Wordle Game Documentation

This documentation explains the **Wordle Game** written in C using the **Raylib** library. The game supports three modes: Classic, Endless, and Time Trial. Below is a breakdown of the program's functions, their purpose, and how they operate.

Constants

- WORD_LENGTH: Length of each word (5 characters).
- MAX_GUESSES: Maximum number of guesses allowed (6).
- CELL_SIZE: Size of each cell in the Wordle grid.
- MAX_WORDS: Maximum number of words that can be loaded from the dictionary.
- SCREEN_WIDTH and SCREEN_HEIGHT: Dimensions of the game window.
- TIME_TRIAL_LIMIT: Time limit in seconds for the Time Trial mode.

Enumerations

GameState

Defines the current state of the game:

- MENU: Main menu screen.
- CLASSIC_MODE: Classic mode gameplay.
- ENDLESS_MODE: Endless mode gameplay.
- TIME_TRIAL_MODE: Time Trial mode gameplay.

Functions

1. void loadWords(const char *filename, char words[MAX_WORDS][WORD_LENGTH + 1], int *wordCount)

- **Purpose**: Loads words from a file into the words array.
- **Parameters**:
    - filename: Path to the file containing words.

- words: 2D array to store loaded words.
- wordCount: Pointer to an integer to store the count of loaded words.

- **How it works**:
  - Opens the file in read mode.
  - Reads words (up to 5 characters) line by line and stores them in the words array.
  - Increments the wordCount until the maximum limit is reached.

---

2. char* chooseRandomWord(const char words[MAX_WORDS][WORD_LENGTH + 1], int wordCount)

- **Purpose**: Selects a random word from the words array.
- **Parameters**:
  - words: 2D array of words.
  - wordCount: Total number of loaded words.
- **How it works**:
  - Seeds the random number generator using time(NULL).
  - Picks a random index from 0 to wordCount - 1 and returns the word at that index.

---

3. int validateWord(const char *guess, const char words[][WORD_LENGTH + 1], int wordCount)

- **Purpose**: Validates whether a guessed word exists in the dictionary.
- **Parameters**:
  - guess: The guessed word.
  - words: 2D array of dictionary words.
  - wordCount: Total number of loaded words.
- **Return Value**: 1 if the word is valid, otherwise 0.

---

4. int checkGuess(const char *guess, const char *targetWord, char feedback[WORD_LENGTH + 1])

- **Purpose**: Compares a guessed word with the target word and generates feedback.
- **Parameters**:
  - guess: The guessed word.
  - targetWord: The correct word to guess.
  - feedback: Output array indicating the result of each letter:
    - 'G': Correct letter in the correct position.
    - 'Y': Correct letter in the wrong position.
    - 'X': Incorrect letter.
- **Return Value**: 1 if the guess is correct, otherwise 0.

---

5. void displayWordleBoard(const char board[MAX_GUESSES][WORD_LENGTH + 1], const char feedback[MAX_GUESSES][WORD_LENGTH + 1], int currentRow)

- **Purpose**: Draws the Wordle grid with feedback for each guess.
- **Parameters**:
  - board: Array of guessed words.
  - feedback: Feedback for each word.
  - currentRow: Index of the current row in the grid.
- **How it works**:
  - Iterates over the grid and draws rectangles for each cell.
  - Colors the cells based on feedback (GREEN for 'G', YELLOW for 'Y', GRAY for 'X').
  - Draws letters in the cells.

---

6. void playClassicMode(const char words[MAX_WORDS][WORD_LENGTH + 1], int wordCount, GameState *state)

- **Purpose**: Implements the Classic mode gameplay.
- **Parameters**:
    - words: Array of words.
    - wordCount: Total number of loaded words.
    - state: Pointer to the current game state.
- **How it works**:
    - Chooses a random word.
    - Accepts input for guessing words, row by row.
    - Updates feedback based on correctness.
    - Ends the game when the word is guessed or attempts are exhausted.

---

7. void playEndlessMode(const char words[MAX_WORDS][WORD_LENGTH + 1], int wordCount, GameState *state)

- **Purpose**: Implements the Endless mode gameplay.
- **Parameters**: Same as playClassicMode.
- **How it works**:
    - Functions similarly to Classic mode but allows continuous play.
    - Resets the board and chooses a new word after each round.

---

8. void playTimeTrialMode(const char words[MAX_WORDS][WORD_LENGTH + 1], int wordCount, GameState *state)

- **Purpose**: Implements the Time Trial mode gameplay.
- **Parameters**: Same as playClassicMode.
- **How it works**:
    - Tracks the remaining time using a countdown.

- Ends the game if the timer reaches zero or the player guesses the word.

---

9. void drawTimeRemaining(int timeRemaining)

- **Purpose**: Draws the remaining time on the screen.
- **Parameters**:
  - timeRemaining: Time left in seconds.
- **How it works**:
  - Formats the time into a string and displays it using DrawText.

---

10. void showMenu(GameState *state)

- **Purpose**: Displays the main menu and handles mode selection.
- **Parameters**:
  - state: Pointer to the current game state.
- **How it works**:
  - Displays menu options.
  - Updates the game state based on user input (C, E, or T).

Program Flow

1. Initialize the Raylib window and load words from words.txt.
2. Start in the MENU state.
3. Based on user input, transition to one of the game modes.
4. Play the selected mode.
5. Return to the menu or exit when the window closes.