**Documentation of Maxim’s Hangman**

This program is a simple Hangman game implemented using Python and the Pygame library. The game randomly selects a word from a predefined list, and the player must guess the word letter by letter before the hangman is fully drawn.

**Features**

* Interactive graphical interface using Pygame
* Clickable letter buttons for user input
* A hint system that reveals the first letter of the word
* Reset functionality to restart the game
* Victory and defeat messages

**Things you need**

Install Pygame using: pip install pygame

**Code Breakdown**

**1. Setting Up the Display**

The game window is initialised with a width of 1300 and a height of 500 pixels using Pygame's display.set\_mode() function. The window title is "Hangman Game!".

**2. Creating the Alphabet Buttons**

* A list called letters stores details for 26 circular buttons, each representing a letter from A to Z.
* The buttons are positioned dynamically based on startx, starty, and an evenly spaced grid.

**3. Fonts and Colors**

* The game uses four fonts: LETTER\_FONT, WORD\_FONT, TITLE\_FONT, and HINT\_FONT.
* Defined colors include WHITE, BLACK, BLUE, and RED.

**4. Loading Images**

Seven images ("hangman0.png" to "hangman6.png") are loaded into a list called images to represent different stages of the hangman.

**5. Game Logic**

* A list of words is predefined, and one word is chosen randomly at the start.
* guessed keeps track of letters that the player has guessed.
* hangman\_status tracks the number of incorrect guesses.
* hint\_displayed determines if a hint should be shown.

**6. The Draw Function**

* Clears the screen and redraws the game state each frame.
* Displays the title, guessed word, remaining letters, reset button, hint button and hangman image
* If the hint is activated, it shows the first letter of the word.

**7. Reset Function**

Resets the game state, choosing a new word and making all letters selectable again.

**8. Display Message Function**

* Displays a message indicating whether the player won or lost and reveals the correct word.
* Pauses before resetting the game.

**9. Main Game Loop**

* Runs the game at 60 FPS using pygame.time.Clock().
* Listens for user input:
  + Clicking a letter removes it from the available choices and checks if it's in the word.
  + Clicking "RESET" restarts the game.
  + Clicking "Hint ?" reveals the first letter.
* Checks if the player has won or lost after each guess.

**Possible Improvements**

* Add more words and/or categories
* Improve the hint system to provide multiple hints.
* Add a score tracking system

**HANGMAN FLOWCHART DIAGRAM**

A screenshot of a computer screen

AI-generated content may be incorrect.

**Pseudocode**

Initialize Pygame

Set up screen dimensions and create the window

Load required assets (images, fonts, and sounds)

Define colors and font styles

Set up game variables:

- word\_to\_guess (choose a random word)

- guessed\_letters (empty list)

- wrong\_guesses (counter starting at 0)

- max\_wrong\_guesses (set a limit, e.g., 6)

- hint\_displayed (boolean, initially set to False)

- game\_state (initialize as "ongoing")

Define a function to display the current state of the game:

- Draw the hangman figure based on wrong\_guesses

- Display the guessed letters and blanks for remaining letters

- Display guessed letters already chosen

- If hint\_displayed is True, display the first letter of word\_to\_guess

Define a function to handle user input:

- Capture key events

- Check if the key is an alphabet character

- If guessed correctly, update guessed\_letters

- If guessed incorrectly, increment wrong\_guesses

Define a function for button handling:

- If "Hint" button is clicked:

- Set hint\_displayed to True

- If "Reset" button is clicked:

- Reset all game variables (guessed\_letters, wrong\_guesses, hint\_displayed, and choose a new word\_to\_guess)

Define the main game loop:

- While game\_state is "ongoing":

- Clear the screen

- Call the display function

- Call the input handling function

- Call the button handling function

- Check for win/loss conditions:

- If all letters in word\_to\_guess are guessed, set game\_state to "win"

- If wrong\_guesses exceeds max\_wrong\_guesses, set game\_state to "loss"

- Refresh the screen and regulate frame rate

Display win or loss messages:

- If win, display congratulatory message

- If loss, display the correct word

Wait for user to close the window or restart the game

Quit Pygame

**Time Journal**

|  |  |
| --- | --- |
| **Date** | **Task Completed** |
| **Feb 26** | Initial project setup, created Pygame window |
| **Feb 29** | Implemented alphabet buttons with click detection |
| **Mar 3** | Added word selection and guessing logic |
| **Mar 7** | Integrated hangman images for incorrect guesses |
| **Mar 10** | Implemented the draw function to update the screen |
| **Mar 14** | Added victory and defeat conditions |
| **Mar 18** | Created reset and hint functionalities |
| **Mar 22** | Final testing and debugging |
| **Mar 26** | Completed documentation and final refinements |