Charles University Faculty of Mathematics and Physics

Programming 1 [NPRG030]

# Tic-tac-toe game

Mozharov Nazar

Prague 2022

# Content

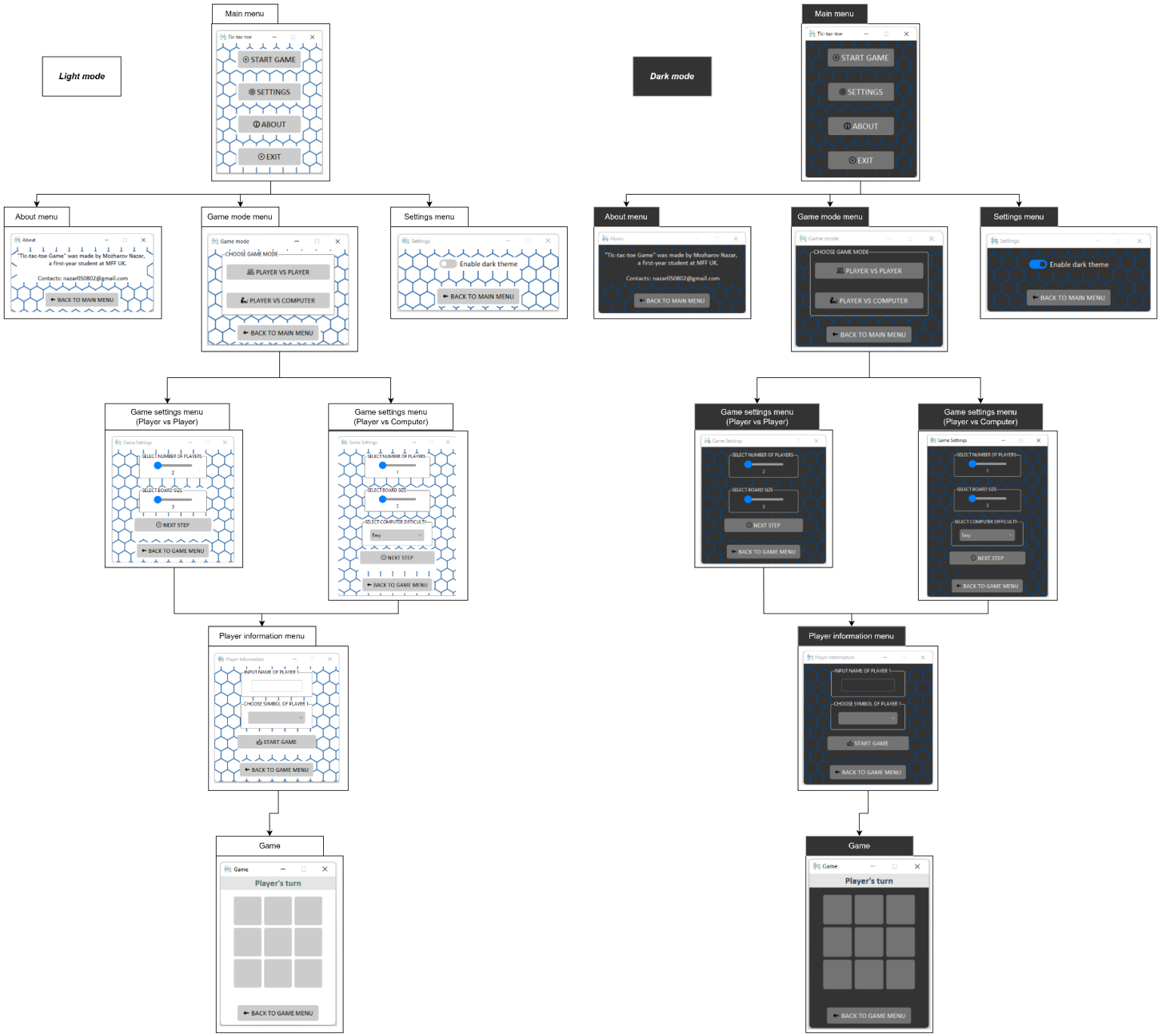
## General information about the program.

