Project by Nikolay Zdravkov (F105268) and Teodor Ivanov (F105440**)**

Chess Game in Pygame: Documentation - Part 1

Introduction

This document presents the first part of the documentation for a chess game developed using Pygame, a popular set of Python modules designed for writing video games. The project aims to create an interactive chess game that users can play using a graphical interface. This part of the documentation covers the initial setup of the game, the rendering of the chessboard, and the display of turn information.

Pygame Initialization and Constants

Initializing Pygame

Pygame is initialized at the beginning of the script:

```
import pygame

pygame.init()
```

This initialization is crucial as it sets up the Pygame library for use.

Constants

Several constants are defined to standardize the game's appearance and behavior:

WIDTH, HEIGHT: The dimensions of the game window.

ROWS, COLS: The number of rows and columns on the chessboard.

SQUARE_SIZE: The size of each square on the chessboard, calculated based on the window width and the number of columns.

These constants ensure consistency in the game's visual elements and provide a reference for positioning and scaling.

Loading Images

The load_images function is responsible for loading the images for each chess piece:

```
def load_images():
    # Function details...
```

This function creates a dictionary of images, with keys representing different pieces (e.g., 'w_king', 'b_queen') and values being the corresponding Pygame image objects. It ensures that each piece is visually represented on the board.

Board Rendering

Drawing the Chessboard

The **draw_board** function is used to draw the chessboard:

```
def draw_board(win, check_position=None):
    # Function details...
```

This function alternates the color of the squares to create the classic chessboard pattern. If a king is in check (**check_position** is not **None**), it highlights the king's position in red.

Displaying Turn or Game Over Status

The **display_turn** function displays whose turn it is, or "Game Over" if the game has ended:

```
def display_turn(win, turn, game_over):
    # Function details...
```

It provides real-time feedback to the players, enhancing the interactivity of the game.

Chess Game in Pygame: Documentation - Part 2

Introduction

This document presents the second part of the documentation for our chess game developed using Pygame. Building upon the foundational aspects covered in the first part, this section delves into the gameplay mechanics, focusing on how pieces are moved and how the game responds to player actions.

Piece Movement and Game Logic

Selecting and Moving Pieces

The process of selecting and moving chess pieces is central to the gameplay. This is handled through a series of functions:

- **get_square_from_pos(pos)**: Converts a pixel position on the screen to a corresponding row and column on the chessboard.
- **get_piece_at_pos(pos, board_layout)**: Returns the piece located at the given position on the board.

- **move_piece(selected_piece, start_pos, end_pos, board_layout)**: Moves the selected piece from the start position to the end position if the move is legal.

Legal Move Validation

The legality of each move is verified using the **is_legal_move** function:

```
def is_legal_move(start_pos, end_pos, piece, board_layout):
    # Function details...
```

This function checks whether a proposed move adheres to the rules of chess, including the unique movement patterns of different pieces and restrictions like not being able to move through other pieces.

Checking for Check and Game Over

Identifying Check

The game continuously checks if either king is in a state of 'check' – threatened to be captured on the next move:

- **get_king_position(board_layout, color)**: Locates the position of the king for the specified color.
- **is_in_check(king_position, board_layout, opponent_color_prefix)**: Determines if the king at the given position is in check.

Game Over Condition

The game ends when one of the kings is captured. This is determined by the **is_game_over** function:

```
def is_game_over(board_layout):
    # Function details...
```

If either king is no longer on the board, the function returns **True**, signaling the end of the game.

Main Game Loop and Event Handling

The heart of the game is the main loop, where Pygame's event handling system processes player inputs and updates the game state:

```
def main():
    # Main game loop details...
```

Key aspects of this loop include:

- Processing events like mouse clicks and window closing.
- Managing the turn-taking mechanism and updating the game state.
- Drawing the updated game board and pieces after each move.
- Checking for and handling the end-of-game scenario.