.5 util module 4

Python Module Index **7**

Index **9**

CHAPTER

ONE

SRC

1.1 gui module

class gui.**GameWindow**(*board*, **args*, ***kwargs*)
Bases: PyQt5.QtWidgets.QWidget

class gui.**GridCell**(*pos*, **args*, ***kwargs*)
Bases: PyQt5.QtWidgets.QWidget

paintEvent(*self*, *QPaintEvent*)

1.2 main module

User Guide

1.3 neutron module

class neutron.**Neutron**(*board*, *position*)
Bases: *neutron.Soldier*

A special case of a *Soldier*.

A Neutron is different from a Soldier only by having a unique color value.

VALUE = 1

class neutron.**NeutronBoard**
Bases: object

The Neutron game board.

It is represented by a 5x5 NumPy array. The purpose of this class is to manage the array, ensuring it doesn't get into an invalid state, and to provide useful functions for the game's logic.

grid
the array containing raw data of this board.

Type numpy.array

white_soldiers
list of Soldier objects representing white soldiers.

Type list

black_soldiers
list of Soldier objects representing black soldiers.

Type list

furthest_empty_spot(*pos*, *dir*)
Get the furthest empty position one can get by moving in direction `dir` from position `pos` without colliding with anything.

Implemented as a while loop checking following conditions:

- if the position after a step in the given direction is still in the board's bounds,
- if the position after the step is empty.

While those conditions are met, the step is performed, adding direction to position. If, after executing the loop, the resulting position is different from the starting position, we return it. Else, the move could not be made, and we return `None`.

Parameters

- **pos** (`util.Vec`) – starting position.
- **dir** (*str or* `util.Vec`) – direction in which to move.

Returns position of the furthest empty spot in the line of sight of source position, or `None` if the movement cannot be made.

Return type *util.Vec*

get_soldiers(*color*)
Get all soldiers of a given color present on the board.

Parameters **color** (*int*) – color of the soldiers.

Returns a list of Soldier objects containing all soldiers of a given color.

Return type list

Raises **ValueError** – if the color given is not a valid soldier color

neighbors(*pos*)
Get values of board cells neighboring cell with position `pos`.

It iterates over coordinates from x-1 to x+1 and y-1 to y+1, making sure they are not out of the bounds of the board, and appends values at those positions to the resulting list. The source position itself is not included.

Parameters **pos** (`util.Vec`) – position of the cell.

Returns list of neighboring cells' values, without the source cell

Return type list

class neutron.**NeutronGame**(*first_player*, *second_player*)
Bases: `object`

The main Neutron game class.

Parameters

- **first_player** (`player.Player`) – the player who will start the game
- **second_player** (`player.Player`) – the second player

check_won()
Checks if the game was won, updating self.winner variable with the color of the winning player.

Returns winning player's color

Return type int

play_round()
Plays one round, swapping players afterwards.

start()
Starts the game, playing rounds until the game is won by either of the players.

class neutron.**Soldier**(*board*, *position*, *color*)
Bases: object

Class representing a soldier on the board.

Its main task is to enforce proper movement rules, to prevent the board from getting into an invalid state from the point of view of the game's rules.

Parameters

- **board** (neutron.NeutronBoard) – home board of this Soldier.
- **position** (util.Vec) – position of this Soldier on the board.
- **color** (*int*) – color of this Soldier.

move(*direction*)
Tries to move this Soldier in the given direction.

For this method to succeed, the direction given must be present in self.possible_directions.

Works by calling *NeutronBoard.furthest_empty_spot()*, setting the position returned by this function to this Soldier's color, and the original position to 0.

Parameters **direction** (*str*) – direction in which to move this Soldier.

Raises **ValueError** – if the given direction is not in self.possible_directions.

property possible_directions
List of directions this Soldier can move

Works by checking for which directions *NeutronBoard.furthest_empty_spot()*

property possible_moves
List of positions this Soldier can be after one move.

Works by supplying *NeutronBoard.furthest_empty_spot()* with all possible directions, then filtering out None results.

1.4 player module

class player.**HumanPlayer**(*color*)
Bases: *player.Player*

move_neutron(*board*)
Method called by the game when it's this player's turn to move the neutron.

