



HANGMAN-EL AHORCADO PROJECT

EL IMPACTO DE LAS NUEVAS TECNOLOGÍAS EN LA SOCIEDAD: VISUALIZACIÓN DEL FUTURO.

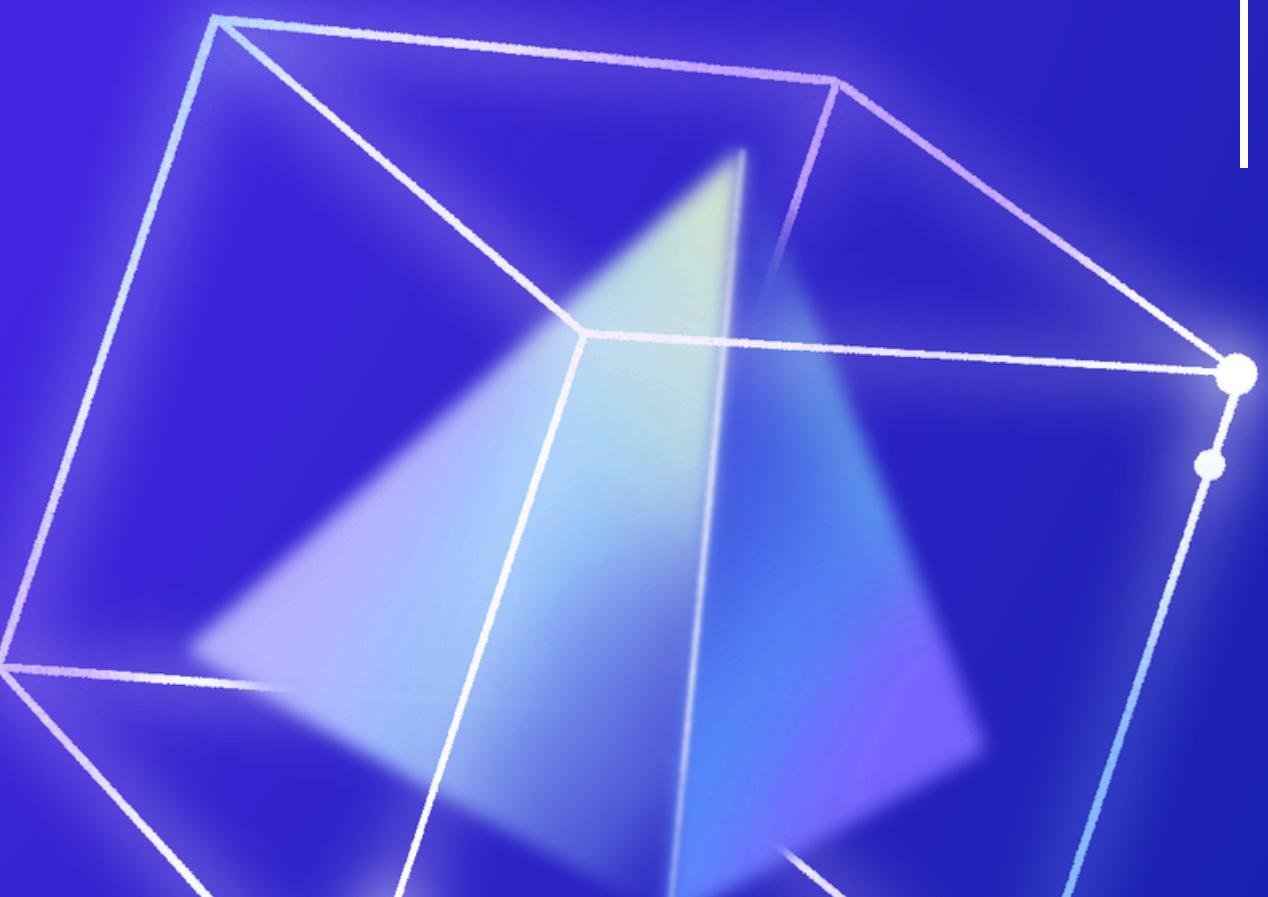
By Samuel Riera

30-06-2024



TABLE OF CONTENTS

- 01 OBJETIVO DEL PROGRAMA**
- 02 PRINCIPALES FUNCIONALIDADES DEL CODIGO**
- 03 CONCLUSIONES A PARTIR DE MI EXPERIENCIA**



