OOP Project

Chess in C++

Jacek Młynarczyk 151747

What have you created?

My project is Object oriented Chess Game with a command line display.

Examples of some classes will be shown on next slides:

class of a piece

```
#include <vector>
#include "common.hpp"
#include "board.hpp"

class Piece {
private:
    PieceColor my_color;

public:
    Piece(PieceColor color);
    virtual PieceColor Color() const;

    virtual std::vector<std::pair<int, int>> possible_moves(std::pair<int, int> my_possition, Board current_board) const = 0;
    virtual PieceId id() const = 0;
    virtual char letter_symbol() const = 0;

    virtual ~Piece() = default;
};
```

Example of class for pieces

```
4   #include "Piece.hpp"
5
6   class King : public Piece {
7   public:
8       King(PieceColor color);
9
10      std::vector<std::pair<int, int>> possible_moves(std::pair<int, int> my_possition, Board currnet_board) const override;
11      PieceId id() const override;
12      char letter_symbol() const override;
13
14  };
```

class Board

```
7   #include "Piece.hpp"
8
9   class Board {
10  private:
11      std::vector<std::vector<Piece*>> board;
12      bool can_en_passant;
13      std::pair<int, int> en_passant_position;
14  public:
15      Board();
16      void print_board();
17      void move_piece(std::pair<int, int> from, std::pair<int, int> to);
18      void add_piece(Piece* piece, std::pair<int, int> position);
19      void remove_piece(std::pair<int, int> position);
20      Piece* piece_at(std::pair<int, int> position);
21      void initalize_start_position();
22      void mark_possible_moves(std::pair<int, int> position);
23      void clear_board();
24      bool en_passant() const;
25      std::pair<int, int> en_passant_square() const;
26  };
```

class GameRunner

```
4   #include "Board.hpp"
5   #include "common.hpp"
6
7   class GameRunner final {
8       Board* current_board;
9       PieceColor on_move;
10  public:
11      GameRunner();
12      void restart();
13      void start_from_position(Board board);
14
15      MoveResult make_move(std::pair<int, int> from, std::pair<int, int> to);
16      void pass_move();
17      void print_currnet_board();
18      void possible_moves_for_piece(std::pair<int, int> position);
19      PieceColor color_on_move() const;
20
21      ~GameRunner();
22  };
```

Initializing the game

```
9   void game_loop(){
10      cout << "Welcome to CliChess!" << endl;
11      GameRunner gameRunner;
12      gameRunner.restart();
13      gameRunner.print_currnet_board();
14
15
16      int from_x, from_y, to_x, to_y;
17      while (true){
18          cout << "Enter move: \n";
19          cout << "From: ";
20          cin >> from_x >> from_y;
21          gameRunner.possible_moves_for_piece({from_x, from_y});
22          cout << "\nTo: ";
23          cin >> to_x >> to_y;
24
25          auto res = gameRunner.make_move({from_x, from_y}, {to_x, to_y});
26          if (res != MoveResult::MoveMade) {
27              cout << "Illegal move" << endl;
28          }
29
30          gameRunner.print_currnet_board();
31      }
32  }
```

Did you have any problems?

Although object themselves appeared not to be a major problem, implementing all parts together to be functional ended up being the thing I spent the most time on.

In the end I had to give up on applying check and checkmate, even though extremely important, but it was nothing crucial in making it object oriented and it’s just a technicality.

Did you learn anything new?

I learned how to work with objects in bigger projects, since chess turned out to be more complex than I expected it to be. I obviously gained some experience in C++, like using makefile. Something that I found helpful was how use CoPilot.

I also realised that I have to start my projects faster, so it’s not necessary to stay all night finishing work.

What could be improved?

- Implementing check and checkmate to make project funcional, as well as adding more functions like castling and promotion
- Just to make it more visible, I could work on the colour of the pieces displayed in the console
- I could add some exceptions to my code in the future