The purpose of the task is to create a game of tic-tac-toe. The game must meet the following requirements:

- Create a game with Graphic User Interface (GUI)
- Implement two game modes:
    - Game mode player versus player
    - Game mode player versus computer
- Create three levels of computer difficulty, which will include various algorithms for each level.
- Implement different conditions for victory depending on the size of the playing board.
- Implement the ability to change the number of players and the size of the playing board.

# Describing the Graphic User Interface (GUI)

I implemented the graphic user interface using the Tkinter library, which is built into the Python programming language. To change the general appearance of the program, I used third-party themes for the Tkinter library. Moreover, I implement a setting to change the application theme from light to dark and vice versa.




*Game GUI structure*

## Describing user actions

When starting the program, the user enters the main menu. At this point, the player can open the settings and change the theme color of the application. Besides this, in the main menu, the player can open window about the author.

When the player presses the start game button, the he enters the game mode selection menu: player vs. player or player vs. computer.

After selecting the game mode, the player sets the game parameters, such as the size of the board, the number of players, the difficulty of the computer. Then player have to enter game data such as the player's name and player’s game symbol. After then player can start the game.

When starting the game, there are two options for the rules:

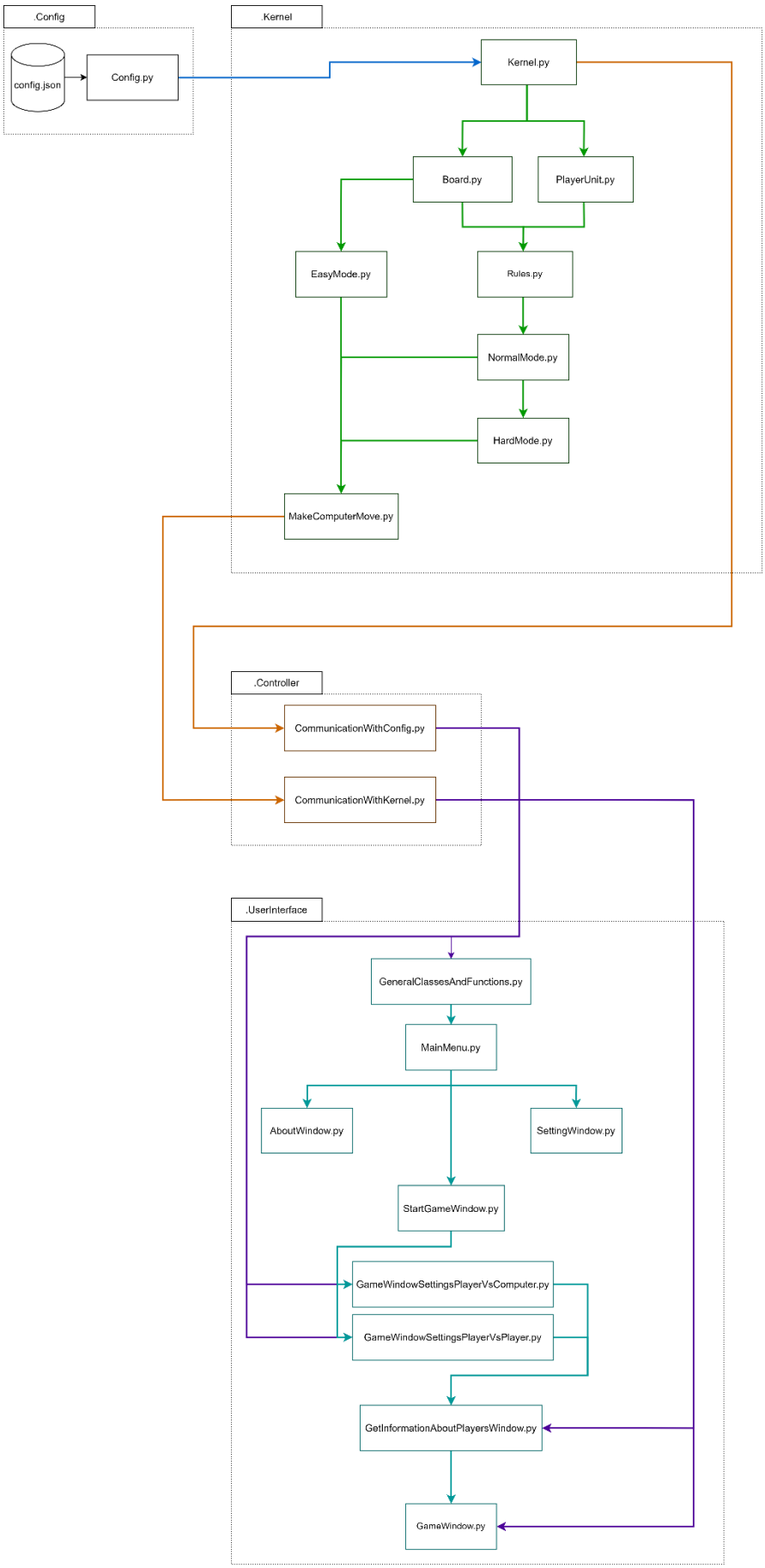
- If the playing field is less than 6x6, then to win players need to put the same number of symbols horizontally or vertically or diagonally. Their number is equal to the size of the current board.
- If the playing board is larger than 6x6, then players need to collect at least 5 identical symbols horizontally or vertically or diagonally.