OBJETIVO

Este proyecto me permitió aprender y aplicar habilidades clave como el desarrollo de interfaces gráficas con Tkinter, la programación orientada a objetos para estructurar el juego, y el manejo de archivos externos para obtener datos. Además, practique la lógica del juego, la resolución de problemas y la mejora de la experiencia de usuario a través de una interfaz intuitiva y mensajes claros. En conjunto, este proyecto proporcionó una valiosa experiencia práctica en programación.

FUNCIONES PRINCIPALES:

1. choose_word():
 - Esta función lee el archivo "palabras.txt" y elige aleatoriamente una palabra de la lista de palabras contenidas en el archivo.

```
5  def choose_word():
6      with open("palabras.txt", "r") as file:
7          words = file.read().splitlines()
8      return random.choice(words)
```



2. `display_word(word, guessed_letters):`

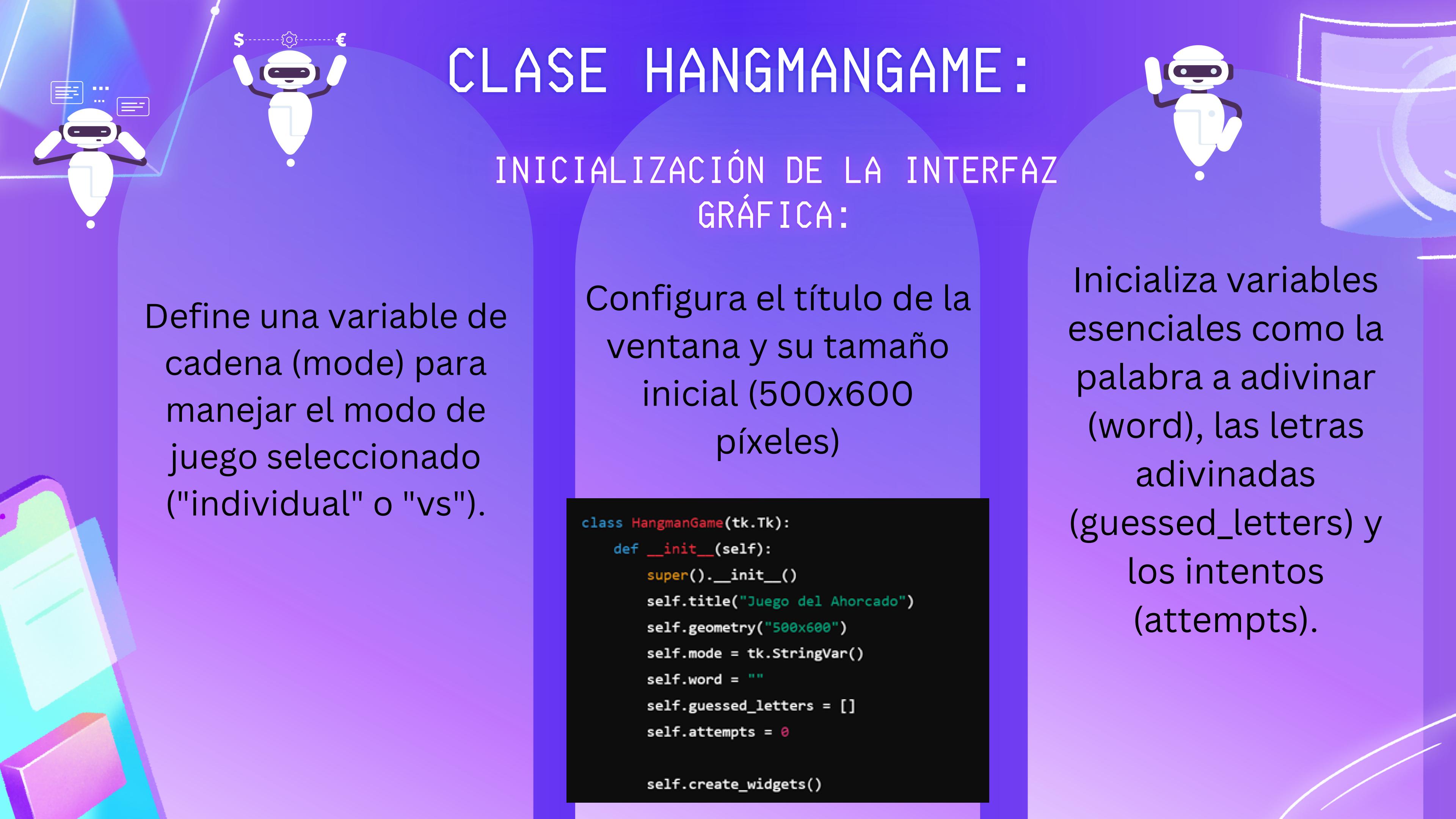
Genera y devuelve una representación visual de la palabra a adivinar, donde las letras adivinadas se muestran y las no adivinadas se representan con guiones bajos.

```
● 10  display_word(word, guessed_letters):
11  displayed_word = ""
12  for letter in word:
13      if letter in guessed_letters:
14          displayed_word += letter + " "
15      else:
16          displayed_word += "_"
17  return displayed_word.strip()
```

