तमसो मा ज्योतिर्गमय

NIRMA
UNIVERSITY

INSTITUTE OF TECHNOLOGY

NAAC ACCREDITED ‘A+’ GRADE

Special Assignment:

Scripting Languages

WORDLE: A GUI based game using python

Submitted to

Dr. Vaishali H. Dhare

Submitted By

Aditya Lal (21BEC004)

Atit Nikhilgiri (21BEC014)

INTRODUCTION:

About Wordle:

Wordle is a web-based word game created and developed by Welsh software engineer Josh Wardle. Here are the general rules for a typical Wordle game:

Objective: The goal of Wordle is to guess a hidden five-letter word within six attempts.

Word Selection: The game randomly selects a five-letter word for each round.

Guessing:

- Players make guesses by proposing five-letter words.
- After each guess, the game provides feedback to indicate the correctness of each letter.
- Correctly guessed letters are typically highlighted in green.
- Correct letters in the wrong position are often marked with yellow.
- Incorrect letters are usually displayed in gray or another neutral color.

Limited Attempts: Players have a maximum of six attempts to guess the correct word.

Feedback:

- The feedback after each guess helps players refine their strategy for subsequent guesses.
- The game provides information on the correctness and position of guessed letters.

Game Completion:

- The game ends when the player correctly guesses the word within the allowed number of attempts.
- If the player exhausts all attempts without guessing the word, the game typically reveals the correct word.

About Pygame:

Pygame is a cross-platform set of Python modules designed for writing video games. It is built on top of the Simple DirectMedia Layer (SDL), a low-level multimedia library that provides access to audio, keyboard, mouse, and display functions. Pygame simplifies game development by offering a range of tools and utilities that handle common tasks, allowing developers to focus on the game logic and design.

Key features of Pygame include:

1. **Graphics and Drawing:** Pygame provides modules for handling graphics, allowing developers to create and manipulate images, draw shapes, and manage sprites easily. It supports various image formats, making it versatile for incorporating artwork into games.
2. **Event Handling:** Pygame simplifies the handling of user input events such as keyboard presses, mouse movements, and clicks. This makes it easier for developers to create interactive and responsive games.
3. **Game Development Utilities:** Pygame includes additional modules for handling time, random numbers, and other common game development tasks. These utilities help streamline the coding process and enhance the overall development experience.

4. **Community and Documentation:** Pygame has a supportive community of developers and a wealth of documentation and tutorials. This makes it accessible for both beginners and experienced developers looking to create 2D games in Python.

CODE:

```
import random
import pygame
```

- Importing the random and pygame libraries

```
def load_dict(file_name):
    with open(file_name) as file:
        return [line[:5].upper() for line in file.readlines()]

DICT_GUESSING = load_dict("SL/dictionary_english.txt")
DICT_ANSWERS = load_dict("SL/dictionary_wordle.txt")
ANSWER = random.choice(DICT_ANSWERS)
```

- `load_dict`: A function that loads a list of words from a given file, truncating each word to five characters and converting them to uppercase.
- `DICT_GUESSING`: List of words used for checking the validity of user guesses.
- `DICT_ANSWERS`: List of words from which a random word is chosen as the answer.
- `ANSWER`: The target word that the player needs to guess.

```
WIDTH, HEIGHT, MARGIN, T_MARGIN, B_MARGIN, LR_MARGIN = 500, 600, 10, 30, 50, 
50
GREY, GREEN, YELLOW, WHITE = (100, 100, 100), (80, 200, 70), (250, 200, 
100), (255, 255, 255)
```

- Constants defining the dimensions and margins for the game window.
- Color constants for the game.

```
pygame.init()
pygame.font.init()
pygame.display.set_caption("Wordle")
```

- Initialize the Pygame library, set the window caption to "Wordle," and initialize the font module

```
SQ_SIZE = (WIDTH - 4 * MARGIN - 2 * LR_MARGIN) // 5
FONT = pygame.font.SysFont('sansbold', SQ_SIZE)
screen = pygame.display.set_mode((WIDTH, HEIGHT))
INPUT = ""
GUESSES = []
ALPHABET = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
UNGUESSED = ALPHABET
GAME_OVER = False
screen = pygame.display.set_mode((WIDTH, HEIGHT))
```

- Calculate the size of each square for letters and set up the Pygame window.
- Initialize variables to store user input, guesses, the alphabet, unguessed letters, and game state.

```
# Main game loop
running = True
while running:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False
        elif event.type == pygame.KEYDOWN:
            if event.key == pygame.K_ESCAPE:
                running = False
            elif event.key == pygame.K_BACKSPACE:
                if INPUT:
                    INPUT = INPUT[:-1]
            elif event.key == pygame.K_RETURN:
                if len(INPUT) == 5 and INPUT in DICT_GUESSING:
                    GUESSES.append(INPUT)
                    GAME_OVER = INPUT == ANSWER
                    INPUT = ""
            elif event.key == pygame.K_SPACE:
                GAME_OVER = False
                ANSWER = random.choice(DICT_ANSWERS)
                GUESSES = []
                INPUT = ""
            elif len(INPUT) < 5 and not GAME_OVER:
                INPUT = INPUT + event.unicode.upper()
```

