VIETNAM NATIONAL UNIVERSITY, HO CHI MINH CITY
UNIVERSITY OF SCIENCE
FALCULTY OF INFORMATION TECHNOLOGY

COURSE: COMPUTATIONAL THINKING

HOMEWORK REPORT
Wordle Game

Lab Teacher	Mr.	Huynh Lam Hai Dang
Lab Teacher	Mr.	Nguyen Thanh Tinh
Students:	24127171	Huynh Thai Hoang

Contents

1 Overview

This report illustrates the design and implementation of a desktop Wordle game created using Python. The project is a fully functional clone of the popular web-based game, developed as a standalone application with a graphical user interface (GUI).

1.1 What is Wordle?

Wordle is a word-guessing puzzle game. The objective is to guess a secret five-letter word within six attempts. After each guess, the game provides color-coded feedback for each letter:

- **Green:** The letter is correct and in the correct position.
- **Yellow:** The letter is in the word but in the wrong position.
- **Gray:** The letter is not in the word at all.

This feedback allows the player to hint the player about the secret word. Our project recreates this logic in a desktop application.

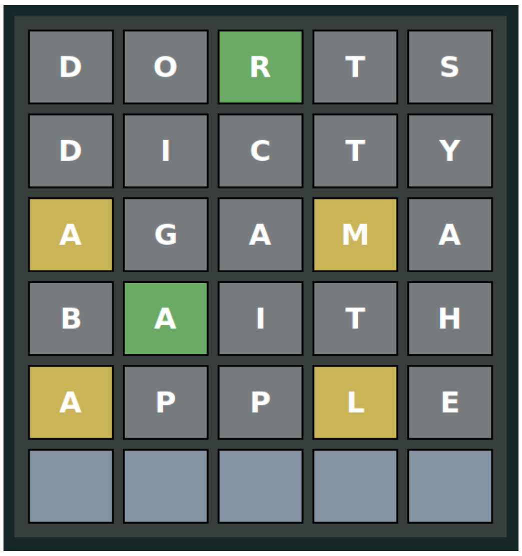


Figure 1: the main Wordle grids

1.2 Technologies Used

The project was built exclusively with Python 3 and its standard libraries. Especially, no external packages are required!.

- **Python 3:** The core programming language used for all game logic.
- **Tkinter:** A Python's GUI library which used to create the application window, the 6x5 game grid, the on-screen keyboard, and to handle all user interactions.
- **Random:** A standard library module used to randomly select the secret word from the word list.

1.3 Source Code and Demonstration

- **Github Repository (Source code):** Link
- **Demonstration Video:** Link

2 Details of the Game

2.1 Game Flow

The game operates on a simple, turn-based event loop, as described below.

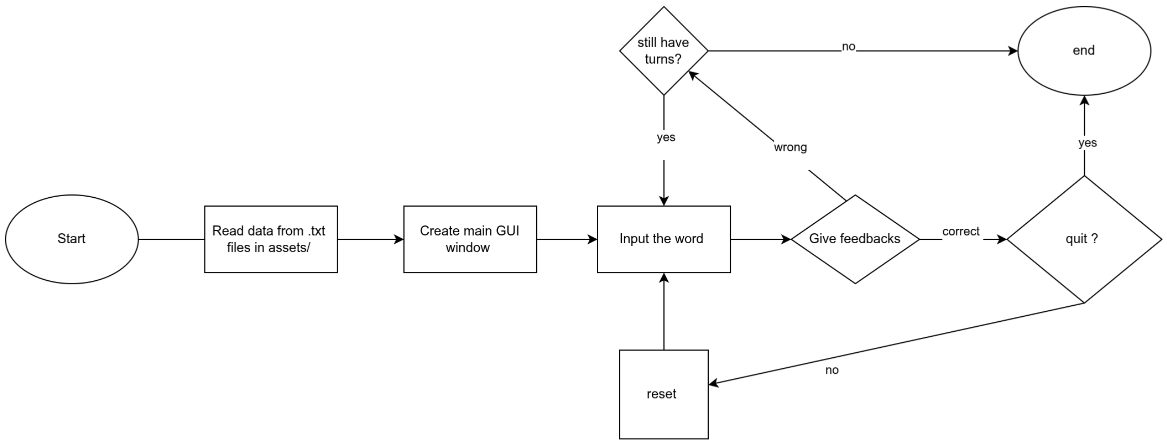


Figure 2: Game Flow

2.2 Feature Implementation

2.2.1 The main GUI class

The `WordleGUI` class holds the game's current state using instance attributes (variables prefixed with `self.`).

- `self.master`: Stores a reference to the main `tk.Tk()` root window, which acts as the primary container for all other widgets.
- `self.secret_word`: A string that stores the hidden word for the current game session. This is initialized once in the constructor.
- `self.current_turn`: An integer that tracks the current guess row (starting from `0`). This is crucial for determining which row of labels on the grid to update after a guess.
- `self.grid_labels`: A 2D list (a list of lists). This data structure directly mirrors the visual grid. It holds all the `tk.Label` widgets, allowing for direct access to any specific cell using coordinates (e.S., `self.grid_labels[row][col]`).

