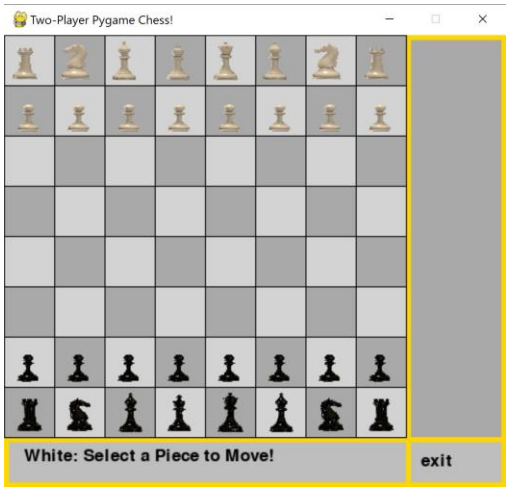
Two-Player Pygame Chess Game Report

1. Introduction

This report covers the design, development, and implementation of a two-player chess game created using the Pygame library in Python. The game allows two players to compete against each other on the same device, taking turns to move pieces according to the standard rules of chess. The game includes features such as move validation, check detection, and game-over conditions.



2. Game Overview

The game is a graphical representation of chess, where two players (White and Black) alternate turns to move pieces on an 8x8 board. The game starts with the standard initial positions of chess pieces, and players can select and move pieces using the mouse. The game ends when one player's king is captured, indicating a checkmate, or when a player decides to exit the game.

3. Design and Implementation

3.1 User Interface

The game window is set to a size of 500x450 pixels. The chessboard occupies the top portion of the screen, while the bottom portion displays game status information and an "exit" button.

- Board: The chessboard consists of 8x8 squares, alternating between light gray and dark gray colors to differentiate between the squares.

- Pieces: Each piece type (king, queen, rook, bishop, knight, pawn) is represented by an image, which is loaded and scaled to fit the board squares.

- Captured Pieces: Captured pieces are displayed on the right side of the screen, with White's captured pieces on the top and Black's on the bottom.

3.2 Game Logic

The game logic is divided into several key components:

- Piece Movement: Each piece type has its own function to check valid moves based on its position and the current state of the board. The game ensures that moves are valid according to chess rules (e.g., pawns can only move forward, knights move in an "L" shape, etc.).

- Turn Management: The game keeps track of whose turn it is (White or Black) and whether a piece has been selected. The turn alternates after each valid move.

- Check Detection: The game checks if a king is in check after each move. If a king is in check, the square it occupies flashes, alerting the player.

- Move Validation: The game validates moves to ensure they are legal according to chess rules. This includes ensuring that moves do not place the player's own king in check.

- Game Over Conditions: The game ends when a king is captured, signifying a checkmate. The game also allows players to restart the game or exit by clicking the "exit" button.

4. Functions and Methods

The game logic is implemented through several key functions:

- `*draw_board()*`: Draws the chessboard and UI elements.

```
# Draw game board
def draw_board():
    for i in range(32):
        col = i % 4
        row = i // 4
        if row % 2 == 0:
            pygame.draw.rect(screen, 'light gray', [300 - (col * 100), row * 50, 50, 50]) ## even row
        else:
            pygame.draw.rect(screen, 'light gray', [350 - (col * 100), row * 50, 50, 50]) ## odd row

    pygame.draw.rect(screen, 'gray', [0, 400, WIDTH, 100]) ## color for lower board
    pygame.draw.rect(screen, 'gold', [0, 400, WIDTH, 50], 5) ## gold color for fram of board
    pygame.draw.rect(screen, 'gold', [400, 0, 100, HEIGHT], 5)
    status_text = ['White: Select a Piece to Move!', 'White: Select a Destination!',
                   'Black: Select a Piece to Move!', 'Black: Select a Destination!']
    screen.blit(big_font.render(status_text[turn_step], True, 'black'), (20, 410))
    screen.blit(big_font.render('exit', True, 'black'), (415, 415)) ## eixt buttom
    for i in range(9): ## black line in board
            pygame.draw.line(screen, 'black', (0, 50 * i), (400, 50 * i), 1)
            pygame.draw.line(screen, 'black', (50 * i, 0), (50 * i, 400), 1)
```

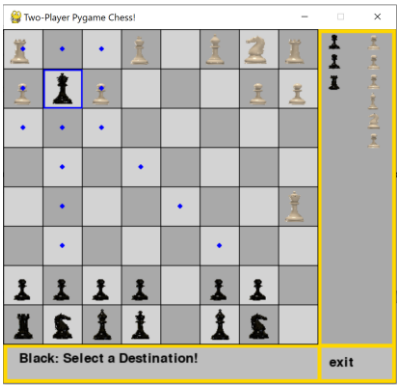
- `*draw_pieces()*`: Places the pieces on the board according to their current positions.