- `running` is a boolean variable that determines whether the game loop should continue running.
- The loop continues as long as `running` is `True`.
- The loop iterates over all the events that have occurred since the last iteration.
- If the event type is `pygame.QUIT`, it sets `running` to `False`, allowing the game loop to exit.
- If the event type is `pygame.KEYDOWN`, it further checks for specific key events (e.g., quitting the game, backspace, enter, etc.)

```
    y = T_MARGIN
    for i in range(6):
        x = LR_MARGIN
        for j in range(5):
            square = pygame.Rect(x, y, SQ_SIZE, SQ_SIZE)
            pygame.draw.rect(screen, GREY, square, width=2, border_radius=3)

            if i < len(GUESSES):
                color = GREEN if GUESSES[i][j] == ANSWER[j] else YELLOW if
GUESSES[i][j] in ANSWER else GREY
                pygame.draw.rect(screen, color, square, border_radius=3)
                letter = FONT.render(GUESSES[i][j], False, (255, 255, 255))
                surface = letter.get_rect(center=(x + SQ_SIZE // 2, y +
SQ_SIZE // 2))
                screen.blit(letter, surface)

            if i == len(GUESSES) and j < len(INPUT):
                letter = FONT.render(INPUT[j], False, GREY)
                surface = letter.get_rect(center=(x + SQ_SIZE // 2, y +
SQ_SIZE // 2))
                screen.blit(letter, surface)

            x += SQ_SIZE + MARGIN
        y += SQ_SIZE + MARGIN
```

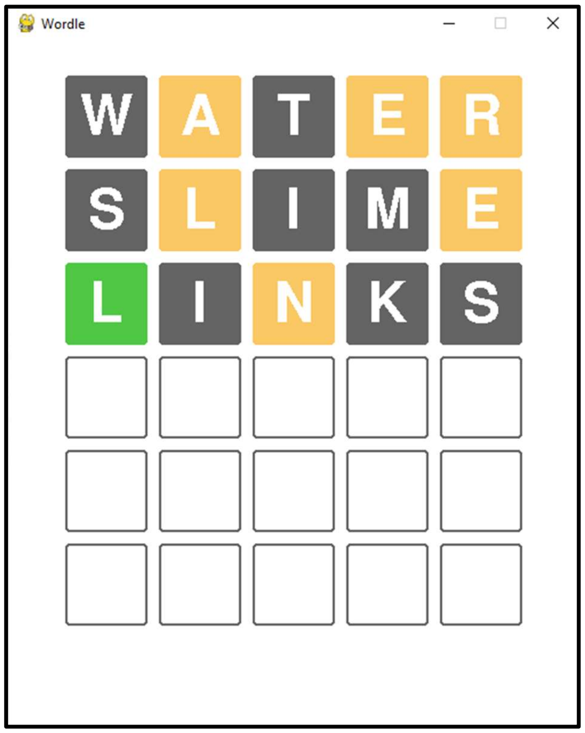
This section is responsible for rendering the current state of the game. It includes drawing unguessed letters, drawing guesses made by the player, and handling the game over condition.

```
    if len(GUESSES) == 6 and GUESSES[5] != ANSWER:
        GAME_OVER = True
        letters = FONT.render(ANSWER, False, GREY)
        surface = letters.get_rect(center=(WIDTH // 2, HEIGHT - B_MARGIN //
2 - MARGIN))
        screen.blit(letters, surface)
    pygame.display.flip()
pygame.quit()
```

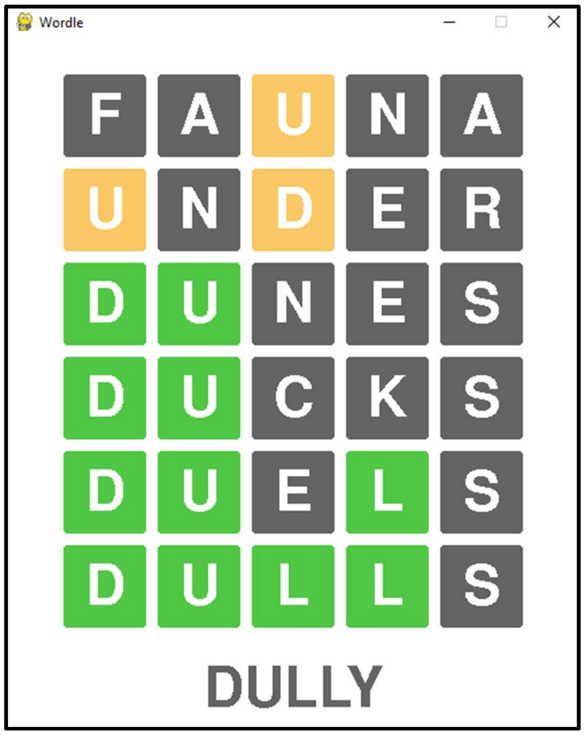
This section draws the correct answer when the game is over and updates the screen.

OUTPUT:

Game window when playing:



Game window at the end of a game:



CONCLUSION:

Throughout the development of this Wordle game, I acquired valuable insights into the realm of game development, refining my programming skills and expanding my comprehension of user interaction and feedback. This endeavor enabled me to implement essential game mechanics, emphasizing the significance of clear code structure and proficient problem-solving. Overcoming challenges in user interface design and integrating Pygame functionalities further enhanced my expertise in crafting captivating and interactive applications. In essence, this project provided an extensive learning experience, nurturing creativity, resilience, and a heightened understanding of the complexities inherent in software development.