# Describing the technical part

## About the game code

To implement the game, I divided code into 3 parts: the kernel, the controller, and the GUI:

- The kernel is responsible for processing and analyzing the playing field, in addition to this, the kernel also interacts with the config file.
- The controller is responsible for the communication between the GUI and the program kernel. In this way, the controller converts data, sends requests, requests information from the config file.
- The GUI is responsible for rendering various menus for user interaction.




*Picture of dependencies between core, controller and GUI*

## About different levels of computer difficulty

The computer has three difficulty levels: easy, normal and hard. Depending on the level, the computer will behave differently and use different tactics and techniques to win.

### Difficulty easy

On the easy difficulty level, the computer makes moves in random order.

### Difficulty normal

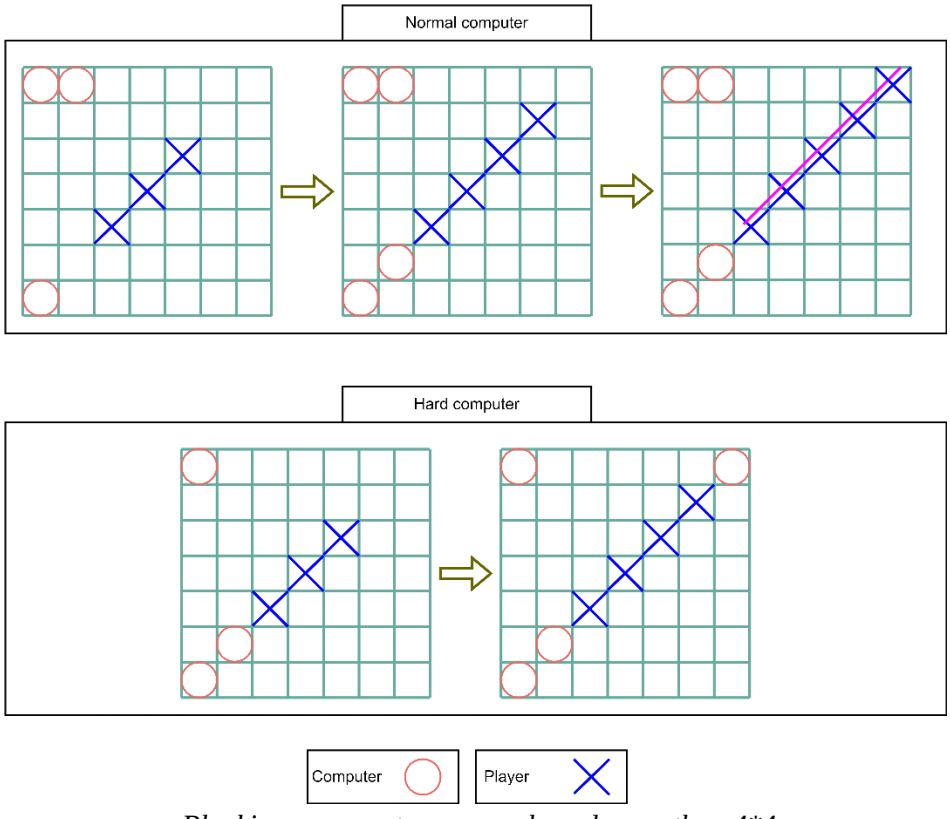
On the normal difficulty level, the computer makes moves according to the following rules:

