Halama Part0

Youssef Elmahdy

6398550

Instructor: Tayyaba Shaheen

CS 470 - Artificial Intelligence

Northern Arizona University (NAU)

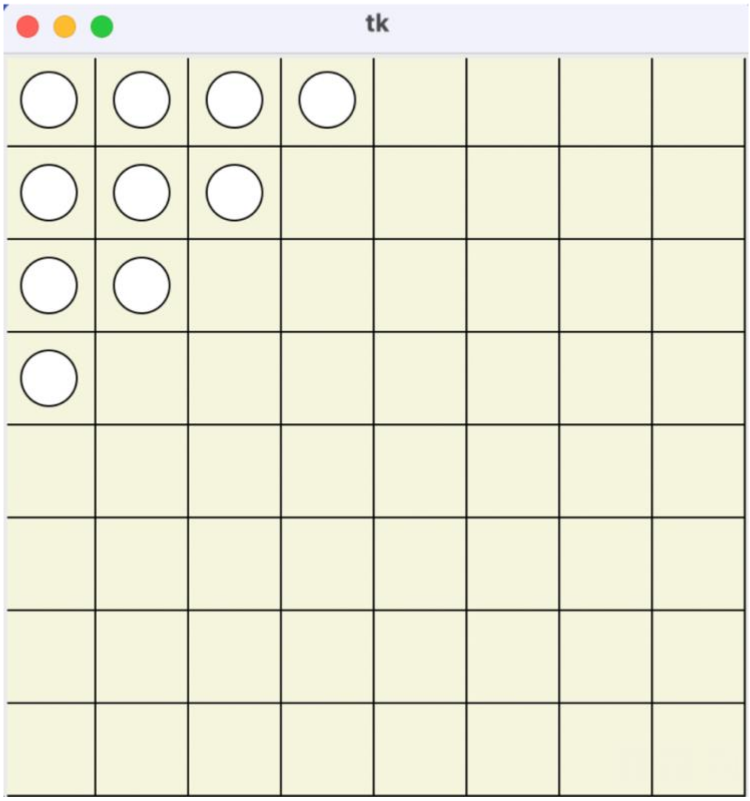
1) Overview:

The HalmaBoard class provides a structure for a simple Halma game board using the tkinter library for GUI. The class has been designed with modularity in mind, encapsulating functionalities like board creation, piece placement, move highlighting, and piece movement. Below is an overview of the key objects and methods:

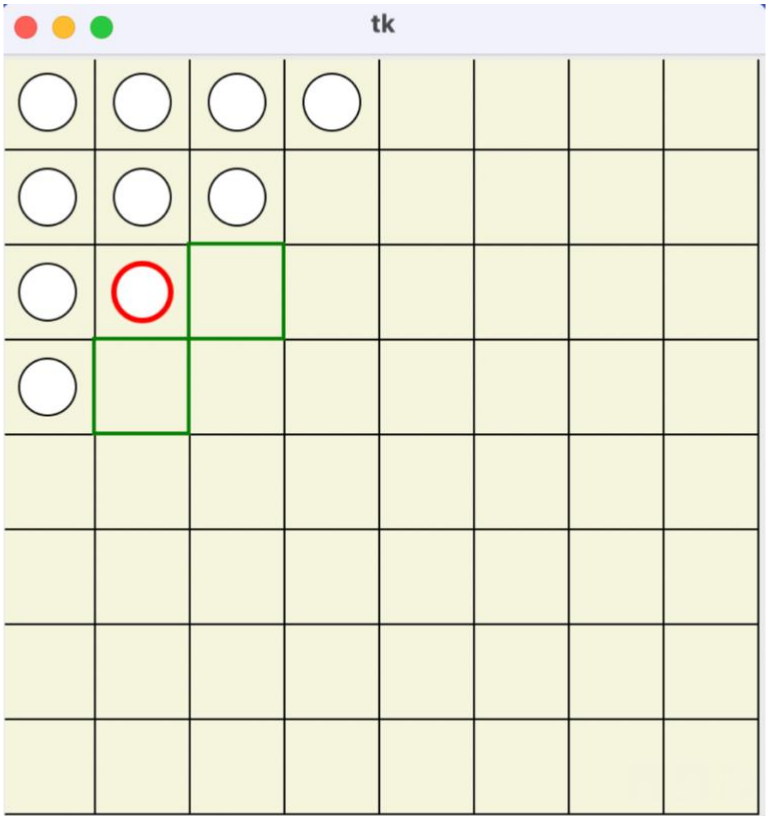
- **Initialization (__init__)**
    - Sets board size and canvas, creates a grid (create_grid), and initializes pieces (initialize_pieces). Manages piece locations (self.pieces) and tracks valid moves.
- **Grid and Piece Setup**
    - **create_grid**: Draws an 8x8 beige grid.
    - **place_piece** & **initialize_pieces**: Position initial game pieces and store them in self.pieces.
- **Move Highlighting (highlight_moves & clear_highlights)**
    - Highlights valid moves for a selected piece, ensuring they're within bounds and unoccupied. Clears highlights when a new piece is selected.
- **Piece Movement (on_click & move_piece)**
    - on_click handles selecting pieces or moves; move_piece updates a piece's position, managing its coordinates and data storage.