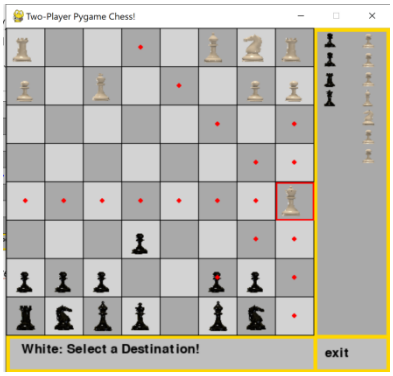
```
def draw_pieces():
    for i in range(len(white_pieces)):
        index=piece_list.index(white_pieces[i])
        if white_pieces[i] == 'pawn': ## if it pawn
            screen.blit(white_pawn, (white_locations[i][0] * 50 + 11, white_locations[i][1] * 50 + 15))
        else: # other pieces
            screen.blit(white_images[index], (white_locations[i][0] * 50+ 5, white_locations[i][1] * 50 + 5))
        if turn_step < 2: ## if it turn of white
            if selection == i: ## if the pieces is select make it in red fram
                pygame.draw.rect(screen, 'red', [white_locations[i][0] * 50 + 1, white_locations[i][1] * 50 + 1,50, 50], 2)

    for i in range(len(black_pieces)):
        index = piece_list.index(black_pieces[i])
        if black_pieces[i] == 'pawn':## if it pawn
            screen.blit(black_pawn, (black_locations[i][0] * 50 + 11, black_locations[i][1] * 50 + 15))
        else:# other pieces
            screen.blit(black_images[index], (black_locations[i][0] * 50 + 10, black_locations[i][1] * 50 + 5))
        if turn_step >= 2:## if it turn of black
            if selection == i:## if the pieces is select make it in red fram
                pygame.draw.rect(screen, 'blue', [black_locations[i][0] * 50 + 1, black_locations[i][1] * 50 + 1,50, 50], 2)
```

- `*check_options()*`: Determines all valid moves for a player's pieces.

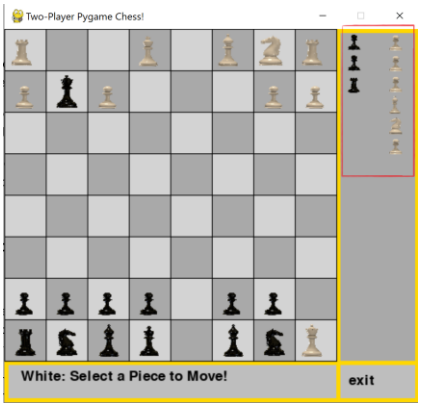
- `*check_pawn()*`, `*check_rook()*`, `*check_knight()*`, `*check_bishop()*`, `*check_queen()*`, `*check_king()*`: Check the valid moves for each piece type.

- `*draw_valid()*` :Highlights valid moves for a selected piece.





- `*draw_captured()*` : Displays captured pieces on the side of the board.



- `*draw_check()*` : Highlights the king if it is in check.

- `*draw_game_over()*` : Displays the game-over message when a player wins.

5. Game Flow

1. Initialization: The game initializes the Pygame environment, sets up the display window, and loads images for the pieces.

2. Main Loop: The game enters a loop that continues until the player quits. During each iteration of the loop:

- The game checks for user input (mouse clicks, key presses).
- The board and pieces are redrawn.
- Valid moves are highlighted if a piece is selected.
- The game checks for a win condition or if the king is in check.

3. Turn Handling: Players take turns selecting and moving pieces. The game ensures that only valid moves are made.

4. Game Over: If a king is captured, the game displays a "Game Over" message and allows the players to restart or exit.

6. Conclusion

The two-player chess game implemented using Pygame provides a functional and enjoyable experience for users. The game successfully simulates standard chess rules and includes features such as move validation, turn management, and check detection. The game can be further enhanced by adding features such as AI opponents, move undo functionality, and multiplayer support over a network.