2.2.2 Word Handling and Validation

The game relies on two external text files located in an `assets/` directory:

- `dictionary.txt`: A list of all valid 5-letter words. This is used to validate player guesses.

- `guesses.txt`: A smaller list of words that can be chosen as the secret word that the player have to guess.

The `load_words` function reads these files, cleans the data, and stores them in Python sets for efficient lookups. Here is the source code of how it stores after reading these files:

```
# MAKE A GUESSING WORDS AND A DICTIONARY
WORDS = load_words('assets/dictionary.txt')
GUESSSES = load_words('assets/guesses.txt')

# Ensure guesses are in words list
WORDS |= GUESSSES

# Converting to list
final_list = list(WORDS)
guess_list = list(GUESSSES)
```

2.2.3 Core Feedback Logic

The most critical part of the game is the `provide_feedback` function. It correctly handles duplicate letters (e.g., guessing "APPLE" when the word is "PAPER").

The logic works in two passes:

1. **Green Pass:** The function first iterates through the guess and checks for exact matches (correct letter in the correct position). These are marked as 'GREEN'. A count of the letters in the secret word is maintained.
2. **Yellow/Gray Pass:** The function iterates a second time. It skips any 'GREEN' letters. It then checks if a non-green letter is present anywhere else in the secret word and if its count is greater than zero. If yes, it's marked 'YELLOW', and the count is decremented. Otherwise, it's marked 'GRAY'.

The source code for this function is as follows:

```
def provide_feedback(self, guess, secret):
    feedback = [""] * WORD_LENGTH
    cnt_secret_letter = {}

    for letter in secret:
        cnt_secret_letter[letter] = cnt_secret_letter.get(letter,0) + 1

    for i in range(WORD_LENGTH):
        if (guess[i] == secret[i]):
            feedback[i] = 'GREEN'
            cnt_secret_letter[guess[i]]-=1

    for i in range(WORD_LENGTH):
        letter = guess[i]
        if feedback[i] == 'GREEN':
            continue
        if (letter in cnt_secret_letter and cnt_secret_letter[letter] > 0):
            feedback[i] = 'YELLOW'
            cnt_secret_letter[letter] -= 1
        else:
            feedback[i] = 'GRAY'

    return feedback
```

2.2.4 Input Management

The game handles two forms of user input:

- **Physical Keyboard:** The main window binds to the `<KeyPress>` event, which calls the `handle_key_press` method. This method processes letter keys, the 'Enter' key, and the 'Backspace' key.
- **Virtual Keyboard:** The on-screen keyboard is a grid of `tk.Button` widgets. Each button's `command` is linked to the `handle_keyboard_click` method, passing its specific letter.

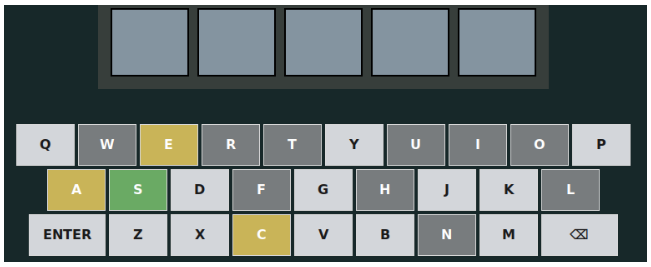
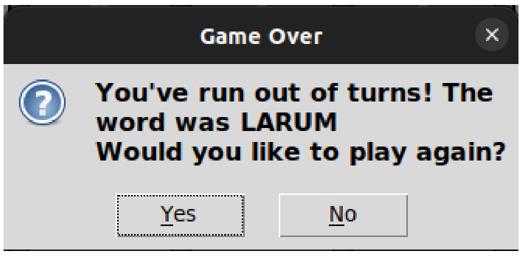


Figure 3: Virtual keyboard

2.2.5 Game State Management

The game state is managed by instance variables like `self.current_turn` and `self.current_guess`. After a game ends, the `messagebox.askyesno` function asks the user to play again or not. If the user clicks "Yes," the `reset_game` method is called. This method clears all text and colors from the grid, resets the keyboard colors, resets the `current_turn` to 0, and selects a new secret word, making the game ready for a new session.


Game Over

You've run out of turns! The word was LARUM
Would you like to play again?

Figure 4: Game over window

References

[1] GeeksforGeeks. *Python | GUI - tkinter*. https://www.geeksforgeeks.org/python/python-gui-tkinter/. Accessed: 2025-10-23.

[2] GeeksforGeeks. *Python Program for Word Guessing Game*. https://www.geeksforgeeks.org/python/python-program-for-word-guessing-game/. Accessed: 2025-10-23.