This structure organizes board creation, piece management, and interactions, making the code easy to extend or modify.

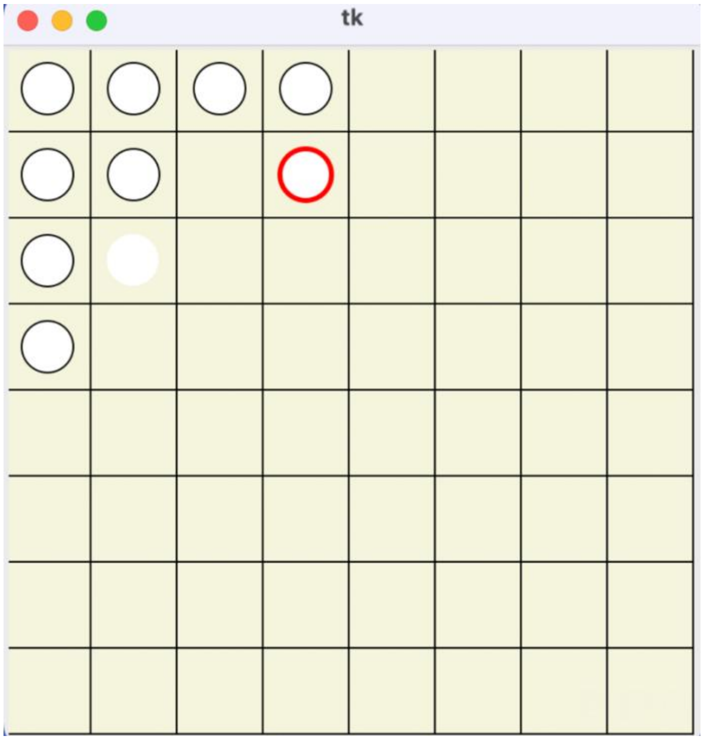
2) Screenshot of initial board: white or black circles in one corner, representing the pieces for one player



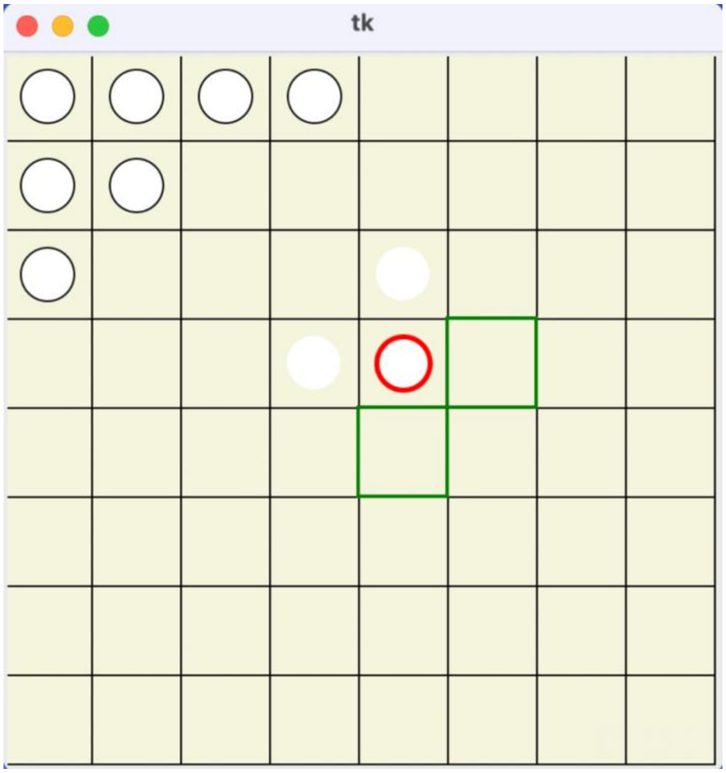
3) Screenshots of possible moves after clicking on a piece. Clicking on a circle should hilight that piece, and make squares appear, representing where moves are possible for that piece.



