24127183 - Trần Duy Khang

Lab HomeWorks: Wordle Project

Course: Computational Thinking

I/ System Design:

The project follows a **Modular Design** separating **Game Logic** and **GUI Logic**:

- Source
    - main.py → game logic (backend)
    - wordle_core.py → GUI logic (frontend)
    - allowed.txt → allowed guesses
    - secret.txt → secret word list
- Report.pdf → this file
- README.md

File Descriptions:

File	**Description**
main.py	Implements the Tkinter GUI interface, user input handling, feedback coloring, and new game setup.
wordle_core.py	Defines the **WordleGame** class that handles word checking, color logic, and game state.
allowed.txt / secret.txt	Contain lists of valid English words
README.md	Explains how to install and run the game locally.

II/ Graphical User Interface (Tkinter)

The game is implemented using the **Tkinter** library — Python's built-in GUI framework.

Interface Elements:

- **Title Label:** "WORDLE GAME" at the top.
- **5×6 Grid:** Each square displays a guessed letter.
- **Menu Bar:** Allows player to start a new game or exit.

- **Message Label:** Displays feedback (win/lose/error).
- **Keyboard Input:** Player types letters directly via the keyboard.

Key Functions Explained:

1) `show_message(text, color="black", duration=2000)`

Purpose:
Displays a temporary message below the game board (e.g. "❌ Word not allowed!" or "🎉 YOU WIN!").

How it works:

- Updates the text and color of the message_label.

- If the message is not a win/loss, it automatically disappears after the specified duration (using root.after()).

Concepts used:

- Tkinter Label widget for display.

- Event scheduling via after() for timed message clearing.

2) `colorize_row(row, feedback)`

Purpose:
Colors each of the 5 cells in the given row according to the feedback from WordleGame.check_guess().

Logic:

🟩 or "G" → Green (#6aaa64)

🟨 or "Y" → Yellow (#c9b458)

⬜ or "_" → Gray (#787c7e)

Concepts used:
Dynamic GUI updates by calling .config(bg=..., fg=...) on Tkinter labels.

3) `start_new_game()`

Purpose:
Resets all game states and starts a new Wordle session.

Steps performed:

1. Creates a new WordleGame object (from wordle_core.py).
2. Randomly selects a secret word from secrets.
3. Clears the letter grid and message label.
4. Rebinds keyboard input events.

Concepts used:

- OOP: class instantiation (WordleGame).
- GUI state management (resetting widgets).
- Randomization (random.choice()).

4) `on_key_press(event)`

Purpose:
Handles user keyboard input in real time.

Behavior:

- Letter keys (A–Z): add letters to current guess.
- Backspace: delete the last letter.
- Enter: submit the guess and get color feedback.

Validation rules:

- Reject repeated guesses.
- Reject words not found in allowed.txt.

- End the game if the player wins or uses all 6 attempts.

Concepts used:

- Event-driven programming (root.bind("<Key>", ...))
- String handling and validation
- Conditional control flow for gameplay logic

5) GUI Setup (Tkinter)

Elements used:

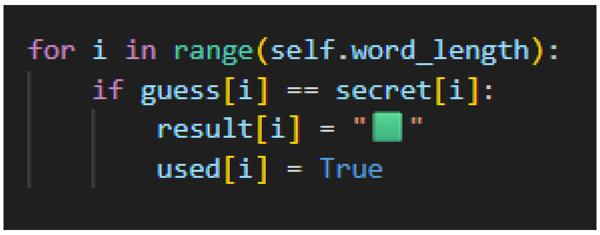
- Tk() main window → root
- Frame → container for the grid
- Label → each letter cell
- Menu → Reset / Exit options
- bind() → capture keyboard events

The GUI uses a 6×5 grid layout with dynamic color updates and centered messages.

III/ Core Algorithm - Check Guess Logic

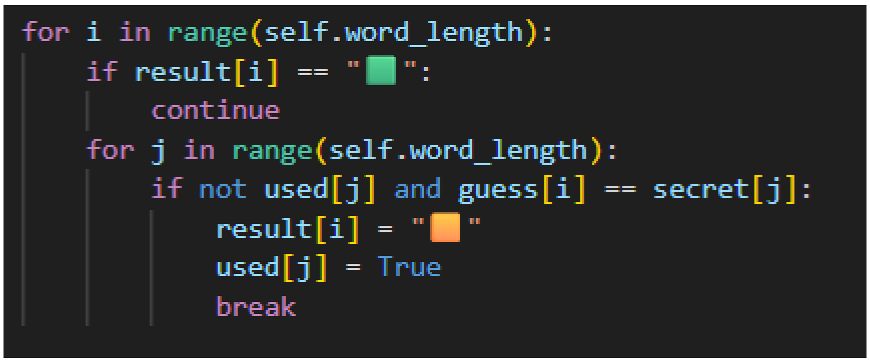
The function check_guess() in wordle_core.py performs a two-pass validation algorithm:

- 1st pass: Mark correct positions (green)



```
for i in range(self.word_length):
    if guess[i] == secret[i]:
        result[i] = "🟩"
        used[i] = True
```

- 2nd pass: Mark correct letters in wrong position (yellow)



```
for i in range(self.word_length):
    if result[i] == "🟩":
        continue
    for j in range(self.word_length):
        if not used[j] and guess[i] == secret[j]:
            result[i] = "🟧"
            used[j] = True
            break
```

IV/ Demo Clip

🎬 **24127183_Demo_WordleGame.mp4**