DISPLAY_HANGMAN(ATTEMPTS):

- Retorna una representación visual del muñeco del ahorcado, progresivamente añadiendo partes del cuerpo con cada intento fallido.

```
20  hangman_stages = [
21      """
22      -----
23      |   |
24      |
25      |
26      |
27      |
28      -----
29      """
30      """
31      -----
32      |   |
33      O
34      |
35      |
36      |
37      -----
38      """
39      """
40      -----
41      |   |
42      O
43      |
44      |
45
46
47
48      """
49      -----
50      |   |
51      O
52      /|\\
53      |
54      -----
55      """
56      ,,
57      """
58      -----
59      |   |
60      O
61      /|\\
62      |
63      |
64      -----
65      """
66      """
67      -----
68      |   |
69      O
70      /|\\
71      /
72
73
74      """
75      """
76      -----
77      |   |
78      O
79      /|\\
80      /\\
81      |
82      -----
83      """
84 ]
```



Define una variable de cadena (mode) para manejar el modo de juego seleccionado ("individual" o "vs").

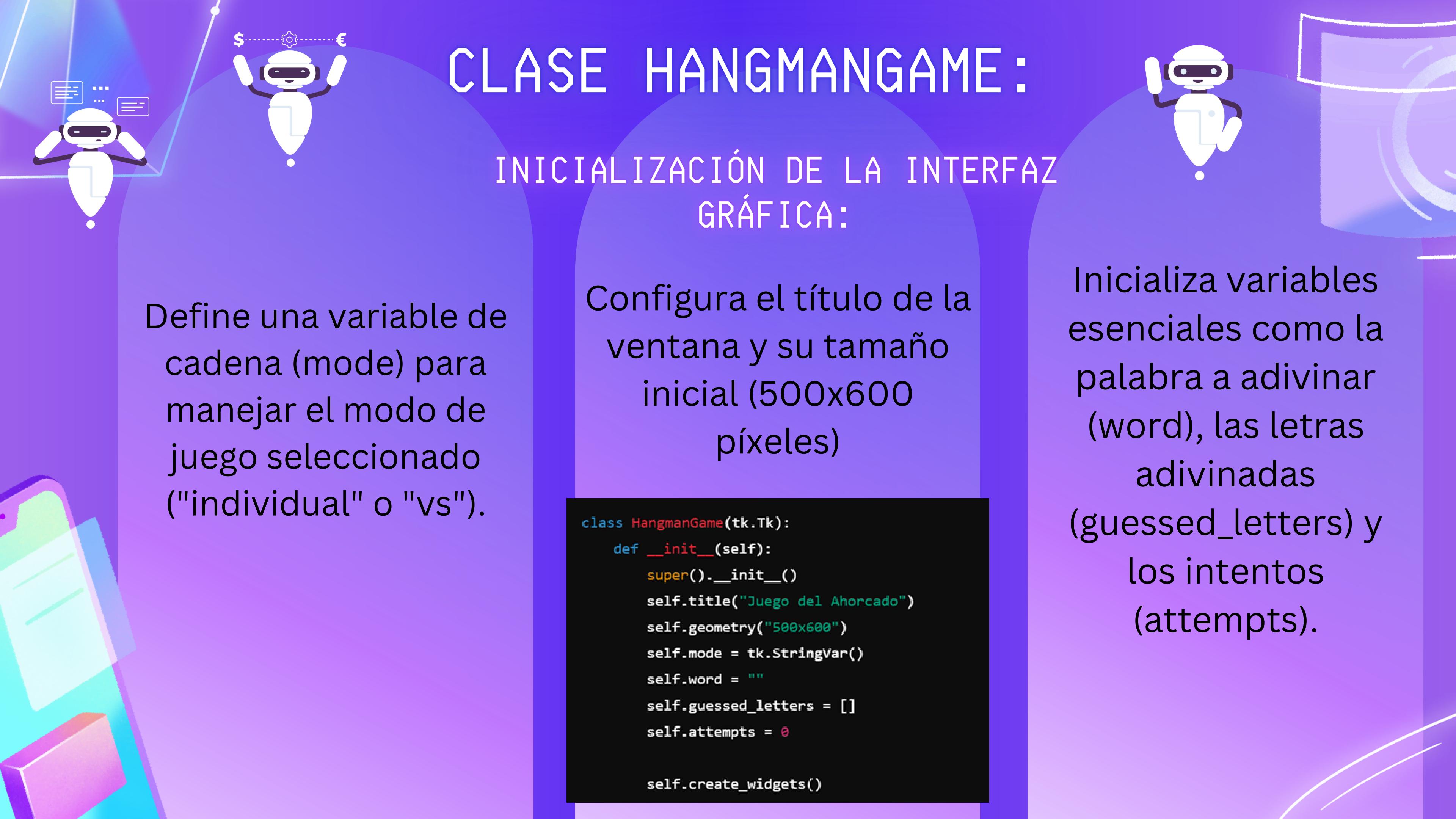
CLASE HANGMANGAME:

INICIALIZACIÓN DE LA INTERFAZ GRÁFICA:

Configura el título de la ventana y su tamaño inicial (500x600 píxeles)

```
class HangmanGame(tk.Tk):
    def __init__(self):
        super().__init__()
        self.title("Juego del Ahorcado")
        self.geometry("500x600")
        self.mode = tk.StringVar()
        self.word = ""
        self.guessed_letters = []
        self.attempts = 0

        self.create_widgets()
```