4) Screenshots after having moved some pieces. Clicking on a square should move the hilighted piece



5) Screenshot of moves available from several consecutive jumps. Start by implementing moves to adjacent squares. After that is working, implement the jumping.



6) Code:

```
import tkinter as tk


class HalmaBoard:
    def __init__(self, root, size=8):
        self.size = size
        self.cell_size = 50
        self.canvas = tk.Canvas(root, width=self.size * self.cell_size, height=self.size *
self.cell_size)
        self.canvas.pack()

        self.create_grid()

        self.pieces = {}
        self.selected_piece = None
        self.valid_moves = []

        self.initialize_pieces()

    def create_grid(self):
        for row in range(self.size):
            for col in range(self.size):
                x1 = col * self.cell_size
                y1 = row * self.cell_size
                x2 = x1 + self.cell_size
                y2 = y1 + self.cell_size
                self.canvas.create_rectangle(x1, y1, x2, y2, fill='beige')

    def place_piece(self, row, col, color):
        x1 = col * self.cell_size + 10
        y1 = row * self.cell_size + 10
        x2 = x1 + self.cell_size - 20
        y2 = y1 + self.cell_size - 20
        piece = self.canvas.create_oval(x1, y1, x2, y2, fill=color)
        self.pieces[(row, col)] = piece

    def initialize_pieces(self):
        initial_positions = [
            (0, 0), (0, 1), (0, 2), (0, 3),
            (1, 0), (1, 1), (1, 2),
            (2, 0), (2, 1),
            (3, 0)
        ]
        for row, col in initial_positions:
            self.place_piece(row, col, 'white')

    def highlight_moves(self, row, col):
        self.clear_highlights()

        possible_moves = [(row - 1, col), (row + 1, col), (row, col - 1), (row, col + 1)]
        valid_moves = []

        for r, c in possible_moves:
            if 0 <= r < self.size and 0 <= c < self.size and (
            r, c) not in self.pieces:
                x1 = c * self.cell_size
                y1 = r * self.cell_size
                x2 = x1 + self.cell_size
                y2 = y1 + self.cell_size
                move_id = self.canvas.create_rectangle(x1, y1, x2, y2, outline='green',
width=2)
                valid_moves.append((r, c, move_id))

        self.valid_moves = valid_moves

    def clear_highlights(self):
        for move in self.valid_moves:
            self.canvas.delete(move[2])
        self.valid_moves = []

    def on_click(self, event):
        row, col = event.y // self.cell_size, event.x // self.cell_size

        if (row, col) in self.pieces:
            if self.selected_piece:
                self.canvas.itemconfig(self.selected_piece, outline="")

            self.selected_piece = self.pieces[(row, col)]  # Select the new piece
            self.canvas.itemconfig(self.selected_piece, outline="red", width=3)
            self.highlight_moves(row, col)

        # If a valid move is clicked, move the piece
        else:
            for move in self.valid_moves:
                if (row, col) == (move[0], move[1]):
                    self.move_piece(self.selected_piece, (move[0], move[1]))
                    break

    def move_piece(self, piece_id, to_pos):
        from_pos = [pos for pos, p_id in self.pieces.items() if p_id == piece_id][0]
        row, col = to_pos
        x1 = col * self.cell_size + 10
        y1 = row * self.cell_size + 10
        x2 = x1 + self.cell_size - 20
        y2 = y1 + self.cell_size - 20

        self.canvas.coords(piece_id, x1, y1, x2, y2)

        del self.pieces[from_pos]
        self.pieces[to_pos] = piece_id

        self.clear_highlights()


root = tk.Tk()
game_board = HalmaBoard(root)

game_board.canvas.bind("<Button-1>", game_board.on_click)

root.mainloop()
```