QuantumTicTacToe

Łukasz Szarejko

February 10, 2021

Task description

This was my project on my first year of studies on Warsaw University of Technology. There is description of the task:
You have to create a program which allows play **quantum tictactoe** between human and computer player. You have to create at least two versions of bot:
1. which chooses random move in each turn
2. which chooses the best move according to simple rules.

Solution description

My program has two game options. First one, in terminal, after command **python3 -m quantum_tictactoe.game**. Second one, with GUI. To play in GUI, player must write command **python3 -m quantum_tictactoe.main**. In both type of game, player can choose to play with other human on the same devices or with computer player. Player constantly gaining informations about what he should now do or what he have done wrong.
These are simple rules of my version of the game:
1. Tile in entanglement contains **'*'**
2. Collapsed tile contains three the same moves, thanks to it, player knows which state it takes on
3. when on board is only one tile which is not collapsed and there is no winner, game ends with draw
4. other rules match with rules on Wikipedia website.

Game in terminal

After proper command in terminal, player have to choose type of the game. There are three modes **none/easy/hard**. **None** is game without bot, **easy** with computer which chooses random moves, **hard** with bot which has some rules. After choosing type, game starts and continues to win of one player or draw. Standard moves are handle by writing in terminal tile number where player wants to make move. *Tiles are numbered from 0 to 8.* Choose during entanglement by writing tile number, comma and move to collapse. For example: 2,x3

GUI Game

After proper command in terminal, on your screen will appear window titled **QuantumTicTacToe**. To start game, player has to choose mode form menu bar. If player repeats it, game will clear itself and start from the beginning. Available modes are the same as in the terminal version of the tictactoe.

Hard mode of computer player

Because game is so complicated and have to many possible moves, **hard** bot doesn't choose the best moves but according to this simple rules:
1. while choosing second tile to move, computer tries to pick empty tile if there is any, if not, it goes to second rule
2. in this case, it tries to make move on tile where are only its moves, if there in no any of that tile, it choose move randomly.

Architecture of the solution

Tile This is major class of the program. It contains information about each tile: moves on this tile, whether tile is in entanglement or is collapsed. Class has methods to make move, change state to entangle or collapse.

Board Contains all tiles in game and list of collapsed tiles numbers. Has methods to show board in terminal, change tile state to collapsed, searching board to check if there is a winner.

Game Contains board and a lot of variables about game state, which are helpful to control: if game is in entanglement, which round it is, which was the last move etc. Has methods to check if move is correct, clear game, who should make move now, make move, check if game went to entangle state or game ends. In this class is also method responsible for process of game in terminal.

Bot Contains info about game type, board and game. Has methods to make move by bot or choose move and tile to collapse, if player made entanglement.

Form This is class generated automatically by **Qt designer**.

Main This class is responsible for game by GUI. It inherits after class **Form**. Its constructor calls parent constructor and connects button clicked actions with other methods of this class. Contains methods responsible for tile click, change tile to entangle state, bot move, show buttons of moves possible to collapse, make move, refresh tiles contents, chose game type, clear game.

Tests

Classes and methods, which are not responsible for GUI were tested by unit tests. Moreover, game was tested just by playing.