Inicializa variables esenciales como la palabra a adivinar (word), las letras adivinadas (guessed_letters) y los intentos (attempts).

Creación de Widgets:

- Crea un marco (mode_frame) para los botones de selección de modo de juego.
- Añade botones de radio para seleccionar entre los modos "Individual" y "VS".
- Inicializa etiquetas para mostrar el muñeco del ahorcado, la palabra a adivinar, los intentos restantes, una entrada para adivinar y botones para adivinar y comenzar el juego.

```
def create_widgets(self):  
    mode_frame = tk.Frame(self)  
    mode_frame.pack(pady=10)  
  
    tk.Label(mode_frame, text="Modo de Juego: ", font=("Courier", 14)).pack()  
    self.mode = tk.StringVar()  
    tk.Radiobutton(mode_frame, text="Individual", variable=self.mode, value="individual").pack()  
    tk.Radiobutton(mode_frame, text="VS", variable=self.mode, value="vs").pack()  
  
    self.hangman_label = tk.Label(mode_frame, text="Ahorcado", font=("Times New Roman", 30))  
    self.hangman_label.pack()  
  
    self.word_label = tk.Label(mode_frame, text="Palabra a Adivinar: ", font=("Times New Roman", 14))  
    self.word_label.pack()  
  
    self.guess_label = tk.Label(mode_frame, text="Introducir Letra: ", font=("Times New Roman", 14))  
    self.guess_label.pack()  
  
    self.guess_entry = tk.Entry(mode_frame, width=10, font=("Times New Roman", 14))  
    self.guess_entry.pack()  
  
    self.start_button = tk.Button(mode_frame, text="Comenzar Juego", font=("Times New Roman", 14), command=self.start_game)  
    self.start_button.pack()
```