Parameters

- **board** (neutron.NeutronBoard) – board of the game played by this
- **player.** –

move_soldier(*board*)
Method called by the game when it's this player's turn to move one of their soldiers.

Parameters

- **board** (neutron.NeutronBoard) – board of the game played by this
- **player.** –

class player.**Player**(*color*)
Bases: abc.ABC

Abstract base class of all Neutron game players. Defines methods called by the game to allow players to make decisions about the next move.

Parameters **color** (*int*) – color of this player's soldiers.

abstract move_neutron(*board*)
Method called by the game when it's this player's turn to move the neutron.

Parameters

- **board** (neutron.NeutronBoard) – board of the game played by this
- **player.** –

abstract move_soldier(*board*)
Method called by the game when it's this player's turn to move one of their soldiers.

Parameters

- **board** (neutron.NeutronBoard) – board of the game played by this
- **player.** –

class player.**RandomPlayer**(*color*)
Bases: *player.Player*

move_neutron(*board*)
Method called by the game when it's this player's turn to move the neutron.

Parameters

- **board** (neutron.NeutronBoard) – board of the game played by this
- **player.** –

move_soldier(*board*)
Method called by the game when it's this player's turn to move one of their soldiers.

Parameters

- **board** (neutron.NeutronBoard) – board of the game played by this
- **player.** –

class player.**StrategyPlayer**(*color*)
Bases: *player.Player*

1.5 util module

class util.**Color**
Bases: object

BLACK = 3

WHITE = 2

color_names = {2: 'white', 3: 'black'}

class util.**Vec**(*x*, *y*)
Bases: object

A very simple implementation of a 2D vector, used to facilitate operations on positions and directions.

Parameters

- **x** (*int*) – vector's x coordinate.
- **y** (*int*) – vector's y coordinate.

static fromtuple(*tuple_pos*)
Creates a *Vec* from tuple (y, x). This order of coordinates was chosen to be compatible with NumPy's way of indexing multidimensional arrays.

Parameters **tuple_pos** (*tuple*) – an (y, x) tuple representing vector's coordinates.

Returns a newly created Vec.

Return type *Vec*

x

y

PYTHON MODULE INDEX

g

gui, 1

m

main, 1

n

neutron, 1

p

player, 3

u

util, 4

INDEX

B

BLACK (*util.Color attribute*), 4
black_soldiers (*neutron.NeutronBoard attribute*), 1

C

check_won() (*neutron.NeutronGame method*), 2
Color (*class in util*), 4
color_names (*util.Color attribute*), 5

F

fromtuple() (*util.Vec static method*), 5
furthest_empty_spot() (*neutron.NeutronBoard method*), 2

G

GameWindow (*class in gui*), 1
get_soldiers() (*neutron.NeutronBoard method*), 2
grid (*neutron.NeutronBoard attribute*), 1
GridCell (*class in gui*), 1
gui (*module*), 1

H

HumanPlayer (*class in player*), 3

M

main (*module*), 1
move() (*neutron.Soldier method*), 3
move_neutron() (*player.HumanPlayer method*), 3
move_neutron() (*player.Player method*), 4
move_neutron() (*player.RandomPlayer method*), 4
move_soldier() (*player.HumanPlayer method*), 3
move_soldier() (*player.Player method*), 4
move_soldier() (*player.RandomPlayer method*), 4

N

neighbors() (*neutron.NeutronBoard method*), 2
Neutron (*class in neutron*), 1
neutron (*module*), 1
NeutronBoard (*class in neutron*), 1
NeutronGame (*class in neutron*), 2

P

paintEvent() (*gui.GridCell method*), 1
play_round() (*neutron.NeutronGame method*), 3
Player (*class in player*), 4
player (*module*), 3
possible_directions() (*neutron.Soldier property*), 3
possible_moves() (*neutron.Soldier property*), 3

R

RandomPlayer (*class in player*), 4

S

Soldier (*class in neutron*), 3
start() (*neutron.NeutronGame method*), 3
StrategyPlayer (*class in player*), 4

U

util (*module*), 4

V

VALUE (*neutron.Neutron attribute*), 1
Vec (*class in util*), 5

W

WHITE (*util.Color attribute*), 4
white_soldiers (*neutron.NeutronBoard attribute*), 1

X

x (*util.Vec attribute*), 5

Y

y (*util.Vec attribute*), 5