1. If the computer's next move can be a win for it, then the computer moves to win.
2. If the opponent's next move could be a win for him, then the computer blocks him.
3. In any other cases, move randomly

### Difficulty hard

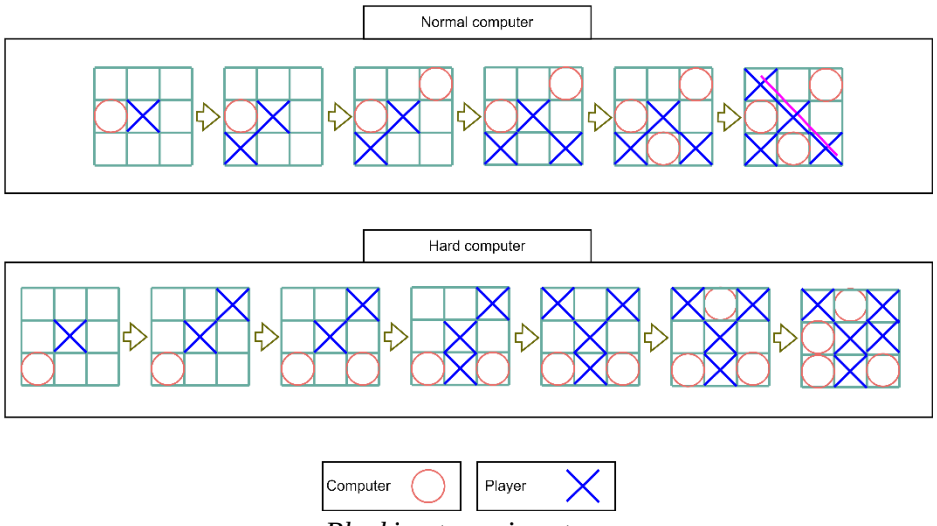
On the hard difficulty level, the computer makes moves according to the following rules:

1. If the computer's next move can be a win for it, then the computer moves to win.
2. If the opponent's next move could be a win for him, then the computer blocks him.
3. Track if the player wants to use one of the tactics to win on the playing board more than 4x4, then block it.




*Blocking opponent moves on board more than 4*4*

4. If the center positions are empty, then computer make move in the center.
5. Block situations where there are two possible moves to win at once.




*Blocking two wins at once*

6. Find where the computer made a move last time and, if possible, make a move nearby.
7. In any other cases, move randomly

## Conclusion about the created game

During the process of creating the game, I learned how to interact better with classes in Python. In addition to it, I learned how to create a user interface using the Tkinter library.

In the future, I would like to recreate the algorithm for the hard difficulty of the computer, because it is extremely slow on large playing fields and sometimes loses to the player. I would like to use better Minimax algorithm, but with a limitation on the recursion depth for large fields.

As for the GUI, I would like to add the ability to save the game. Besides this, I would like to implement the ability for the player to choose their own color, as well as new symbols for the players.

And at the end I also would like to implement the statistics of the played games.