Método create_widgets:

- Configuración de Botones y Etiquetas:
- Crea botones para iniciar el juego y para adivinar letras.
- Configura etiquetas para mostrar el estado del muñeco del ahorcado, la palabra a adivinar y los intentos restantes.

```
def create_widgets(self):  
    mode_frame = tk.Frame(self)  
    mode_frame.pack(pady=10)  
    tk.Label(mode_frame, text="Modo de Juego: ", font=("Courier",  
    tk.Radiobutton(mode_frame, text="Individual", variable=self.mode,  
    tk.Radiobutton(mode_frame, text="VS", variable=self.mode, value=2)  
    self.mode.set("individual")
```

MÉTODO START_GAME:

- Inicio del Juego:
 - Verifica el modo de juego seleccionado.
 - Si es modo "VS", solicita al Jugador 1 que ingrese una palabra secreta.
 - Si es modo "Individual", elige una palabra aleatoria utilizando la función choose_word().
 - Reinicia las variables para iniciar un nuevo juego y actualiza la interfaz gráfica.

```
125     def start_game(self):  
126         if self.mode.get() == "vs":  
127             player1_word = simpledialog.askstring("Jugador 1", "Jugador 1, por favor ingrese su palabra:",  
128             if not player1_word:  
129                 messagebox.showerror("Error", "Por favor ingrese una palabra válida.")  
130                 return  
131             self.word = player1_word.lower()  
132         else:  
133             self.word = choose_word()
```



Método guess_letter:

- Adivinanza de Letras o Palabras:
 - Captura la letra o palabra ingresada por el jugador.
 - Valida la entrada para asegurarse de que no esté vacía y no se haya intentado antes.
 - Compara la entrada con la palabra secreta y actualiza la pantalla en consecuencia.
 - Muestra mensajes de felicitación si se completa la palabra o de derrota si se agotan los intentos.

```
def guess_letter(self):
    guess = self.guess_entry.get().lower()
    self.guess_entry.delete(0, tk.END)

    if not guess:
        messagebox.showwarning("Advertencia", "Por favor, introduce una letra o una palabra.")
        return

    if guess in self.guessed_letters:
        messagebox.showwarning("Advertencia", "Ya has intentado esta letra. Prueba con otra.")
        return

    self.guessed_letters.append(guess)

    if guess == self.word:
        messagebox.showinfo("Felicitaciones", "¡Has ganado!")
        self.reset_game()
    elif len(guess) == 1 and guess in self.word:
        self.update_display()
        if all(letter in self.guessed_letters for letter in self.word):
            messagebox.showinfo("Felicitaciones", "¡Has ganado!")
            self.reset_game()
    else:
        self.attempts += 1
        self.update_display()
        if self.attempts == 6:
            messagebox.showinfo("Game Over", f"¡No! ¡Te has quedado sin intentos!\nLa palabra era {self.word}")
            self.reset_game()
```



Métodos update_display y reset_game:

ACTUALIZACIÓN Y REINICIO DEL JUEGO

- **update_display:** Actualiza las etiquetas de la interfaz para reflejar el estado actual del juego (muñeco del ahorcado, palabra a adivinar, intentos restantes).
- **reset_game:** Reinicia todas las variables y etiquetas para iniciar un nuevo juego, permitiendo que el jugador elija otro modo o inicie una nueva partida.

```
def update_display(self):  
    self.hangman_label.config(text=display_hangman(self.attempts))  
    self.word_label.config(text=display_word(self.word, self.guessed_letters))  
    self.attempts_label.config(text=f"Intentos restantes: {6 - self.attempts}")
```

```
def reset_game(self):  
    self.word = ""  
    self.guessed_letters = []  
    self.attempts = 0  
    self.word_label.config(text="")  
    self.hangman_label.config(text=display_hangman(self.attempts))  
    self.attempts_label.config(text=f"Intentos restantes: {6 - self.attempts}")
```





GRACIAS POR SU ATENCION

LINK GITHUB



<https://github.com/sriera21/Juego-del-ahorcado>

